Brandt's fully private auction protocol revisited

Jannik Dreier, Jean-Guillaume Dumas, Pascal Lafourcade

Brandt's Fully Private Auction Protocol Revisited

Jannik Dreier[1,2,3], Jean-Guillaume Dumas[4], and Pascal Lafourcade[5]

[1]Université de Lorraine, Loria, UMR 7503, F-54506 Vandoeuvre-lès-Nancy, France
[2]Inria, F-54600 Villers-lès-Nancy, France
[3]CNRS, Loria, UMR 7503, F-54506 Vandoeuvre-lès-Nancy, France, E-mail: jannik.dreier@loria.fr
[4]Université Grenoble Alpes, CNRS, Laboratoire Jean Kuntzmann, 51 av. des Mathématiques, BP 53X, 38041 Grenoble Cedex 9, France, E-mail: jean-guillaume.dumas@imag.fr
[5]University Clermont Auvergne, LIMOS, Campus Universitaire des Cézeaux, BP 86, 63172 Aubière Cedex, France, E-mail: pascal.lafourcade@udamail.fr

November 30, 2015

Abstract

Auctions have a long history, having been recorded as early as 500 B.C. [Kri02]. Nowadays, electronic auctions have been a great success and are increasingly used in various applications, including high performance computing [BAGS02]. Many cryptographic protocols have been proposed to address the various security requirements of these electronic transactions, in particular to ensure privacy. Brandt [Bra06] developed a protocol that computes the winner using homomorphic operations on a distributed ElGamal encryption of the bids. He claimed that it ensures full privacy of the bidders, i.e. no information apart from the winner and the winning price is leaked. We first show that this protocol – when using malleable interactive zero-knowledge proofs – is vulnerable to attacks by dishonest bidders. Such bidders can manipulate the publicly available data in a way that allows the seller to deduce all participants' bids. We provide an efficient parallelized implementation of the protocol and the attack to show its practicality. Additionally we discuss some issues with verifiability as well as attacks on non-repudiation, fairness and the privacy of individual bidders exploiting authentication problems.

1 Introduction

Auctions are a simple method to sell goods and services. Typically a *seller* offers a good or a service, and the *bidders* make offers. Depending on the type of auction, the offers might be sent using sealed envelopes which are opened simultaneously to determine the winner (the "sealed-bid" auction), or an *auctioneer* could announce prices decreasingly until one bidder is willing to pay the announced price (the "dutch auction"). Additionally there might be several rounds, or offers might be announced publicly directly (the "English" or "shout-out" auction). The winner usually is the bidder submitting the highest bid, but in some cases he might only have to pay the second highest offer as a price (the "second-price"- or "Vickrey"-Auction). In general a bidder wants to win the auction at the lowest possible price, and the seller wants to sell his good at the highest possible price. For more information on different auction methods see [Kri02]. To address this huge variety of possible auction settings and to achieve different security and efficiency properties numerous protocols have been developed, e.g. [Bra06, CPS07, NPS99, OM01, PBDV02, SSS02, Sak00] and references therein.

One of the many applications of auctions lies is in the context of distributed scheduling of jobs for high performance computing grids [BAGS02, AGG+05]. In this and other applications, one of the key security requirements of electronic auction (e-Auction) protocols is privacy, i.e. the bids of losing bidders remain private as they can contain sensitive information. For example, in the case of a grid shared between multiple companies, the job meta-data (length, resource requirements, as well as the offered payment) can leak sensitive information to competitors about the nature, content or importance of the computation: if the job is very important, a company might be willing to pay more even for a small job, or changes in the job types might indicate that the company is developing new computation methods. Moreover, the offered prices might leak information about the bidders' strategies (in particular in multi-round auctions) or their financial status.

Brandt proposed a first-price sealed-bid auction protocol [Bra06, Bra03, Bra02] and claimed that it is fully private, i.e. it leaks no information apart from the winner, the winning bid, and what can be deduced from these two facts (for example that the other bids were lower).

Our Contributions. The protocol is based on an algorithm that computes the winner using bids encoded as bit vectors. In this paper we show that the implementation using the homomorphic property of a distributed Elgamal encryption proposed in the original paper suffers from a weakness. In fact, we prove that any two different inputs (i.e. different bids) result in different outcome values, which are only hidden using random values. We show how a dishonest participant can re-

move this random noise, if malleable interactive zero-knowledge proofs are used. The seller can then efficiently compute the bids of all bidders, hence completely breaking privacy. We provide a parallel implementation for the protocol and the attack, and show that the computations are efficient in practice. We also discuss two problems with verifiability, and how the lack of authentication enables attacks on privacy even if the above attack is prevented via non-malleable non-interactive proofs. Additionally we show attacks on non-repudiation and fairness, and propose solutions to all discovered flaws in order to recover a fully resistant protocol.

A preliminary version of this paper was presented in [DDL13], with a theoretical attack of complexity $O(n^2k^2)$ where n is the number of bidders and k the number of possible bids. In this paper we correct an error in the attack given in [DDL13] and present an improved attack of complexity only $O(nk)$. We also give a detailed description of the new efficient algorithm, and present a practical performance evaluation on realistic inputs: With thousands of different bids and bidders, we were able to break the privacy of the original protocol in a few seconds on a 32 cores shared memory machine.

Outline. In the next section, we recall the protocol of Brandt. Then, in the following sections, we present our attacks in several steps. In Section 3, we first study the protocol using interactive zero-knowledge proofs and without random noise. Then we show how a dishonest participant can remove the noise, thus mount the attack on the protocol with noise, and discuss countermeasures. Finally, in Section 4, we discuss verifiability and in Section 5 we discuss attacks on fairness, non-repudiation and privacy exploiting the lack of authentication.

2 The Protocol

The protocol of Brandt [Bra06] was designed to ensure full privacy in a completely distributed way. It exploits the homomorphic properties of a distributed El-Gamal encryption scheme [EG85] for a secure multi-party computation of the winner. Then it uses zero-knowledge proofs of knowledge of discrete logarithms to ensure correctness of the bids while preserving privacy. We first give a high level description of the protocol and then present details on its main cryptographic primitives.

2.1 Informal Description

The participating n bidders and the seller communicate essentially using broadcast messages. The latter can for example be implemented using a bulletin board, i.e.

an append-only memory accessible to everybody. The bids are encoded as k-bit-vectors where each entry corresponds to a price. If the bidder a wants to bid the price bid_a, all entries will be 1, except the entry bid_a which will be Y (a public constant). Each entry of the vector is then encrypted separately using a n-out-of-n-encryption scheme set up by all bidders. The bidders use multiplications of the encrypted values to compute one value v_{aj} for each price, by exploiting the homomorphic property of the encryption scheme. Each one of this values (v_{aj}) is 1 if the bidder a wins at price j, and is a random number otherwise. The decryption of the final values takes place in a distributed way to ensure that nobody can access intermediate values.

2.2 Mathematical Description (Brandt [Bra06])

Let $\mathbb{G}_q$ be a multiplicative subgroup of order q, prime, and g a generator of the group. We consider that $i, h \in \{1, \ldots, n\}$, $j, bid_a \in \{1, \ldots, k\}$ (where bid_a is the bid chosen by the bidder with index a), $Y \in \mathbb{G}_q \setminus \{1\}$. More precisely, the n bidders execute the following five steps of the Brandt's protocol [Bra06]:

1. **Key Generation**
   Each bidder a, whose bidding price is bid_a among $\{1, \ldots, k\}$ does the following:
   - chooses a secret $x_a \in \mathbb{Z}/q\mathbb{Z}$
   - chooses randomly m_{ij}^a and $r_{aj} \in \mathbb{Z}/q\mathbb{Z}$ for each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, k\}$.
   - publishes $y_a = g^{x_a}$ and proves the knowledge of y_a's discrete logarithm
   - using all the published y_i, then computes $y = \prod_{i=1}^{n} y_i$.
2. **Bid Encryption**
   Each bidder a knows the public value Y and
   - for all $j \in \{1, \ldots, k\}$ sets $b_{aj} = \begin{cases} Y & \text{if } j = bid_a \\ 1 & \text{otherwise} \end{cases}$
   - publishes $\alpha_{aj} = b_{aj} \cdot y^{r_{aj}}$ and $\beta_{aj} = g^{r_{aj}}$ for each j
   - proves that for all j, $\log_g(\beta_{aj})$ equals $\log_y(\alpha_{aj})$ or $\log_y\left(\frac{\alpha_{aj}}{Y}\right)$, and that

$$\log_y\left(\frac{\prod_{j=1}^{k} \alpha_{aj}}{Y}\right) = \log_g\left(\prod_{j=1}^{k} \beta_{aj}\right)$$

3. **Outcome Computation**

   - Each bidder a computes and publishes for all i and j:

$$\gamma_{ij}^a = \left(\left(\prod_{h=1}^{n} \prod_{d=j+1}^{k} \alpha_{hd} \right) \cdot \left(\prod_{d=1}^{j-1} \alpha_{id} \right) \cdot \left(\prod_{h=1}^{i-1} \alpha_{hj} \right) \right)^{m_{ij}^a}$$

$$\delta_{ij}^a = \left(\left(\prod_{h=1}^{n} \prod_{d=j+1}^{k} \beta_{hd} \right) \cdot \left(\prod_{d=1}^{j-1} \beta_{id} \right) \cdot \left(\prod_{h=1}^{i-1} \beta_{hj} \right) \right)^{m_{ij}^a}$$

     and proves its correctness.

