Brandt's Fully Private Auction Protocol Revisited

Jannik Dreier[1], Jean-Guillaume Dumas[2], Pascal Lafourcade[1]

[1]Verimag and [2]Laboratoire Jean Kuntzmann (LJK),
Université Grenoble 1, CNRS, FRANCE

Africacrypt, Cairo, Egypt

June 23, 2013

Challenges in e-Auctions

- Competing parties:
  - Bidders/Buyers

  - Seller

  - Auctioneer

- Many possible mechanisms: English, Dutch, Sealed Bid, ...

Fairness

Verifiability

Non-Repudiation

Non-Cancellation

Security Requirements

Privacy

Receipt-Freeness

Coercion-Resistance

Anonymity

Plan

Protocol by Brandt [Bra06]

- Completely distributed protocol, no authorities
- Distributed homomorphic n-out-of-n threshold ElGamal encryption
- Bidders compute function f where $f_{ij} = 1$ if bidder i won at price j, $f_{ij} \neq 1$ otherwise.
- Each bidder i only learns "his" f_{ij}, i.e. only if he won or lost
- Zero-Knowledge Proofs (ZKP) to protect against misbehaving parties

Protocol execution

1. Distributed key setup

Protocol execution

1. Distributed key setup

2. Encrypted bids

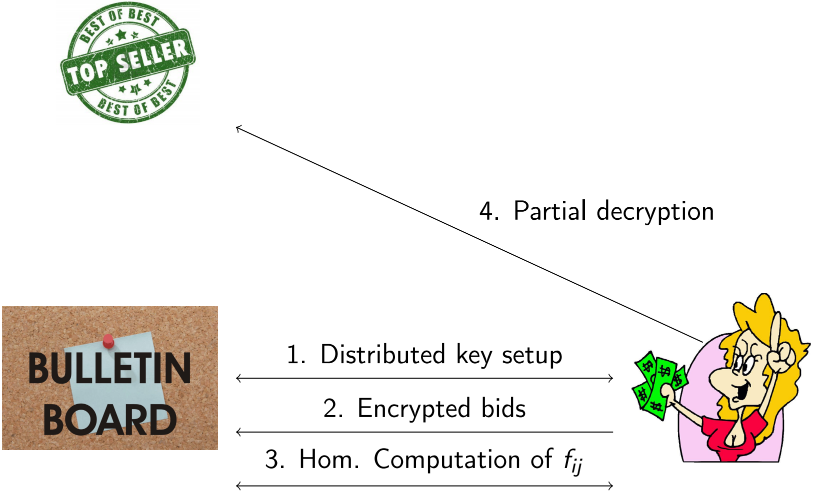
Protocol execution

1. Distributed key setup ⟷

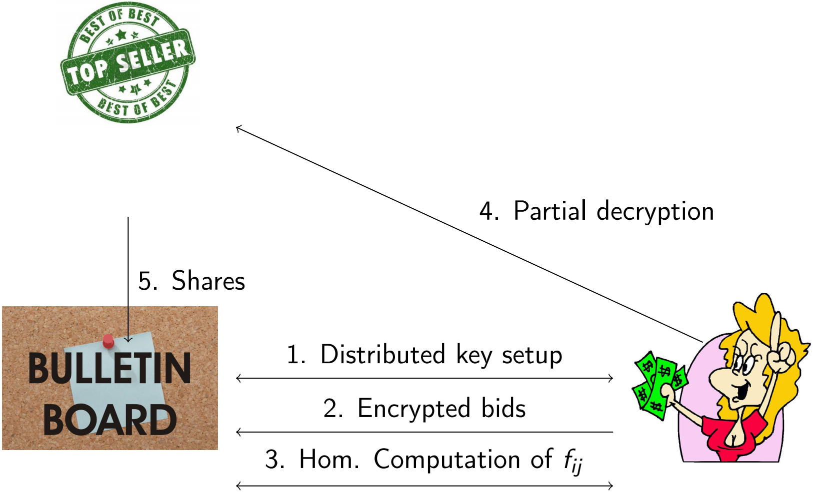
2. Encrypted bids ⟵

3. Hom. Computation of f_{ij} ⟷

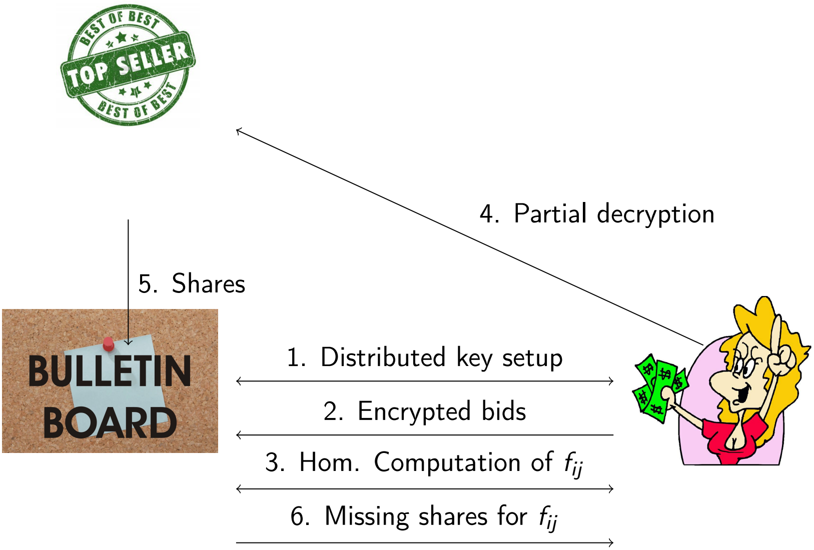
Protocol execution



Protocol execution



Protocol execution



For a public constant $Y \neq 1$:

$$b_{aj} = \begin{cases} Y & \text{if } j = bid_a \\ 1 & \text{otherwise} \end{cases}$$

Example: $bid_1 = 3$, $bid_2 = 1$ and $bid_3 = 2$. Then

