Analysis of a Receipt-Free Auction Protocol in the Applied Pi Calculus

Naipeng Dong*, Hugo Jonker, and Jun Pang

Faculty of Sciences, Technology and Communication
University of Luxembourg, Luxembourg

Abstract. We formally study two privacy-type properties in online auction protocols, bidding-price-secrecy and receipt-freeness. These properties are formalised as observational equivalences in the applied π calculus. We analyse the receipt-free auction protocol by Abe and Suzuki. Bidding-price-secrecy of the protocol is verified using ProVerif, whereas receipt-freeness of the protocol is proved manually.

1 Introduction

Auctions are ways to negotiate the exchange of goods and commodities. In an auction, a seller offers an item for bid, buyers submit bids, and the seller sells the item to the buyer with the highest bid. Nowadays, with the widely spread use of the Internet, online auctions are more and more often used as a convenient way to trade. Real-life examples are well-known websites like *eBay, eBid, Yahoo!auctions* and so on. Online auction protocols are also the subject of an active field of research [1–6].

For online auction systems, privacy is a fundamental property, e.g. personal information of bidders should not be revealed to other persons or companies. In order to protect the privacy of bidders, the following basic privacy-type properties are required.

Bidding-price-secrecy: A protocol preserves bidding-price-secrecy if an adversary cannot determine the bidding price of any bidder.

Receipt-freeness: A protocol satisfies receipt-freeness if a bidder cannot prove how he bids to an adversary.

We study the protocol AS02 proposed by Abe and Suzuki [4]. Abe and Suzuki claim that their protocol satisfies the above two requirements for non-winning bidders and provide an informal analysis. However, security protocols are notoriously difficult to design and analyse, and proofs of security protocols are known to be error-prone, thus we do not want to rely on an informal analysis. In several cases, formal verification found security flaws in protocols which were thought to be secure [7, 8]. Formal verification has shown its strength in finding attacks and proving correctness of security protocols. In this paper, we

* Supported by a grant from the Fonds National de la Recherche (Luxembourg).

formally verify whether bidding-price-secrecy and receipt-freeness hold in their protocol. We model the AS02 protocol using the applied π calculus [9]. The applied π calculus provides an intuitive way to model concurrent systems, especially security protocols. Moreover, it is supported by ProVerif [10], a verification tool which can be used to verify a number of security properties automatically. As suggested in [11], we use observational equivalence to express bidding-price-secrecy and receipt-freeness in the applied π calculus. Previously, formalisation of privacy-type properties has already been successfully executed in the domain of voting [12, 11] (similar ideas were developed in a different formal framework [13]). Bidding-price-secrecy for the AS02 protocol is verified automatically using ProVerif, whereas receipt-freeness is proven manually. We show that both of the two properties hold for non-winning bidders.

2 The applied π calculus

For better understanding of the paper, we introduce the applied π calculus briefly in this section, including its syntax, semantics and the definition of observational equivalence (for more details, see [9]). The applied π calculus is a language for modelling concurrent systems, in particular security protocols. We use the applied π calculus for its two main advantages: it provides an intuitive way to describe a protocol and cryptographic primitives can be defined by users.

Syntax. The calculus assumes an infinite set of names (which are used to represent communication channels or other atomic data), an infinite set of variables and a signature Σ consisting of a finite set of function symbols, which are used to model cryptographic primitives. Terms are defined as names, variables, and function symbols applied to terms. An equational theory E is defined as a set of equations on terms. The equivalence relation induced by E is denoted as $=_E$. Systems are described as processes: plain processes and extended processes. Plain processes are defined as:

$P, Q, R ::=$	plain processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction
if $M =_E N$ then P else Q	conditional
$\text{in}(u, x).P$	message input
$\text{out}(u, M).P$	message output.