4. **Outcome Decryption**

   - Each bidder a sends $\phi_{ij}^a = (\prod_{h=1}^{n} \delta_{ij}^h)^{x_a}$ for each i and j to the seller and proves its correctness. After having received all values, the seller publishes ϕ_{ij}^h for all i, j, and $h \neq i$.

5. **Winner determination**

   - Everybody can now compute $v_{aj} = \dfrac{\prod_{i=1}^{n} \gamma_{aj}^i}{\prod_{i=1}^{n} \phi_{aj}^i}$ for each j.
   - If $v_{aw} = 1$ for some w, then the bidder a wins the auction at price p_w.

2.3 Malleable proofs of knowledge and discrete logarithms

In the original paper [Bra06] the author suggests using zero-knowledge proofs of knowledge to protect against active adversaries. The basic protocols he proposes are interactive and malleable, but can be converted into non-interactive proofs using the Fiat-Shamir heuristic [FS86], as advised by the author. We first recall the general idea of such proofs, then we expose the man-in-the-middle attacks on the interactive version of the zero-knowledge proofs, which we will use as part of our first attack on Brandt's protocol in Section 3.1.

Let PDL denote a *proof of knowledge of a discrete logarithm*. A first scheme for PDL was developed in 1986 by Chaum et al. [CEvdGP86]. In the original auction paper [Bra06] Brandt proposes to use a non-interactive variant of PDL as developed by Schnorr [Sch91], which are malleable. Unfortunately, interactive malleable PDL are subject to man-in-the-middle attacks [Kat02]. Many PDL's are obtained via three-move structure protocols (commitment, challenge and response) called Σ-protocols. We first recall the classic Σ-protocol for PDL, on a group with

generator g and order q [BCM05, BDPW89, CEvdG87]: Peggy and Victor know v and g, but only Peggy knows x, so that $v = g^x$. She can prove this fact, without revealing x, by executing the following protocol:

1. (commitment) Peggy chooses r at random and sends $z = g^r$ to Victor.
2. (challenge) Victor chooses a challenge c at random and sends it to Peggy.
3. (response) Peggy sends $s = (r + c \cdot x) \mod q$ to Victor.
4. Victor checks that $g^s = z \cdot v^c$.

2.3.1 Generalization of Katz' man-in-the-middle attack on interactive PDL

Suppose Peggy possesses some secret discrete logarithm x. We present here first the man-in-the-middle attack of [Kat02], and then generalize it to any affine transform of a secret discrete logarithm. In this attack, an attacker can pretend to have knowledge of any affine combination of the secret x, even providing the associated proof of knowledge, without breaking the discrete logarithm. To prove this possession to say Victor, the attacker will start an interactive proof knowledge session with Peggy and another one with Victor. The attacker will transform Peggy's outputs and forward Victor's challenges to her. The idea of [Kat02] is to use the proof of possession of Peggy's x, to prove possession of $1 - x$ to Victor. Indeed to prove for instance possession of just x to Victor, an attacker would only have to forward Peggy's messages to Victor and Victor's messages to Peggy. The idea of the attack is similar, except that one needs to modify the messages of Peggy. We show the example of $1 - x$ in Figure 1 since it is used in Section 3.4 to mount our attack. Upon demand by Victor to prove knowledge of $1 - x$, Mallory, the man-in-the-middle, simply starts a proof of knowledge of x with Peggy. Peggy chooses a random exponent r and sends the commitment $z = g^r$ to Mallory. Mallory simply inverts z and sends $y = z^{-1}$ to Victor. Then Victor presents a challenge c that Mallory simply forwards without modification to Peggy. Finally Peggy sends a response s that Mallory combines with c, as $u = c - s$, to provide a correct answer to Victor. This is summarized in Figure 1. Victor is convinced by Mallory's proof since we have $g^s = g^{r+c \cdot x} = g^r g^{x \cdot c} = z v^c$ and $g^u = g^{c-s} = g^{c-(r+c \cdot x)} = g^{-r} g^{(1-x) \cdot c} = z^{-1}(g v^{-1})^c = y w^c$.

Now we show in Lemma 1 that it is possible to adapt the attack to make it work in the generic settings of [BDPW89, Mau09] or of Σ-protocols [CD98]. We use this generalization to prevent possible countermeasures of our first attack in Section 3.7.

We let $f : \Gamma \to \Omega$ denote a one way homomorphic function between two commutative groups $(\Gamma, +)$ and $(\Omega, \times)$. For an integral value α, $\alpha \cdot x \in \Gamma$ (resp.

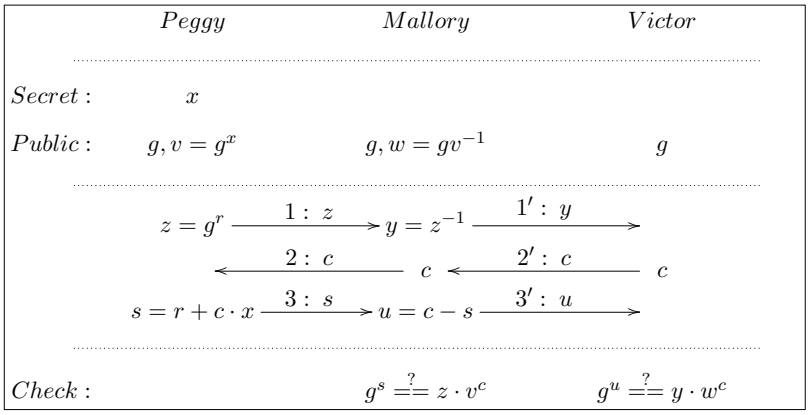


Figure 1: [Kat02] Man-in-the-middle PDL of $1 - x$, with x an unknown discrete logarithm.

$y^\alpha \in \Omega$) denotes α applications of the group law $+$ (resp. $\times$). For a secret $x \in \Gamma$, and any $(h, \alpha, \beta) \in \Gamma \times \mathbb{Z}^2$, the attacker can build a proof of possession of $\alpha \cdot h + \beta \cdot x$. In the setting of the example of Figure 1, we used $f(x) = g^x$, $h = 1$, $\alpha = 1$ and $\beta = -1$.

In the general case also, upon demand of proof by Victor, Mallory starts a proof with Peggy. The secret of Peggy is x, and the associated witness v is $v = f(x)$. Then Mallory wants to prove that his witness w corresponds to any combination of x with a logarithm h that he knows. With only public knowledge and his chosen $(h, \alpha, \beta) \in \Gamma \times \mathbb{Z}^2$, Mallory is able to compute $w = f(h)^\alpha \cdot v^\beta$. For the proof of knowledge, Mallory still modifies the commitment $z = f(r)$ of Peggy to $y = z^\beta$. Mallory forwards the challenge c of Victor without modification. Finally Mallory transforms the response s of Peggy, still with only public knowledge and his chosen $(h, \alpha, \beta) \in \Gamma \times \mathbb{Z}^2$, as $u = c \cdot (\alpha \cdot h) + \beta \cdot s$. We summarize this general attack on Figure 2.

Lemma 1. *In the generalized man-in-the-middle attack described above (cf. Figure 2) on the interactive proof of knowledge of a discrete logarithm, Victor is convinced by Mallory's proof of knowledge of* $\alpha \cdot h + \beta \cdot x$.

Proof. Indeed,

$$u = c \cdot (\alpha \cdot h) + \beta \cdot s = c \cdot (\alpha \cdot h) + \beta \cdot (r + c \cdot x) = \beta \cdot r + c \cdot (\alpha \cdot h + \beta \cdot x). \quad (1)$$

Now, since $z = f(r)$, $y = z^\beta$, $v = f(x)$ and $f(h)^\alpha \times v^\beta = w$, the latter Equation (1) proves in turn that

$$f(u) = f(r)^\beta \times f(\alpha \cdot h + \beta \cdot x)^c = z^\beta \times (f(h)^\alpha \times f(x)^\beta)^c = y \times w^c. \quad (2)$$

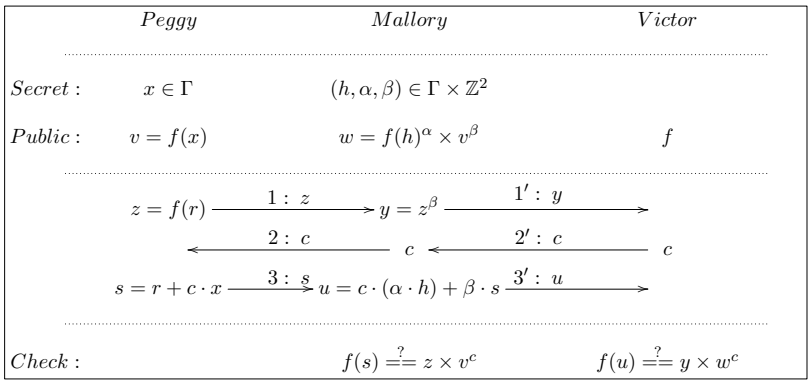


Figure 2: Man-in-the-middle attack proving knowledge of any affine transform of a secret discrete logarithm in the generic setting.