$$b_1 = \begin{pmatrix} b_{1,4} \\ b_{1,3} \\ b_{1,2} \\ b_{1,1} \end{pmatrix} = \begin{pmatrix} 1 \\ Y \\ 1 \\ 1 \end{pmatrix}, b_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ Y \end{pmatrix}, b_3 = \begin{pmatrix} 1 \\ 1 \\ Y \\ 1 \end{pmatrix}$$

f_{ij}, example

Definition:

$$\tilde{f}_{ij}(X) = \left(\overbrace{\prod_{h=1}^{n} \prod_{d=j+1}^{k} X_{hd}}^{\text{bigger prices, all bidders}} \right) \cdot \left(\overbrace{\prod_{d=1}^{j-1} X_{id}}^{\text{lower prices, same bidder}} \right) \cdot \left(\overbrace{\prod_{h=1}^{i-1} X_{hj}}^{\text{ties using index}} \right), \; f_{ij} = \left(\tilde{f}_{ij}(b) \right)^{r_{i,j}}$$

Hence:

$$b_1 = \begin{pmatrix} 1 \\ Y \\ 1 \\ 1 \end{pmatrix} \qquad \tilde{f}_1(b) = \begin{pmatrix} & & & Y*1*1 \\ 1* & 1* & 1* & 1*1 \\ Y*1* & 1*1* & 1*1* & 1 \\ 1*Y*1* & 1*1*1* & Y*1*1 & \end{pmatrix} = \begin{pmatrix} Y \\ 1 \\ Y \\ Y^2 \end{pmatrix}$$

$$b_2 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ Y \end{pmatrix} \qquad \tilde{f}_2(b) = \begin{pmatrix} & 1*1*Y* & 1 \\ 1* & 1*Y* & Y \\ Y* & Y* & 1 \\ Y^2* & & 1 \end{pmatrix} = \begin{pmatrix} Y \\ Y^2 \\ Y^2 \\ Y^2 \end{pmatrix}$$

$$b_3 = \begin{pmatrix} 1 \\ 1 \\ Y \\ 1 \end{pmatrix} \qquad \tilde{f}_3(b) = \begin{pmatrix} & 1*Y*1* & 1*1 \\ 1* & Y*1* & Y*1 \\ Y* & 1* & 1*1 \\ Y^2* & & 1*Y \end{pmatrix} = \begin{pmatrix} Y \\ Y^2 \\ Y \\ Y^3 \end{pmatrix}$$

$$b = (b_1, b_2, b_3)$$

- Observation: If $r_{ij} = 1$ for all i and j, then f is injective and efficiently invertible (proof in the paper).
- r_{ij} is jointly chosen by the bidders
- If malleable proofs of knowledge are used, a malicious bidder can set $r_{ij} = 1$
- Allows the seller to invert f and obtain all bidders' private bids