Null process 0 does nothing. Parallel composition $P \mid Q$ represents sub-process P and sub-process Q running in parallel. Replication $!P$ represents an infinite number of process P running in parallel. Name restriction $\nu n.P$ bounds name n in process P, which means name n is secret to adversary. Term $M =_E N$ represents equality over the equational theory rather than strict syntactic identity. Message input $\text{in}(u, x).P$ reads a message from channel u, and bounds the

message to variable x in the following process P. Message output $\text{out}(u, M).P$ sends message M on channel u, and then runs process P. We can also write "let $x = M$ in P" to represent $P\{^M/_x\}$. Extended processes add variable restrictions and active substitutions. By restricting names and variables, we can bind a name or a variable to certain processes. An active substitution $\{^M/_x\}$ means a variable x can be replaced by term M in every process it comes into contact with. We say an extended process is closed if all its variables are either bounded or defined by an active substitution. The process $\nu x.(\{^M/_x\} \mid P)$ corresponds exactly to "let $x = M$ in P". Active substitutions allow us to map an extended process A to its frame $\varphi(A)$ by replacing every plain process in A with the null process 0, which does nothing. A frame is defined as an extended process built up from 0 and active substitutions by parallel composition and restrictions. The frame $\varphi(A)$ can be viewed as an approximation of A that accounts for the static knowledge A exposes to its environment, but not A's dynamic behaviour. The domain of a frame φ, denoted as $\mathsf{dom}(\varphi)$, is the set of variables for which the frame φ defines a substitution. A context $C[\ ]$ is defined as an extended process with a hole. An evaluation context is a context whose hole is not in the scope of a replication, a condition, an input, or an output. A context $C[\ ]$ closes A when $C[A]$ is closed.

Semantics. Two operational semantics are mentioned in this paper: internal reductions, denoted as $\rightarrow$, and labelled reductions, denoted as $\xrightarrow{\alpha}$. Internal reductions allow a process to execute without contacting its environment, for example, internal sub-processes communicate with each other, or the process evaluates and executes conditional operations (if-then-else). Labelled reductions are used to reason about processes that interact with their contexts. The transition $A \xrightarrow{\alpha} B$ means process A performs α action and continues as process B. Action α is either reading a term M from the process's context, or sending a name or a variable of base type to the context. Specificlly, when the output is a term M, $out(u, M).P$ is rewritten into $\nu x.(\{^M/_x\} \mid P)$.

Adversary model. To model security protocols, adversaries need to be taken into consideration. Following the Dolev-Yao model [14], an adversary has full control of the network. An adversary can eavesdrop, replay, block and inject messages. An adversary can be modelled as an arbitrary process running in parallel with the protocol, which can interact with the protocol in order to gain information.

Observational equivalence. Observational equivalence means an adversary cannot distinguish two processes. Intuitively, two processes are equivalent if they output on the same channels, no matter what the context they are placed in.

Definition 1 (Observational equivalence [9]). Observational equivalence *is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\ \mathcal{R}\ B$ implies:*

1. *if A can send a message on channel c, then B can also send a message on channel c;*
2. *if $A \rightarrow^* A'$ then, for some B', there exists $B \rightarrow^* B'$, and $A'\ \mathcal{R}\ B'$;*

3. $C[A]\ \mathcal{R}\ C[B]$ *for all closing evaluation contexts* C.

In practice, observational equivalence is hard to use, because of the quantification over contexts. Therefore, labelled bisimilarity is introduced. Labelled bisimilarity is easier to reason with manually or automatically. Two notations are used in labelled bisimilarity: *static equivalence* ($\approx_s$) and *labelled bisimilarity* ($\approx_\ell$). Static equivalence compares the static states of processes (represented by their frames), while labelled bisimilarity examines their dynamic behaviour.

Definition 2 (Labelled bisimilarity [9]). Labelled bisimilarity *($\approx_\ell$) is defined as the largest symmetric relation $\mathcal{R}$ on closed extended processes, such that process $A\ \mathcal{R}\ B$ implies:*

1. $A \approx_s B$;
2. *if* $A \to A'$ *then* $B \to^* B'$ *and* $A'\ \mathcal{R}\ B'$ *for some* B';
3. *if* $A \xrightarrow{\alpha} A'$ *and* $\mathsf{fv}(\alpha) \subseteq \mathsf{dom}(A)$ *and* $\mathsf{bn}(\alpha) \cap \mathsf{fn}(B) = \emptyset$; *then* $B \to^* \xrightarrow{\alpha} \to^* B'$ *and* $A'\ \mathcal{R}\ B'$ *for some* B'.

Labelled bisimilarity and observational equivalence coincide [9].