Now Victor has to verify the commitment-challenge-response (y, c, u) of Mallory for his witness w. Then Victor needs to checks whether $f(u)$ corresponds to $y \times w^c$, which is the case as shown by the latter Equation (2). □

2.3.2 Generalizations to equality of discrete logarithms

We let EQDL denote a *proof of equality of several discrete logarithms*. Any PDL can in general easily be transformed to an EQDL by applying it k times on the same witness. It is often more efficient to combine the application in one as in [CP92, CMW10], or more generally as composition of Σ-protocols, here with two logarithms and two generators g_1 and g_2. Peggy wants to prove that she knows x such that $v = g_1^x$ and $w = g_2^x$:

1. Peggy chooses r at random and sends $\lambda = g_1^r$ and $\mu = g_2^r$ to Victor.
2. Victor chooses a challenge c at random and sends it to Peggy.
3. Peggy computes $s = (r + c \cdot x) \mod q$ and sends it to Victor.
4. Victor tests if $g_1^s = \lambda \cdot v^c$ and $g_2^s = \mu \cdot w^c$.

This protocol remains malleable, and the previous attacks are still valid since the response remains of the form $r + c \cdot x$.

2.3.3 Countermeasures

Direct countermeasures to the above attacks are to use non-interactive and/or non-malleable proofs:

- An interactive protocol can be converted into a non-interactive one using the Fiat-Shamir heuristic [FS86]. In this case the challenge is computed as a cryptographic hash of all previous values, and is not submitted by the verifier. This makes it impossible for Mallory to choose the challenge according to his needs (since for him the previous values and hence the hash are different), which prevents the attack.

- Also the first PDL by [CEvdGP86] uses bit-flipping, and more generally non-malleable protocols like [FF09] could be used.

We will show in the following that if the proofs proposed in the original paper are not converted into non-interactive proofs, there is an attack on privacy. Note that even if non-interactive non-malleable zero-knowledge proofs are used, a malicious attacker in control of the network can nonetheless recover any bidder's bid as the messages are not authenticated, as we show in Section 5.

3 Attacking the fully private computations

The first attack we present uses some algebraic properties of the computations performed during the protocol execution. More precisely, it relies on the fact that the outcome computation function is invertible, apart from some added random noise. In the following, we prove that the function without random noise is injective, and give an algorithm to invert it. Then we show that an attacker can actually remove the random noise, and that this is unnoticeable to the other participants if malleable zero-knowledge proofs are used. Finally we give a detailed description of the attack, and discuss countermeasures.

3.1 Analysis of the outcome computation

The idea is to analyze the computations done in Step 3 of the protocol. Consider the following example with three bidders and three possible prices. Then the first

bidder computes

$$
\begin{array}{lrrrcrcrr}
\gamma^1_{11} = (& (\alpha_{12}\cdot\alpha_{13}\cdot & \alpha_{22}\cdot\alpha_{23}\cdot & \alpha_{32}\cdot\alpha_{33}) & \cdot & (1) & \cdot & (1) &)^{m^1_{11}} \\
\gamma^1_{12} = (& (\alpha_{13}\cdot & \alpha_{23}\cdot & \alpha_{33}) & \cdot & (\alpha_{11}) & \cdot & (1) &)^{m^1_{12}} \\
\gamma^1_{13} = (& & & (1) & \cdot & (\alpha_{11}\cdot\alpha_{12}) & \cdot & (1) &)^{m^1_{13}} \\
\gamma^1_{21} = (& (\alpha_{12}\cdot\alpha_{13}\cdot & \alpha_{22}\cdot\alpha_{23}\cdot & \alpha_{32}\cdot\alpha_{33}) & \cdot & (1) & \cdot & (\alpha_{11}) &)^{m^1_{21}} \\
\boldsymbol{\gamma^1_{22}} = (& (\boldsymbol{\alpha_{13}}\cdot & \boldsymbol{\alpha_{23}}\cdot & \boldsymbol{\alpha_{33}}) & \cdot & (\boldsymbol{\alpha_{21}}) & \cdot & (\boldsymbol{\alpha_{12}}) &)^{\mathbf{m^1_{22}}} \\
\gamma^1_{23} = (& & & (1) & \cdot & (\alpha_{21}\cdot\alpha_{22}) & \cdot & (\alpha_{13}) &)^{m^1_{23}} \\
\gamma^1_{31} = (& (\alpha_{12}\cdot\alpha_{13}\cdot & \alpha_{22}\cdot\alpha_{23}\cdot & \alpha_{32}\cdot\alpha_{33}) & \cdot & (1) & \cdot & (\alpha_{11}\cdot\alpha_{21}) &)^{m^1_{31}} \\
\gamma^1_{32} = (& (\alpha_{13}\cdot & \alpha_{23}\cdot & \alpha_{33}) & \cdot & (\alpha_{31}) & \cdot & (\alpha_{12}\cdot\alpha_{22}) &)^{m^1_{32}} \\
\gamma^1_{33} = (& & & (1) & \cdot & (\alpha_{31}\cdot\alpha_{32}) & \cdot & (\alpha_{13}\cdot\alpha_{23}) &)^{m^1_{33}}
\end{array}
$$

The second and third bidder do the same computations, but using different random values m^a_{ij}. Since each α_{ij} is either the encryption of 1 or Y, for example the value γ^1_{22} will be an encryption of 1 only if

- nobody submitted a higher bid (the first block) and
- bidder 2 did not bid a lower bid (the second block) and
- no bidder with a lower index submitted the same bid (the third block).

If we ignore the exponentiation by m^a_{ij}, each γ^a_{ij} is the encryption of the product of several b_{ij}'s. Each b_{ij} can be either 1 or Y, hence $(\gamma^a_{ij})^{-m^a_{ij}}$ will be the encryption of a value $Y^{l_{ij}}$, where $0 \leq l_{ij} \leq n$. The lower bound of l_{ij} is trivial, the upper bound follows from the observation that each α_{ij} will be used at most once, and that each bidder will encrypt Y at most once.

Assume for now that we know all l_{ij}. We show next that this is sufficient to obtain all bids. Consider the function f which takes as input the following vector[1]:

$$b = \log_Y \left(\left(\; b_{11}, \;\; \ldots, \;\; b_{1k}, \qquad b_{21}, \;\; \ldots, \;\; b_{2k}, \qquad \ldots, \qquad b_{n1}, \;\; \ldots, \;\; b_{nk} \; \right)^T \right),$$

and returns the values l_{ij}. The input vector is thus a vector of all bid-vectors, where 1 is replaced by 0 and Y by 1. Consider our above example with three bidders and three possible prices, then we have:

$$b = \log_Y \left(\left(\; b_{11}, \;\; b_{12}, \;\; b_{13}, \qquad b_{21}, \;\; b_{22}, \;\; b_{23}, \qquad b_{31}, \;\; b_{32}, \;\; b_{33} \; \right)^T \right).$$

A particular instance where bidder 1 and 3 submit price 1, and bidder 2 submits price 2 would then look as: $b = \left(\; 1, \;\; 0, \;\; 0, \qquad 0, \;\; 1, \;\; 0, \qquad 1, \;\; 0, \;\; 0 \; \right)^T$.

[1] By abuse of notation we write $log_s \left(x_1, \;\; \ldots, \;\; x_n \right)$ for $\left(log_s(x_1), \;\; \ldots, \;\; log_s(x_n) \right)$.

Hence only the factors α_{11}, α_{22} and α_{31} are encryptions of Y, all other α's are encryptions of 1. By simply counting how often the factors α_{11}, α_{22} and α_{31} show up in each equation as described above, we can compute the following result: $f(b) = \big(\ 1,\ \ 1,\ \ 1,\ \ \ \ 2,\ \ 0,\ \ 1,\ \ \ \ 2,\ \ 1,\ \ 1\ \big)^T$. Note that since we chose the input of f to be a bit-vector, we have to simply count the ones (which correspond to Y's) in particular positions in b, where the positions are determined by the factors inside γ_{ij}^a. Hence we can express f as a matrix, i.e. $f(b) = M \cdot b$ for the following matrix M:

$$f(b) = M \cdot b = \left[\begin{array}{ccc|ccc|ccc} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right] \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ \mathbf{0} \\ 1 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

To see how the matrix M is constructed, consider for example $(\gamma_{22}^a)^{-m_{22}^a} = (\alpha_{13} \cdot \alpha_{23} \cdot \alpha_{33}) \cdot (\alpha_{21}) \cdot (\alpha_{12})$ which corresponds to the **second row** in the second vertical block:

- α_{12} and α_{13}; hence the two ones at position 2 and 3 in the first horizontal block
- α_{21} and α_{23}; hence the two ones at position 1 and 3 in the second horizontal block
- α_{33}; hence the one at position 3 in the third horizontal block

More generally, we can see that each 3×3 block consists of potentially three parts:

- An upper triangular matrix representing all bigger bids.
- On the block diagonal we add a lower triangular matrix representing a lower bid by the same bidder,
- In the lower left half we add an identity matrix representing a bid at the current price by a bidder with a lower index.