When computing

$$\gamma_{ij}^{a} = \left(\tilde{f}_{ij}(\alpha)\right)^{m_{ij}^{a}} \text{ and } \delta_{ij}^{a} = \left(\tilde{f}_{ij}(\beta)\right)^{m_{ij}^{a}},$$

wait until all other bidders published their γ_{ij}^{a} and δ_{ij}^{a}. Submit

$$\gamma_{ij}^{\omega} = \left(\tilde{f}_{ij}(\alpha)\right) \cdot \left(\prod_{k \neq \omega} \gamma_{ij}^{k}\right)^{-1} \text{ and } \delta_{ij}^{\omega} = \left(\tilde{f}_{ij}(\beta)\right) \cdot \left(\prod_{k \neq \omega} \delta_{ij}^{k}\right)^{-1}.$$

Then $r_{ij} = \sum_{a} m_{ij}^{a} = 1 - \sum_{a \neq \omega} m_{ij}^{a} + \sum_{a \neq \omega} m_{ij}^{a} = 1$.

Proof of Knowledge of x:

	Peggy		*Victor*
Secret :	x		
Public :	$g, v = g^x$		g
	$z = g^r$	$\xrightarrow{1:\ z}$	
		$\xleftarrow{2:\ c}$	c
	$s = r + c \cdot x$	$\xrightarrow{3:\ s}$	
Check :			$g^s \stackrel{?}{==} z \cdot v^c$

How to fake the proofs

Proof of Knowledge of x:

	Peggy		*Victor*
Secret :	x		
Public :	$g, v = g^x$		g
	$z = g^r$	$\xrightarrow{1:\ z}$	
		$\xleftarrow{2:\ c}$	c
	$s = r + c \cdot x$	$\xrightarrow{3:\ s}$	
Check :			$g^s \stackrel{?}{==} z \cdot v^c$

$$g^s = g^{r+c\cdot x} = g^r \cdot g^{x\cdot c} = z \cdot v^c$$

How to fake the proofs

Proof of Knowledge of $(1-x)$ using Proof of Knowledge of x:

	Peggy		Mallory		Victor
Secret :	x				
Public :	$g, v = g^x$		$g, w = gv^{-1} = g^{1-x}$		g
	$z = g^r$	$\xrightarrow{1:\ z}$	$y = z^{-1}$	$\xrightarrow{1':\ y}$	
		$\xleftarrow{2:\ c}$	c	$\xleftarrow{2':\ c}$	c
	$s = r + c \cdot x$	$\xrightarrow{3:\ s}$	$u = c - s$	$\xrightarrow{3':\ u}$	
Check :			$g^s \stackrel{?}{==} z \cdot v^c$		$g^u \stackrel{?}{==} y \cdot w^c$

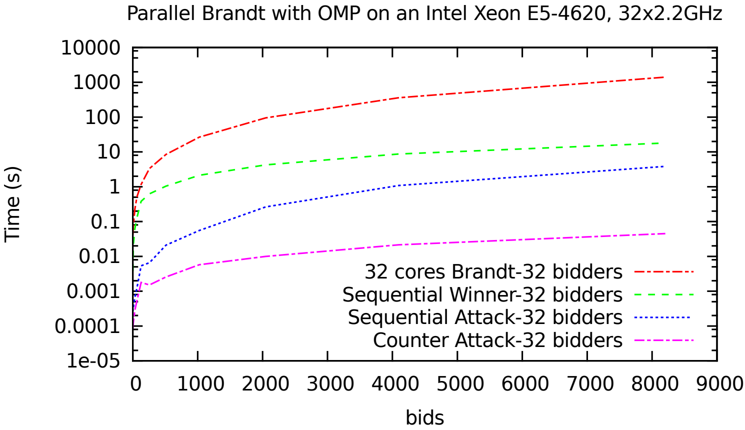
How to fake the proofs

Proof of Knowledge of $(1 - x)$ using Proof of Knowledge of x:

	Peggy		Mallory		Victor
Secret :	x				
Public :	$g, v = g^x$		$g, w = gv^{-1} = g^{1-x}$		g
	$z = g^r$	$\xrightarrow{1:\ z}$	$y = z^{-1}$	$\xrightarrow{1':\ y}$	
		$\xleftarrow{2:\ c}$	c	$\xleftarrow{2':\ c}$	c
	$s = r + c \cdot x$	$\xrightarrow{3:\ s}$	$u = c - s$	$\xrightarrow{3':\ u}$	
Check :			$g^s \stackrel{?}{==} z \cdot v^c$		$g^u \stackrel{?}{==} y \cdot w^c$

$$g^u = g^{c-s} = g^{c-r-c\cdot x} = g^{-r+(1-x)\cdot c} = g^{-r} \cdot g^{(1-x)\cdot c} = y \cdot w^c$$

- Bug in the $\mathcal{O}(nk^2)$ algorithm in the paper, corrected version in $\mathcal{O}(n^2k^2)$ in technical report [DDL12]
- With optimizations in $\mathcal{O}(nk)$
- Prototype implementation:



Privacy, second attack

Exploit the lack of authentication:

- Target one bidder
- Impersonate all other bidders
- Resubmit the targeted bidder's bid as their bids
- Impersonate the seller
- Obtain winning price=targeted bidder's bid

Verifiability:

- No authentication of the bids, hence no verification who actually submitted the bids
- $r_{ij} = 0$ implies $f_{ij} = 1$, hence several “winners” possible
- Partial decryption phase: Need to prove the use of the correct key, otherwise “nobody wins”

Other attacks

- Non-repudiation: Lack of authentication
- Fairness: An attacker can impersonate all bidders, hence controlling winner and winning price.

How to fix the protocol

Countermeasures against the identified issues:

- Use of non-interactive or non-malleable zero-knowledge proofs
- Authentication of all messages
- Bidders need to prove that the value x_a they use to decrypt is the same they used to generate their public key
- When computing the γ_{ij}^a and δ_{ij}^a the bidders can check if the product is equal to one – if yes, they restart the protocol using different keys and random values

- Analyzed Brandt's Fully Private Auction Protocol
- Completely distributed protocol designed for high privacy
- However: No authentication of the messages
- Attacks on Verifiability, Privacy, Fairness and Non-Repudiation
- Malleable ZKPs allow for an efficient attack on privacy
- Corner cases can lead to unexpected results, but are detectable
- Proposed four simple fixes

Thank you for your attention!

Questions?

jannik.dreier@imag.fr

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

Jannik Dreier, Jean-Guillaume Dumas, and Pascal Lafourcade.
Attacking privacy in a fully private auction protocol.
CoRR, abs/1210.6780, 2012.

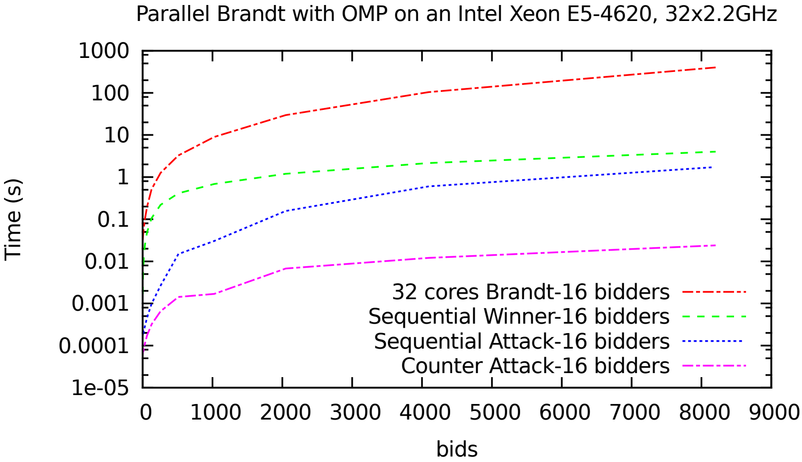
Protocol description I

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 protocol:

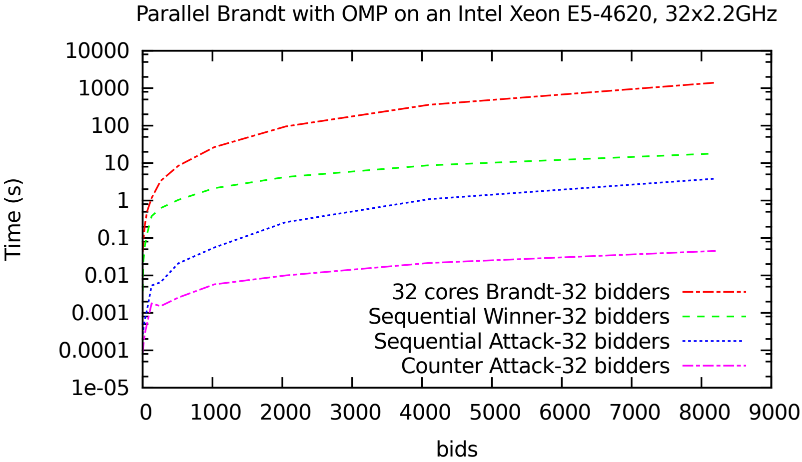
1. **Key Generation**
   Each bidder a, whose bidding price is bid_a among $\{1, \ldots, k\}$ does the following:
   - chooses a secret $x_a \in \mathbb{Z}/q\mathbb{Z}$
   - chooses randomly m^a_{ij} and $r_{aj} \in \mathbb{Z}/q\mathbb{Z}$ for each i and j.
   - publishes $y_a = g^{x_a}$ and proves the knowledge of y_a's discrete logarithm.
   - using the published y_i then computes $y = \prod_{i=1}^{n} y_i$.

1. **Bid Encryption**
   Each bidder a
   - 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 $\log_y\left(\frac{\prod_{j=1}^{k}\alpha_{aj}}{Y}\right) = \log_g\left(\prod_{j=1}^{k}\beta_{aj}\right)$.
2. **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.

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

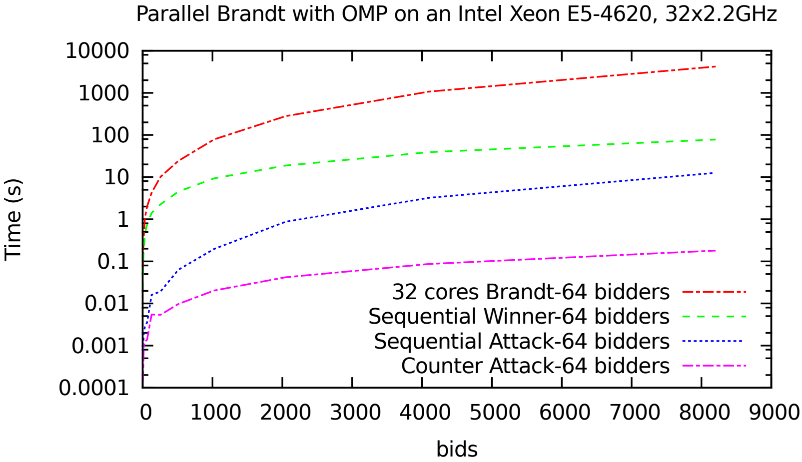

Parallel Brandt with OMP on an Intel Xeon E5-4620, 32x2.2GHz
Time (s)
1000
100
10
1
0.1
0.01
0.001
0.0001
1e-05
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
bids
32 cores Brandt-16 bidders
Sequential Winner-16 bidders
Sequential Attack-16 bidders
Counter Attack-16 bidders

Timings II



Parallel Brandt with OMP on an Intel Xeon E5-4620, 32x2.2GHz
Time (s)
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
1e-05
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
bids
32 cores Brandt-32 bidders
Sequential Winner-32 bidders
Sequential Attack-32 bidders
Counter Attack-32 bidders

Timings III



Parallel Brandt with OMP on an Intel Xeon E5-4620, 32x2.2GHz
Time (s)
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
bids
32 cores Brandt-64 bidders
Sequential Winner-64 bidders
Sequential Attack-64 bidders
Counter Attack-64 bidders