3 AS02 sealed-bid online auction protocol

Sealed-bid auctions are a type of auction in which bidders submit their bids without knowing what other bidders bid. The bidder with the highest bid wins the auction and pays the price he submitted.

Abe and Suzuki propose a sealed-bid auction protocol [4]. This protocol involves n bidders $\mathsf{b}_1, \ldots, \mathsf{b}_n$ and k auctioneers $\mathsf{a}_1, \ldots, \mathsf{a}_k$. A price list is published before the protocol. During the protocol, each bidder sends a commit for *every* price in the price list: 'yes' if he wants to bid that price, 'no' otherwise. Auctioneers work together to open the commitments of all bidders from the highest price down until the winning bid(s) is/are found.

In order to ensure privacy of bidders, the protocol has two physical assumptions: a bidding booth for the bidders, and a one-way untappable channel from every bidder to every auctioneer. The bidding booth enables a bidder to privately submit a bid free from control or observation of a coercer/buyer. The untappable channels ensure no eavesdropper can see messages sent.

Before starting the protocol, one auctioneer publishes an increasing price list $\mathsf{p}_1, \ldots, \mathsf{p}_m$, a message M_{yes} for "I bid", a message M_{no} for "I do not bid", a generator g of subgroup of $\mathbb{Z}_p^*$ with order q, where q, p are large primes with $p = 2q + 1$. The protocol consists of two phases: bidding and opening.

Bidding phase. A bidder in the bidding booth chooses a secret key x, publishes his public key $h = g^x$ with a predetermined signature. Then the bidder chooses a series of random numbers $r_1, \ldots, r_m$ as secret seeds, one random number for each price, and decides a price p to bid for. Then he generates a bit-commitment for each price p_ℓ $(1 \le \ell \le m)$, using the following formula:

$$Commit_\ell = \begin{cases} g^{M_{yes}} h^{r_\ell} & \text{if } \mathsf{p}_\ell = \mathsf{p} \quad \text{(a bid for price } \mathsf{p}\text{)} \\ g^{M_{no}} h^{r_\ell} & \text{if } \mathsf{p}_\ell \neq \mathsf{p} \quad \text{(not a bid for price } \mathsf{p}_\ell\text{)} \end{cases}$$

Next, the bidder publishes the sequence of the bit-commitments with his signature. Then he proves to each auctioneer that he knows the secret key $\log_g h = x$ and the discrete logs $\log_g Commit_\ell$ using interactive zero-knowledge proofs. Finally, he computes t-out-of-k[1] secret shares r_ℓ^i for each secret seed r_ℓ and each auctioneer a_i, and then sends the signed secret share r_ℓ^i over the one-way untappable channel to the auctioneer a_i.

Opening phase. Auctioneers together iterate the following steps for each price $p_\ell = \mathsf{p}_m, \mathsf{p}_{m-1}, \ldots, \mathsf{p}_1$ until the winning bid is determined.

Each auctioneer a_i publishes secret shares r_ℓ^i (the ℓ-th secret share of a bidder sent to auctioneer a_i) of all bidders. For each bidder, all auctioneers work together to reconstruct the secret seed r_ℓ, and check for each bidder whether

$$Commit_\ell \stackrel{?}{=} g^{M_{yes}} h^{r_\ell}.$$

If there exists some bidders that the above equivalences of those bidders are satisfied, the auctioneers finish checking the current price and stop. In this case, the price ℓ is the winning price, those bidders are winning bidders. If there is no equivalence existing, which means there is no bidder bidding for the price ℓ, the auctioneers repeat the above process on the next lower price.

Informal reasoning of receipt-freeness. We use M to represent either M_{yes} or M_{no}, the formula for computing $Commit_\ell$ is of the following form:

$$Commit_\ell = g^M \cdot h^{r_\ell} = g^M \cdot (g^x)^{r_\ell} = g^{M+xr_\ell} \ ,$$

since $h = g^x$. Thus, $\log Commit_\ell = M + xr_\ell$. By using interactive zero-knowledge proofs, a bidder is proved to know his secret key x and discrete logs $\log Commit_\ell$. An interesting property of chameleon bit commitments is that if the bidder bids for price p_ℓ,

$$\log Commit_\ell = M_{yes} + xr_\ell$$

he can calculate a fake r'_ℓ such that:

$$\log Commit_\ell = M_{no} + xr'_\ell \quad \text{and} \quad r'_\ell = (M_{yes} + xr_\ell - M_{no})/x.$$

Using the fake r'_ℓ, the bidder can show that bit-commitment $Commit_\ell$ is opened as message M_{no}, which means bidder did not bid for price ℓ. Using the same method, a bidder can open a 'no' bit-commitment, as a 'yes' bit-commitment. Thus, the commit leaks no information concerning the bid, thus the bidder cannot prove how he bid, e.g. receipt-freeness is satisfied.