This corresponds exactly to the structure of the products inside each γ_{ij}^a. It is also equivalent to formula (1) in Section 4.1.1 of the original paper [Bra06] without the random matrix R_k^*. In the following we prove that the function f is injective. We then discuss how this function can be efficiently inverted (i.e. how to compute the bids when knowing all l_{ij}'s).

3.2 Linear algebra toolbox

Let I_k be the $k \times k$ identity matrix; let L_k be a lower $k \times k$ triangular matrix with zeroes on the diagonal, ones in the lower part and zeroes elsewhere; and let U_k be an upper $k \times k$ triangular matrix with zeroes on the diagonal, ones in the upper part, and zeroes elsewhere:

$$I_k = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \quad L_k = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \quad U_k = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 0 \end{bmatrix}$$

By abuse of notation we use I, L and U to denote respectively I_k, L_k and U_k. For a $k \times k$-matrix M_k we define $(M_k)^r = M_k \cdots M_k$ (r times) and $(M_k)^0 = I_k$. Let $(e_1, \ldots, e_k)$ be the canonical basis.

The matrix U (resp L) are called *strictly upper (lower) triangular* and are defined such as for all $i \leq j$ (resp $j \leq i$) $U_{ij} = 0$. (resp $L_{ij} = 0$).

Lemma 2. *Matrices L_k and U_k are nilpotent, i.e. $(U_k)^k = 0$ and $(L_k)^k = 0$.*

Proof. We only do the proof for a strictly upper triangular matrix U, the proof for a strictly lower triangular matrix is similar. We prove by induction that $U_{ij}^k = 0$ for $i > j - k$, this implies that if U is of size $m \times m$ then for all $k \geq m$ we have $U^k = 0$. We prove this result by induction on the exponent of U:

- base case: $U_{ij}^1 = 0$ for all $i > j - 1$, since by definition of U for all $i \leq j$ we have $U_{ij} = 0$.
- induction step: We assume that $U_{ij}^{k-1} = 0$ for all $i > j - (k-1)$, we prove that $U_{ij}^k = 0$ holds for all $i > j - (k-1)$. By definition of the matrix product

we have:

$$\begin{aligned} U_{ij}^k &= (UU^{k-1})_{ij} \\ &= \sum_{r=1}^{m} U_{ir} U_{rj}^{k-1} \\ &= \sum_{r=1}^{i} U_{ir} U_{rj}^{k-1} + \sum_{r=i+1}^{m} U_{ir} U_{rj}^{k-1} \\ &= \sum_{r=1}^{i} 0 \cdot U_{rj}^{k-1} + \sum_{r=i+1}^{m} U_{ir} \cdot 0 \\ &= 0. \end{aligned}$$

In the first sum we have $U_{ir} = 0$ since $r \leq i$ and since U is strictly upper triangular. In the second sum, we apply the induction hypothesis with $r \geq i + 1 > (j - k) + 1 = j - (k - 1)$ to obtain $U_{rj}^{k-1} = 0$.

□

Lemma 3. *Let $w = (w_1, \ldots, w_k)$, if $\sum_{j=1}^{k} w_j = 1$ then we have $L_k \cdot w = (1, \ldots, 1)^T - (I_k + U_k) \cdot w$.*

Proof. First note that since $\sum_{j=1}^{k} w_j = 1$,

$$L_k \cdot w = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 1 & \cdots & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} = \begin{bmatrix} 0 \\ w_1 \\ w_1 + w_2 \\ \vdots \\ \sum_{j=1}^{k-1} w_j \end{bmatrix} = \begin{bmatrix} 1 - \sum_{j=1}^{k} w_j \\ 1 - \sum_{j=2}^{k} w_j \\ \vdots \\ 1 - w_k \end{bmatrix}$$

On the other hand, if we let $\mathbf{1} = (1, \ldots, 1)^T$, we have also:

$$\mathbf{1} - (I_k + U_k) \cdot w = \mathbf{1} - \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ \vdots \\ w_k \end{bmatrix} = \begin{bmatrix} 1 - \sum_{j=1}^{k} w_j \\ 1 - \sum_{j=2}^{k} w_j \\ \vdots \\ 1 - w_k \end{bmatrix}$$

□

Lemma 4. *For $z = e_i - e_j$, we have that $(L_k + U_k) \cdot z = -z$.*

Proof. If $i = j$, then $z = 0$ and the results is true. Suppose w.l.o.g. that $i > j$ (otherwise we just prove the result for $-z$). Then

$$U_k \cdot (e_i - e_j) = \sum_{s=1}^{i-1} e_s - \sum_{s=1}^{j-1} e_s = \sum_{s=j}^{i-1} e_s.$$

Similarly

$$L_k \cdot (e_i - e_j) = \sum_{s=i+1}^{k} e_s - \sum_{s=j+1}^{k} e_s = \sum_{s=j+1}^{i} -e_s.$$

Therefore

$$(L_k + U_k) \cdot (e_i - e_j) = \sum_{s=j}^{i-1} e_s - \sum_{s=j+1}^{i} e_s = e_j - e_i = -z.$$

□

3.3 How to recover the bids when knowing the l_{ij}'s

As discussed above, we can represent the function f as a matrix multiplication. Let M be the following square matrix of size $nk \times nk$:

$$M = \begin{bmatrix} (U+L) & U & \dots & \dots & U \\ (U+I) & (U+L) & U & \dots & U \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ (U+I) & \dots & (U+I) & (U+L) & U \\ (U+I) & \dots & \dots & (U+I) & (U+L) \end{bmatrix}. \text{ Then } f(b) = M \cdot b.$$

The function takes as input a vector composed of n vectors, each of k bits. It returns the nk values l_{ij}, $1 \leq i \leq n$ and $1 \leq j \leq k$. As explained above, the structure of the matrix is defined by the formula that computes γ_{ij}^a, which consists essentially of three factors: first we multiply all α_{ij} which encode bigger bids (represented by the matrix U), then we multiply all α_{ij} which encode smaller bids by the same bidder (represented by adding the matrix L on the diagonal), and finally we multiply by all α_{ij} which encode the same bid by bidders with a smaller index (represented by adding the matrix I on the lower triangle of M). In our encoding there will be a "1" in the vector for each Y in the protocol, hence f will count how many Ys are multiplied when computing γ_{ij}^a.

Example 5 (f in the case of two bidders and two prices)**.** *In the case of two bidders and two possible prices, we obtain the following matrix M:*

$$M = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

In this example, we have four possible cases for the bids:

$$b_0 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \; b_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \; b_2 = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \; b_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

We can compute f for all cases:

$$f(b_0) = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \; f(b_1) = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 0 \end{pmatrix}, \; f(b_2) = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 2 \end{pmatrix}, \; f(b_3) = \begin{pmatrix} 2 \\ 0 \\ 2 \\ 1 \end{pmatrix}$$

In this example it is easy to see that for all $i \neq j$ we have $f(b_i) \neq f(b_j)$, even if the the winner and the winning price is the same, as for example in the case of b_2 and b_3.

In general, we can prove the following theorem stating that f is injective.

Theorem 6. *f is injective on valid bid vectors, i.e. for two different correct bid vectors $u = [u_1, \ldots, u_k]^T$ and $v = [v_1, \ldots, v_k]^T$ with $u \neq v$ we have $M \cdot u \neq M \cdot v$.*

Proof. Let u and v be two correct bid vectors such that $u \neq v$. We want to prove that $M \cdot u \neq M \cdot v$. We make a proof by contradiction, hence we assume that $M \cdot u = M \cdot v$ or that $M \cdot (u - v) = 0$. Because u and v are two correct bid vectors, each one of them is an element of the canonical basis $(e_1, \ldots, e_k)$, i.e. $u = e_i$ and $v = e_j$, as shown in Section 3.1. We denote $u - v$ by z, and consequently $z = e_i - e_j$. Knowing that $M \cdot z = 0$, we prove by induction on a that for all a the following property $P(a)$ holds:

$$P(a) : \forall l, 0 < l \leq a, diag(U^{k-l}) \cdot z = 0$$

where $diag(U^{k-x})$ is a $nk \times nk$ block diagonal matrix containing only diagonal blocks of the same matrix U^{k-x}. The validity of $P(k)$ proves in particular that $diag(U^0) \cdot z_l = 0$, i.e. $z = 0$ which contradicts our hypothesis.

- Case $a = 1$: we also prove this base case by induction, i.e. for all $b \geq 1$ the property $Q(b)$ holds, where:

$$Q(b) : \forall m, 0 < m \leq b, U^{k-1} \cdot z_m = 0$$

which gives us that $U^{k-1} \cdot z = 0$.

  - Base case $b = 1$: We start by looking at the multiplication of the first row of M with z. We obtain:

$$(L + U) \cdot z_1 + U \cdot (z_2 + \ldots + z_k) = 0.$$

We can multiply each side by U^{k-1}, and use Lemma 4 to obtain:

$$U^{k-1} \cdot [-z_1 + U \cdot (z_2 + \ldots + z_k)] = 0.$$

Since U is nilpotent, according to Lemma 2 the latter gives $-U^{k-1} \cdot z_1 = 0$. Hence we know $Q(1) : U^{k-1} \cdot z_1 = 0$, i.e. the last entry of z_1 is 0.

  - Inductive step $b + 1$: assume $Q(b)$. Consider now the multiplication of the $(b + 1)$-th row of the matrix M:

$$(U+I)\cdot z_1+\ldots+(U+I)\cdot z_b+(L+U)\cdot z_{b+1}+U\cdot(z_{b+2}+\ldots+z_k) = 0.$$

Then by multiplying by U^{k-1} and using Lemma 4 we obtain:

$$U^{k-1}\cdot[(U+I)\cdot z_1+\ldots+(U+I)\cdot z_b-z_{b+1}+U\cdot(z_{b+2}+\ldots+z_k)] = 0.$$

Since U is nilpotent according to Lemma 2 we have

$$U^{k-1} \cdot z_1 + \ldots + U^{k-1} \cdot z_b - U^{k-1} \cdot z_{b+1} = 0.$$

Using the fact that for all $m < b$ we have $U^{k-1} \cdot z_m = 0$, the latter gives $-U^{k-1} \cdot z_{b+1} = 0$.

- Inductive step $a + 1$: assume $P(a)$. By induction on $b \geq 1$ we will show that $Q'(b)$ holds, where

$$Q'(b) : \forall m, 0 < m \leq b, U^{k-(a+1)} \cdot z_m = 0$$

which gives us that $U^{k-(a+1)} \cdot z = 0$, i.e. $P(a + 1)$.

– Base case $b = 1$: Consider the multiplication of the first row with $U^{k-(a+1)}$:

$$U^{k-(a+1)} \cdot [(L+U) \cdot z_1 + U \cdot (z_2 + \ldots + z_k)] = 0$$

which can be rewritten as

$$-U^{k-(a+1)} \cdot z_1 + U^{k-a} \cdot (z_2 + \ldots + z_k) = 0.$$

Using $U^{k-a} \cdot z_l = 0$ for all l, we can conclude that $-U^{k-(a+1)} \cdot z_1 = 0$, i.e. $Q'(1)$ holds.

– Inductive step $b+1$: assume $Q'(b)$. Consider now the $(b+1)$-th row of the matrix M:

$$(U+I) \cdot z_1 + \ldots + (U+I) \cdot z_b + (L+U) \cdot z_{b+1} + U \cdot (z_{b+2} + \ldots + z_k) = 0.$$

Then by multiplying by $U^{k-(a+1)}$ and using Lemma 4 we obtain:

$$U^{k-(a+1)} \cdot [(U+I) \cdot z_1 + \ldots + (U+I) \cdot z_b - z_{b+1} + U \cdot (z_{b+2} + \ldots + z_k)] = 0.$$

Using $U^{k-a} \cdot z_l = 0$ for all l, we can conclude that

$$U^{k-(a+1)} \cdot z_1 + \ldots + U^{k-(a+1)} \cdot z_b - U^{k-(a+1)} \cdot z_{b+1} = 0.$$

Now, for all $m < b$, we have $U^{k-(a+1)} \cdot z_m = 0$, so that $-U^{k-(a+1)} \cdot z_{b+1} = 0$; i.e. $Q'(b+1)$ holds.

□

This theorem shows that if there is a constellation of bids that led to certain values l_{ij}, this constellation is unique. Hence we are able to invert f on valid outputs. We will now show that this can be efficiently done.

3.3.1 Recursive formulae for the bids

Our aim is to solve the following linear system: $M \cdot x = l$. We will use the same steps we used for the proof of injectivity to solve this system efficiently. Consider the r-th block of size k of the latter system. We have $x_r = (x_{r,1}, x_{r,2}, \ldots, x_{r,k})$ and the r-th block of $M \cdot x$ is

$$(U+I)x_1 + \ldots + (U+I)x_{r-1} + (L+U)x_r + Ux_{r+1} + \ldots + Ux_n = U(\sum_{i=1}^{n} x_i) + (\sum_{i=1}^{r-1} x_i) + Lx_r.$$

As the r-th block of l is l_r, we thus have:

$$U\left(\sum_{i=1}^{n} x_i\right) + \sum_{i=1}^{r-1} x_i + Lx_r = l_r$$

Using Lemma 3, with $w_j = x_{r,j}$ for $j = 1..k$, we can exchange L in the latter to get

$$U\left(\sum_{i=1}^{n} x_i\right) + \sum_{i=1}^{r-1} x_i + (\mathbf{1} - (I+U)\, x_r) = l_r.$$

Hence,

$$U\left(\sum_{i=1}^{n} x_i\right) + \sum_{i=1}^{r-1} x_i + \mathbf{1} - x_r - Ux_r = l_r,$$

so that we now have:

$$\begin{cases} x_1 = \mathbf{1} - l_1 + U\left(\sum_{i=2}^{n} x_i\right) \\ x_r = \mathbf{1} - l_r + \sum_{i=1}^{r-1} x_i + U\left(\sum_{i=1, i\neq r}^{n} x_i\right) \quad \text{if } 1 < r \leq n \end{cases} \tag{3}$$

This gives us a formula to compute the values of $x_{i,j}$, starting with the last element of the first block $x_{1,k}$. Then we can compute the last elements of all other blocks $x_{2,k}, \ldots, x_{n,k}$, and then the second to last elements $x_{1,k-1}, \ldots, x_{n,k-1}$, etc.

The idea is to project the above Equation (3) on the t-th coordinate. Then, the t-th row of U has ones only starting at index $t+1$, and thus the t-th row of Uz involves only the elements $z_{t+1}, \ldots, z_k$. We thus have:

$$e_t^T U\left(\sum_{i=1, i\neq r}^{n} x_i\right) = \sum_{j=t+1}^{k} \sum_{i=1, i\neq r}^{n} x_{i,j}$$

for $t < k$ and $e_k^T U = 0$. Now $e_t^T x_r = x_{r,t}$, $e_t^T l_r = l_{r,t}$ and $e_t^T \mathbf{1} = 1$. Hence, we therefore get the following where at row t, the right hand side involves only

already computed values:

$$\begin{cases} x_{1,k} = 1 - l_{1,k} & \\ x_{r,k} = 1 - l_{r,k} + \sum_{i=1}^{r-1} x_{i,k} & \text{if } 1 < r \leq n \\ x_{1,t} = 1 - l_{1,t} + \sum_{j=t+1}^{k} \sum_{i=2}^{n} x_{i,j} & \text{if } 1 \leq t < k \\ x_{r,t} = 1 - l_{r,t} + \sum_{i=1}^{r-1} x_{i,t} + \sum_{j=t+1}^{k} \sum_{i=1, i\neq r}^{n} x_{i,j} & \text{if } 1 \leq t < k \text{ and } 1 < r \leq n \end{cases} \tag{4}$$

3.3.2 Efficient Algorithm.

To obtain all values, we have to apply the above Formula (4) for each $1 \leq r \leq n$ and $1 \leq t \leq k$, but we know that the bids also satisfy the following constraint: there is a single non zero value, a one, per block of x of size k. Therefore, the last Equation (4) can be replaced by the following choice:

- either the location of the bid for the r-th bidder, i.e. the non zero value in x_r, has already been found at step t. Then $\exists t + 1 \leq j_0 \leq k, x_{r,j_0} = 1$ and the remaining part of x_r is all zeroes, so that $x_{r,t} = 0$.
- or $\forall t + 1 \leq j \leq k, x_{r,j} = 0$. Then the $i \neq r$ can be removed in the last sum of the formula and

$$x_{r,t} = 1 - l_{r,t} + \sum_{i=1}^{r-1} x_{i,t} + \sum_{j=t+1}^{k} \sum_{i=1}^{n} x_{i,j}.$$

From this remark, one sees that either we already know the $x_{r,t}$ value or it is just $1-l_{r,t}$ plus the sum of all the previously computed $x_{i,j}$. Therefore to efficiently compute all these values, it is sufficient to maintain a counter of the known values as in Algorithm 1.