[1] t is a threshold, k is the number of auctioneers, it means only more than t auctioneers together can reconstruct the secret seeds.

4 Modelling

We use the applied π calculus to model the AS02 protocol.[2] We use two simplifications. In the protocol, auctioneers are cooperating to find the winning bid. It takes at least t auctioneers to decide the winner, thus guaranteeing t-out-of-k secrecy. As we focus on bidder privacy, we need to consider only one honest auctioneer. Thus, we simplified the model to have only one honest auctioneer. The AS02 protocol uses interactive zero knowledge proofs to guarantee that each bidder knows his secret key and the discrete logs of bit-commitments. However, the details of these proofs are left unspecified, and thus we did not include them in the model. We simply assume that each bidder knows his secret key and discrete logs of bit-commitments.

Signature and equational theory. The signatures and the equational theory model cryptographic primitives used in the protocol. We fix a list of bidders $(\mathsf{b}_1, \ldots, \mathsf{b}_n)$ and an ordered list of prices $(\mathsf{p}_1, \ldots, \mathsf{p}_m)$, which are modelled as functions with arity 0. We define function $\mathsf{nextbidder}$ to find the next bidder in the bidder list, and function $\mathsf{nextprice}$ to find the next lower price in the price list. Function $\mathsf{checksign}$ is used to check whether the public signature key is the right one for the signed message, and we use function getmsg to get the original message from a signed message. Particularly, chameleon bit commitments are modeled as a function commit with arity 3 (random number, public key of the bidder and message M either M_{yes} or M_{no}). The relevant properties of chameleon bit commitments are captured in the following equational theory.

$$\begin{aligned}
\mathsf{commit}(r, \mathsf{pk}(sk_b), \mathsf{M}_{yes}) &= \mathsf{commit}(\mathsf{f}(r), \mathsf{pk}(sk_b), \mathsf{M}_{no}) \\
\mathsf{commit}(r, \mathsf{pk}(sk_b), \mathsf{M}_{no}) &= \mathsf{commit}(\mathsf{f}(r), \mathsf{pk}(sk_b), \mathsf{M}_{yes}) \\
\mathsf{open}(\mathsf{commit}(r, pk, m), r, pk) &= m
\end{aligned}$$

Constants M_{no} and M_{yes} represent "I do not bid" and "I bid", respectively. The parameter $\mathsf{pk}(sk_b)$ is the public key of bidder b, and r is the secret seed the bidder chooses. Function $\mathsf{f}(r)$ returns the fake secret seed of a secret seed r. We can model the function f by just giving one parameter - the real secret seed. Because we assume that each bidder knows his secret key and discrete logs of bit-commitments, he can compute the fake secret seed for each real secret seed, as explained in the previous section. The first equivalence means that if a bidder chooses a secret seed r, bids for a price, and calculates the bit commitment $\mathsf{commit}(r, \mathsf{pk}(sk_b), \mathsf{M}_{yes})$, he can compute a fake secret seed $\mathsf{f}(r)$, and by using this fake secret seed, the bit-commitment can be opened as message M_{no}, which means "I do not bid". The second equivalence shows that the opposite situation also holds. A bidder can also open a bit-commitment as if he bids for that price, when actually he does not.