Theorem 7. *Algorithm 1 is correct and its arithmetic cost can be bounded by* $O(nk)$.

Proof. The correctness follows from the analysis in the beginning of this section. Then for each entry in the output vector, we have to compute two additions, one test and potentially a counter increment. □

This is two orders of magnitude less than the number of operations required just to compute all the encrypted bids.

Algorithm 1 Bids recovery

Require: The l_{rt} exponents of Y in the encryption of Brandt's protocol, for $1 \leq r \leq n$ and $1 \leq t \leq k$.

Ensure: The bids x_{rt} satisfying Equations (4).

Let $counter = 0$;
for $r = 1..n$ **do**
  $bidfound[r] = false$;
end for
$x_{1,k} = 1 - l_{1,k}$
if $x_{1,k} == 1$ **then**
  $++counter$;
  $bidfound[1] = true$;
end if
for $r = 1..n$ **do**
  $x_{r,k} = 1 - l_{r,k} + counter$;
  if $x_{r,k} == 1$ **then**
    $++counter$;
    $bidfound[r] = true$;
  end if
end for
for $t = k - 1$ down **to** 1 **do**
  if not $bidfound[1]$ **then**
    $x_{1,t} = 1 - l_{1,t} + counter$;
    if $x_{1,t} == 1$ **then**
      $++counter$;
      $bidfound[1] = true$;
    end if
    for $r = 2..n$ **do**
      if not $bidfound[r]$ **then**
        $x_{r,t} = 1 - l_{r,t} + counter$;
        if $x_{r,t} == 1$ **then**
          $++counter$;
          $bidfound[r] = true$;
        end if
      end if
    end for
  end if
end for

3.4 Attack on the random noise: how to obtain the l_{ij}'s

In the previous section we showed that knowing the l_{ij}'s allows us the efficiently break the privacy of all bidders. Here is how to obtain the l_{ij}'s. The seller will learn all $v_{ij} = \left(Y^{l_{ij}}\right)^{(\sum_{h=1}^{n} m_{ij}^h)}$ at the end of the protocol. Since the m_{ij}^h are randomly chosen, this will be a random value if $l_{ij} \neq 0$. However a malicious bidder ("Mallory", of index a) can cancel out the m_{ij}^h as follows: in Step 3 of the protocol each bidder will compute his γ_{ij}^a and δ_{ij}^a. Mallory waits until all other bidders have published their values (the protocol does not impose any synchronization or special ordering) and then computes his values γ_{ij}^ω and δ_{ij}^ω as:

$$\gamma_{ij}^\omega = \left(\left(\prod_{h=1}^{n} \prod_{d=j+1}^{k} \alpha_{hd} \right) \cdot \left(\prod_{d=1}^{j-1} \alpha_{id} \right) \cdot \left(\prod_{h=1}^{i-1} \alpha_{hj} \right) \right) \cdot \left(\prod_{k \neq \omega} \gamma_{ij}^k \right)^{-1}$$

$$\delta_{ij}^\omega = \left(\left(\prod_{h=1}^{n} \prod_{d=j+1}^{k} \beta_{hd} \right) \cdot \left(\prod_{d=1}^{j-1} \beta_{id} \right) \cdot \left(\prod_{h=1}^{i-1} \beta_{hj} \right) \right) \cdot \left(\prod_{k \neq \omega} \delta_{ij}^k \right)^{-1}$$

The first part is a correct encryption of $Y^{l_{ij}}$, with $m_{ij}^\omega = 1$ for all i and j. The second part is the inverse of the product of all the other bidders γ_{ij}^k and δ_{ij}^k, and thus it will eliminate the random exponents. Hence after decryption the seller obtains $v_{ij} = Y^{l_{ij}}$, where $l_{ij} < n$ for a small n. He can compute l_{ij} by simply (pre-)computing all possible values Y^r and testing for equality. This allows the seller to obtain the necessary values and then to use the resolution algorithm to obtain each bidder's bid. Note that although we changed the intermediate values, the output still gives the correct result (i.e. winning bid). Therefore, the attack might even be unnoticed by the other participants. Note also that choosing a different Y_i per bidder does not prevent the attack, since all the Y_i need to be public in order to prove the correctness of the bid in Step 2 of the protocol.

However the protocol requires Mallory to prove that γ_{ij}^ω and δ_{ij}^ω have the same exponent. This is obviously the case, but Mallory does not know the exact value of this exponent. Thus it is impossible for him to execute the proposed zero-knowledge protocol directly.

In the original paper [Bra06] the malleable interactive proof of [CP92], presented in Section 2.3, is used to prove the correctness of γ_{ij}^a and δ_{ij}^a in Step 3 of the protocol. If this proof is not converted into a non-interactive proof, then Mallory is able to fake it as follows.

3.5 Proof of equality of the presented outcomes

Note that we can rewrite γ_{ij}^{ω} and δ_{ij}^{ω} as:

$$v = \gamma_{ij}^{\omega} = \underbrace{\left(\left(\prod_{h=1}^{n}\prod_{d=j+1}^{k}\alpha_{hd}\right)\cdot\left(\prod_{d=1}^{j-1}\alpha_{id}\right)\cdot\left(\prod_{h=1}^{i-1}\alpha_{hj}\right)\right)}_{g_1}^{1-\left(\sum_{k\neq\omega} m_{ij}^{k}\right)}$$

$$w = \delta_{ij}^{\omega} = \underbrace{\left(\left(\prod_{h=1}^{n}\prod_{d=j+1}^{k}\beta_{hd}\right)\cdot\left(\prod_{d=1}^{j-1}\beta_{id}\right)\cdot\left(\prod_{h=1}^{i-1}\beta_{hj}\right)\right)}_{g_2}^{1-\left(\sum_{k\neq\omega} m_{ij}^{k}\right)}$$

When Mallory, the bidder m, is asked by Victor for a proof of correctness of his values, he starts by asking all other bidders for proofs to initialize the man-in-the-middle attack of Figure 1. Each of them answers with values $\lambda_o = g_1^{z_o}$ and $\mu_o = g_2^{z_o}$. Mallory can then answer Victor with values $\lambda = \prod_o \lambda_o^{-1}$ and $\mu = \prod_o \mu_o^{-1}$, where $o \in ([1,n] \setminus m)$. Victor then sends a challenge c, which Mallory simply forwards to the other bidders. They answer with $r_o = z_o + c \cdot m_{ij}^{o}$, and Mallory sends $r = c - \sum_o r_o$ to Victor, who can check that $g_1^r = \lambda \cdot v^c$ and $g_2^r = \mu \cdot w^c$. If the other bidders did their proofs correctly, then Mallory's proof will appear valid to Victor:

$$\lambda \cdot v^c = \prod_o \lambda_o^{-1} \cdot \left(g_1^{1-\left(\sum_o m_{ij}^{o}\right)}\right)^c = \prod_o g_1^{-z_o} \cdot g_1^{c-c\left(\sum_o m_{ij}^{o}\right)} = g_1^{c-\sum_o\left(z_o + c m_{ij}^{o}\right)}$$

$$\mu \cdot w^c = \prod_o \mu_o^{-1} \cdot \left(g_2^{1-\left(\sum_o m_{ij}^{o}\right)}\right)^c = \prod_o g_2^{-z_o} \cdot g_2^{c-c\left(\sum_o m_{ij}^{o}\right)} = g_2^{c-\sum_o\left(z_o + c m_{ij}^{o}\right)}$$

Hence in the case of malleable interactive zero-knowledge proofs Mallory is able to modify the values γ_{ij}^{ω} and δ_{ij}^{ω} as necessary, and even prove the correctness using the bidders. Hence the modifications may stay undetected and the seller will be able to break privacy.

3.6 Efficiency of the attack

In order to measure the practicability of the attack we have implemented the protocol in C++ using the Givaro library[2] for the large modular computations. We have used a 32 cores shared memory Intel Xeon E5-4620 to simulate several bidders,

[2] http://givaro.forge.imag.fr/

each core running at 2.2 GHz. The computations of the seller (namely the computation of the winner) and of Mallory, our attacker, have been performed on a single core of this machine.

First we report on Table 1 our timings for a different number of bidders and 8192 different possible prices. The "Set-up" column represents the total time required to set-up Brandt's protocol on our machine with 32 cores whereas the "Set-up / bidder" column represent the time required for each bidder. On the one hand, this shows that Brandt's protocol requires quite a lot of computing time and that as predicted by the arithmetic complexity, the set-up time can be quite resource demanding for each bidder, as well as for the seller ("Winner computation" is done by the seller). On the other hand, we see that both our attacks are even some orders of magnitude faster than just setting up the protocol and that our novel attack requires only a few seconds to break the privacy of realistic auctions (with hundreds of bidders and thousands of prices).

Table 1: Parallel Brandt for 8192 bids with OpenMP on an Intel Xeon E5-4620, 32x2.2GHz

Bidders	Bids	Set-up	Set-up / bidder	Winner computation	[DDL13] Attack	Novel Attack
2	8192	80.354 s	80.354 s	0.454 s	0.100 s	0.001 s
4	8192	124.370 s	124.370 s	0.758 s	0.181 s	0.002 s
8	8192	219.641 s	219.641 s	1.768 s	0.594 s	0.004 s
16	8192	396.398 s	396.398 s	3.620 s	1.560 s	0.011 s
32	8192	1399.480 s	1399.480 s	16.945 s	4.601 s	0.043 s
64	8192	4181.830 s	2090.915 s	72.924 s	14.171 s	0.192 s
128	8192	14559.700 s	3639.925 s	420.497 s	54.200 s	0.664 s
256	8192	50546.800 s	6318.350 s	2942.650 s	173.255 s	2.910 s

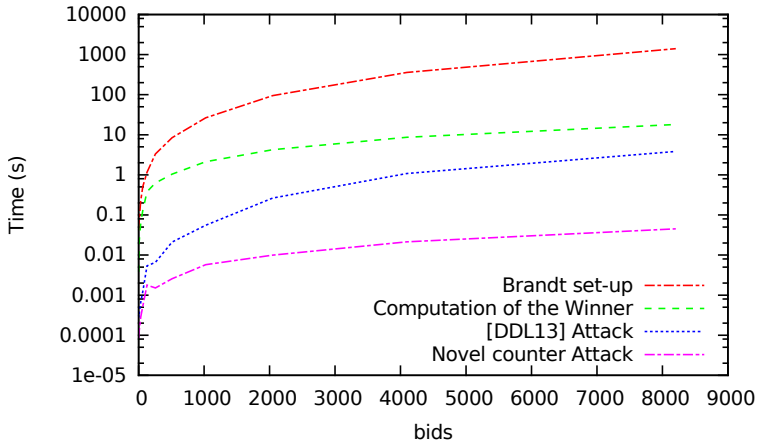
Next we also show in the log scale Figure 3 that the arithmetic complexity bounds given in Section 3.3.2 are indeed tight: quadratic in the number of bids and bidders for the setting-up and the previous attack; only linear for the novel attack.

3.7 The complete attack and countermeasures

Putting everything together, the attack works as follows:

1. The bidders set up the keys as described in the protocol.
2. They encrypt and publish their bids.
3. They compute γ_{ij}^h and δ_{ij}^h and publish them.

Figure 3: Parallel Brandt for 32 bidders with OpenMP on an Intel Xeon E5-4620, 32x2.2GHz



4. Mallory, who is a bidder himself, waits until all other bidders have published their values. He then computes his values as defined above, and publishes them.

5. If he is asked for a proof, he can proceed as explained above in Section 3.5.

6. The bidders (including Mallory) jointly decrypt the values.

7. The seller obtains all $Y^{l_{ij}}$'s. He can then compute the l_{ij}'s by testing at most n possibilities.

8. Once he has all values, he can invert the function f as explained above.

9. He obtains all bidders bids.

Again, note that for all honest bidders, this execution will look normal, so they might not even notice that an attack took place. To prevent this attack, one could perform the following actions:

- To counteract the removal of the noise of Section 3.4, the bidders could check whether the product of the $\gamma_{i,j}^a$ for all bidders a is equal to the product of the α_{hd} without any noise (exponent is 1). Unfortunately, the man-in-the-middle attack generalizes to any exponent as shown in Figure 2. Therefore the attacker could use a randomly chosen exponent only known to him.

- As mentioned above, another countermeasure is the use of non-interactive, non-malleable proofs of knowledge. In this case, we will show in Section 5 that it is still possible to attack a targeted bidder's privacy.

4 Attacking verifiability

Brandt claims that the protocol is verifiable as the parties have to provide zero-knowledge proofs for their computations, however there are two problems.

4.1 Exceptional values

First, a winning bidder cannot verify if he actually won. To achieve privacy, the protocol hides all outputs of v_{aj} except for the entry containing "1"[3]. This is done by exponentiation with random values m_{ij}^a inside all entries γ_{ij}^a and δ_{ij}^a, i.e. by computing $x_{ij}^{\sum_a m_{ij}^a}$ where x_{ij} is the product of some α_{ij} as specified in the protocol. If $x_{ij} = 1$, for any k we have $x_{ij}^k = 1$. For any other value of x_{ij}, x_{ij}^k should be different from 1. However the random values m_{ij}^a may add up to zero (mod q), hence the returned value will be $x_{ij}^{\sum_a m_{ij}^a} = x_{ij}^0 = 1$ and the bidder will conclude that he won, although he actually lost ($x_{ij} \neq 1$). Hence simply verifying the proofs is not sufficient to be convinced that the observed outcome is correct. For the same reason the seller might observe two or more "1"-values, even though all proofs are correct. In such a situation he is unable to decide which bidder actually won since he cannot determine which "1"s correspond to a real bids, and hence which bid is the highest real bid. If two "1"s correspond to real bids, he could even exploit such a situation to his advantage: he can tell both bidders that they won and take money from both, although there is only one good to sell – this is normally prohibited by the protocol's tie-breaking mechanism. If the bidders do not exchange additional data there is no way for them to discover that something went wrong, since the seller is the only party having access to all values.

A solution to this problem could work as follows: when computing the γ_{ij}^a and δ_{ij}^a, the bidders can check if the product

$$x_{ij} = \left(\prod_{h=1}^{n} \prod_{d=j+1}^{k} \alpha_{hd} \right) \cdot \left(\prod_{d=1}^{j-1} \alpha_{id} \right) \cdot \left(\prod_{h=1}^{i-1} \alpha_{hj} \right)$$

is equal to one – if yes, they restart the protocol using different keys and random values. If not, they continue, and check if $\prod_a \gamma_{ij}^a = 1$. If yes, they choose different

[3] Note that the protocol contains a mechanism to resolve ties, i.e. there should always be exactly one entry equal to 1, even in the presence of ties.

random values m_{ij}^a and re-compute the γ_{ij}^a and δ_{ij}^a, otherwise they continue. Since the probability of the random values adding up to zero is low, this will rapidly lead to correct values.

4.2 Different private keys

Second, the paper does not precisely specify the proofs that have to be provided in the joint decryption phase. If the bidders only prove that they use the same private key on all decryptions *and not also that it is the one they used to generate their public key*, they may use a wrong one. This will lead to a wrong decryption where with very high probability no value is "1", as they will be random. Hence all bidders will think that they lost, thus allowing a malicious bidder to block the whole auction, as no winner is determined. Hence, if we assume that the verification test consists in verifying the proofs, a bidder trying to verify that he lost using the proofs might perform the verification successfully, although the result is incorrect and he actually won – since he would have observed a "1" if the vector had been correctly decrypted.

This problem can be addressed by requiring the bidders to also prove that they used the same private key as in the key generation phase.

5 Attacks using the lack of authentication

The protocol as described in the original paper does not include any authentication of the messages. This means that an attacker in control of the network can impersonate any party, which can be exploited in many ways. However, the authors supposed in the original paper a "reliable broadcast channel, i.e. the adversary has no control of communication" [Bra06]. Yet even under this assumption dishonest participants can impersonate other participants by submitting messages on their behalf. Additionally, this assumption is difficult to achieve in asynchronous systems [FLP85]. In the following we consider an attacker in control of the network, however many attacks can also be executed analogously by dishonest parties (which are considered in the original paper) in the reliable broadcast setting.

5.1 Another attack on privacy

Our first attack on privacy only works in the case of malleable interactive proofs. If we switch to non-interactive non-malleable proofs, Mallory cannot ask the other bidders for proofs using a challenge of his choice.

However, even with non-interactive non-malleable zero-knowledge proofs, the protocol is still vulnerable to attacks on a targeted bidder's privacy if an attacker

can impersonate any bidder of his choice as well as the seller, which is the case for an attacker controlling the network due to the lack of authentication. In particular, if he wants to know Alice's bid he can proceed as follows:

1. Mallory impersonates all other bidders. He starts by creating keys on their behalf and publishes the values y_i and the corresponding proofs for all of them.
2. Alice also creates her secret keyshare and publishes y_a together with a proof.
3. Alice and Mallory compute the public key y.
4. Alice encrypts her bid and publishes her α_{aj} and β_{bj} together with the proofs.
5. Mallory publishes $\alpha_{ij} = \alpha_{aj}$ and $\beta_{ij} = \beta_{aj}$ for all other bidders i and also copies Alice's proofs.
6. Alice and Mallory execute the computations described in the protocol and publish γ_{ij}^a and δ_{ij}^a.
7. They compute ϕ_{ij}^a and send it to the seller.
8. The seller publishes the ϕ_{ij}^a and computes the v_{aj}.

Since all submitted bids are equal, the seller (which might also be impersonated by Mallory) will obtain Alice's bid as the winning price, hence it is not private any more. This attack essentially simulates a whole instance of the protocol to make Alice indirectly reveal a bid that was intended for another, probably real auction. To counteract this it is not sufficient for Alice to check that the other bids are different: Mallory can produce different $\alpha_{ij} = \alpha_{aj} y^x$ together with $\beta_{ij} = \beta_{aj} g^x$ which are still correct encryptions of Alice bids.

Note that the same attack also works if dishonest bidders collude with the seller: they simply re-submit the targeted bidders bid as their own bid.

5.2 Attacking fairness, non-repudiation and verifiability

The lack of authentication obviously entails that a winning bidder can claim that he did not submit his bid, hence violating non-repudiation (even in the case of reliable broadcast). Additionally, this also enables an attack on fairness: an attacker in control of the network can impersonate all bidders vis-à-vis the seller, submitting bids of his choice on their behalf and hence completely controlling the winner and winning price. This also causes another problem with verifiability: it is impossible to verify if the bids were submitted by the registered bidders, or by somebody else.

5.3 Countermeasures

The solution to these problems is simple: all the messages need to be authenticated, e.g. using signatures or Message Authentication Codes (MACs). This requires a trust anchor, for example a Public Key Infrastructure (PKI) that verifies the identities of the participants and certifies their keys.

6 Conclusion

In this paper we analyzed the protocol of Brandt [Bra06] from various angles. We showed that the underlying computations have a weakness which can be exploited by malicious bidders to break privacy if malleable interactive zero-knowledge proofs are used. Using an implementation of the protocol and the attack we illustrated its practical efficiency. We also identified two problems with verifiability and proposed solutions. Finally we showed how the lack of authentication can be used to mount different attacks on privacy, verifiability as well as fairness and non-repudiation. Again we suggested a solution to address the discovered flaws.

So sum up, the following countermeasures have to be implemented:

- Use of non-interactive or non-malleable zero-knowledge proofs.
- All messages have to be authenticated, e.g. using a Public-Key Infrastructure (PKI) and signatures.
- In the outcome computation step: when computing the γ_{ij}^a and δ_{ij}^a, the bidders can check if

$$x_{ij} = \left(\prod_{h=1}^{n} \prod_{d=j+1}^{k} \alpha_{hd} \right) \cdot \left(\prod_{d=1}^{j-1} \alpha_{id} \right) \cdot \left(\prod_{h=1}^{i-1} \alpha_{hj} \right)$$

  is equal to one – if yes, they restart the protocol using different keys and random values. If not, they continue, and check if $\prod_a \gamma_{ij}^a = 1$. If yes, they choose different random values m_{ij}^a and re-compute the γ_{ij}^a and δ_{ij}^a, otherwise they continue.
- In the outcome decryption step: the bidders have to prove that the value x_a they used to decrypt is the same x_a they used to generate their public key y_a in the first step.

The attacks show that properties such as authentication can be necessary to achieve other properties like for instance privacy, which might appear to be unrelated at first sight. It also points out that there is a difference between computing the

winner in a fully private way, and ensuring privacy for the bidders: in the second attack we use modified inputs to break privacy even though the computations themselves are secure. Additionally our analysis highlights that the choice of interactive or non-interactive, malleable or non-malleable proofs is an important decision in any protocol design.

As for possible generalizations of our attacks, of course the linear algebra part of our first attack is specific to this protocol. Yet the man-in-the-middle attack on malleable proofs as well as the need of authentication for privacy are applicable to any protocol. Similarly, checking all exceptional cases and ensuring that the same keys are used all along the process are also valid insights for other protocols.

As future work we would like to realize a full formal security proof of the fixed protocol.

Acknowledgments. This work was partly supported by the ANR projects ProSe (decision ANR-2010-VERS-004-01), HPAC (ANR-11-BS02-013) and the support of the "Digital Trust" Chair from the University of Auvergne Foundation. We also thank Dorian Arnaud, Jean-Baptiste Gheeraert, Maud Lefevre, Simon Moura and Jérémy Pouzet for spotting an error in the description of our first attack in [DDL13].

References

[AGG+05] Andrea Attanasio, Gianpaolo Ghiani, Lucio Grandinetti, Emanuela Guerriero, and Francesca Guerriero. Operations research methods for resource management and scheduling in a computational grid: a survey. In Lucio Grandinetti, editor, *Grid Computing The New Frontier of High Performance Computing*, volume 14 of *Advances in Parallel Computing*, pages 53 – 81. North-Holland, 2005.

[BAGS02] Rajkumar Buyya, David Abramson, Jonathan Giddy, and Heinz Stockinger. Economic models for resource management and scheduling in grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1507–1542, 2002.

[BCM05] Endre Bangerter, Jan Camenisch, and Ueli M. Maurer. Efficient proofs of knowledge of discrete logarithms and representations in groups with hidden order. In *Proceedings of the 8th international conference on Theory and Practice in Public Key Cryptography*, PKC'05, pages 154–171, Berlin, Heidelberg, 2005. Springer-Verlag.

[BDPW89] Mike Burmester, Yvo Desmedt, Fred Piper, and Michael Walker. A general zero-knowledge scheme. In *Advances in Cryptology - EUROCRYPT '89, Houthalen, Belgium*, volume 434 of *LNCS*, pages 122–133. Springer, April 1989.

[Bra02] Felix Brandt. A verifiable, bidder-resolved auction protocol. In R. Falcone, S. Barber, L. Korba, and M. Singh, editors, *Proceedings of the 5th AAMAS Workshop on Deception, Fraud and Trust in Agent Societies*, pages 18–25, 2002.

[Bra03] Felix Brandt. Fully private auctions in a constant number of rounds. In *Financial Cryptography 2003*, volume 2742 of *LNCS*, pages 223–238. Springer, 2003.

[Bra06] Felix Brandt. How to obtain full privacy in auctions. *International Journal of Information Security*, 5:201–216, 2006.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98*, volume 1462 of *LNCS*, pages 424–441. Springer Berlin Heidelberg, 1998.

[CEvdG87] David Chaum, Jan-Hendrik Evertse, and Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology - EUROCRYPT '87, Amsterdam, The Netherlands, April 13-15, 1987*, volume 304 of *LNCS*, pages 127–141, 1987.

[CEvdGP86] D. Chaum, J.H. Evertse, J. van de Graaf, and R. Peralta. Demonstrating possession of a discrete logarithm without revealing it. In *CRYPTO'86*, volume 263 of *LNCS*, pages 200–212. Springer, 1986.

[CMW10] Sherman S. M. Chow, Changshe Ma, and Jian Weng. Zero-knowledge argument for simultaneous discrete logarithms. In My T. Thai and Sartaj Sahni, editors, *COCOON*, volume 6196 of *LNCS*, pages 520–529. Springer, 2010.

[CP92] David Chaum and Torben P. Pedersen. Wallet databases with observers. In *Crypto'92, California, USA*, volume 0740 of *LNCS*, pages 89–105. Springer, 1992.

[CPS07] Brian Curtis, Josef Pieprzyk, and Jan Seruga. An efficient eAuction protocol. In *ARES*, pages 417–421. IEEE Computer Society, 2007.

[DDL13] Jannik Dreier, Jean-Guillaume Dumas, and Pascal Lafourcade. Brandt's fully private auction protocol revisited. *Lecture Notes in Computer Science (AfricaCrypt'2013, Proceedings of the 6th International Conference on Cryptology in Africa, Cairo, Egypt)*, 7918:88–106, June 2013.

[EG85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[FF09] Marc Fischlin and Roger Fischlin. Efficient non-malleable commitment schemes. *Journal of Cryptology*, 22:530–571, 2009.

[FLP85] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[Kat02] Jonathan Katz. *Efficient cryptographic protocols preventing "man-in-the-middle" attacks*. PhD thesis, Columbia University, 2002.

[Kri02] Vijay Krishna. *Auction Theory*. Academic Press, San Diego, USA, 2002.

[Mau09] Ueli Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *Progress in Cryptology âĂŞ AFRICACRYPT 2009*, volume 5580 of *LNCS*, pages 272–286. Springer Berlin Heidelberg, 2009.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *ACM Conference on Electronic Commerce*, pages 129–139, 1999.

[OM01] Kazumasa Omote and Atsuko Miyaji. A practical English auction with one-time registration. In Vijay Varadharajan and Yi Mu, editors, *ACISP*, volume 2119 of *LNCS*, pages 221–234, 2001.

[PBDV02] Kun Peng, Colin Boyd, Ed Dawson, and Kapali Viswanathan. Robust, privacy protecting and publicly verifiable sealed-bid auction. In Robert H. Deng, Sihan Qing, Feng Bao, and Jianying Zhou, editors, *ICICS*, volume 2513 of *LNCS*, pages 147–159. Springer, 2002.

[Sak00] Kazue Sako. An auction protocol which hides bids of losers. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1751 of *LNCS*, pages 422–432. Springer, 2000.

[Sch91] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[SSS02] Ahmad-Reza Sadeghi, Matthias Schunter, and Sandra Steinbrecher. Private auctions with multiple rounds and multiple items. In *DEXA Workshops*, pages 423–427. IEEE, 2002.