[2] The complete model in ProVerif is available on `http://satoss.uni.lu/members/naipeng/publications.php`.

$$
\begin{array}{l}
P \triangleq \nu\ privch_{\mathsf{b}_1} \cdot\ \nu\ privch_{\mathsf{b}_2} \cdot\ \ldots \cdot\ \nu\ privch_{\mathsf{b}_n} \cdot \\
\quad \nu\ untapch_{\mathsf{b}_1} \cdot\ \nu\ untapch_{\mathsf{b}_2} \cdot\ \ldots \cdot\ \nu\ untapch_{\mathsf{b}_n} \cdot \\
\quad \nu\ synch \cdot \\
\quad (P_K \mid (\mathtt{let}\ p_b = p_{\mathsf{b}_1}\ \mathtt{in\ let}\ untapch = untapch_{\mathsf{b}_1}\ \mathtt{in} \\
\qquad \mathtt{let}\ privch = privch_{\mathsf{b}_1}\ \mathtt{in\ let}\ ch = ch_1\ \mathtt{in}\ P_B) \mid \\
\quad \ldots \mid (\mathtt{let}\ p_b = p_{\mathsf{b}_n}\ \mathtt{in\ let}\ untapch = untapch_{\mathsf{b}_n}\ \mathtt{in} \\
\qquad \mathtt{let}\ privch = privch_{\mathsf{b}_n}\ \mathtt{in\ let}\ ch = ch_n\ \mathtt{in}\ P_B) \mid P_A)
\end{array}
$$

Fig. 1. The main process.

$$
\begin{array}{l}
P_K \triangleq \nu\ ssk_{\mathsf{b}_1} \cdot\ \nu\ ssk_{\mathsf{b}_2} \cdot\ \ldots \cdot\ \nu\ ssk_{\mathsf{b}_n} \cdot \\
\quad \mathtt{let}\ spk_{\mathsf{b}_1} = \mathsf{pk}(ssk_{\mathsf{b}_1})\ \mathtt{in} \\
\quad \ldots \\
\quad \mathtt{let}\ spk_{\mathsf{b}_n} = \mathsf{pk}(ssk_{\mathsf{b}_n})\ \mathtt{in} \\
\quad (\mathtt{out}(privch_{\mathsf{b}_1}, ssk_{\mathsf{b}_1}) \mid \ldots \mid \mathtt{out}(privch_{\mathsf{b}_n}, ssk_{\mathsf{b}_n}) \mid \\
\quad \mathtt{out}(ch, spk_{\mathsf{b}_1}) \mid \ldots \mid \mathtt{out}(ch, spk_{\mathsf{b}_n}))
\end{array}
$$

Fig. 2. The key distribution process.

$$
\begin{array}{l}
P_B \triangleq \mathtt{in}(privch, ssk_b) \cdot \\
\quad \nu\ sk_b \cdot \mathtt{out}(ch, \mathsf{sign}(\mathsf{pk}(sk_b), ssk_b)) \cdot \\
\quad \nu\ r_1 \cdot \ldots \cdot \nu\ r_m \cdot \\
\quad \mathtt{if}\ \mathsf{p}_1 = p_b \\
\quad \mathtt{then\ let}\ cmt^{\mathsf{p}_1} = \mathsf{commit}(r_1, \mathsf{pk}(sk_b), \mathsf{M}_{yes})\ \mathtt{in} \\
\quad \mathtt{else\ let}\ cmt^{\mathsf{p}_1} = \mathsf{commit}(r_1, \mathsf{pk}(sk_b), \mathsf{M}_{no})\ \mathtt{in} \\
\quad \ldots \\
\quad \mathtt{if}\ \mathsf{p}_m = p_b \\
\quad \mathtt{then\ let}\ cmt^{\mathsf{p}_m} = \mathsf{commit}(r_m, \mathsf{pk}(sk_b), \mathsf{M}_{yes})\ \mathtt{in} \\
\quad \mathtt{else\ let}\ cmt^{\mathsf{p}_m} = \mathsf{commit}(r_m, \mathsf{pk}(sk_b), \mathsf{M}_{no})\ \mathtt{in} \\
\quad \mathtt{out}(ch, \mathsf{sign}((cmt^{\mathsf{p}_1}, \ldots, cmt^{\mathsf{p}_m}), ssk_b)) \cdot \\
\quad \mathtt{out}(untapch, (r_1, \ldots, r_m))
\end{array}
$$

Fig. 3. The bidder process.

$$
\begin{array}{l}
P_A \triangleq \mathtt{let}\ b = \mathsf{b}_1\ \mathtt{in}\ readinfo \mid \ldots \mid \mathtt{let}\ b = \mathsf{b}_n\ \mathtt{in}\ readinfo \mid \\
\quad \underbrace{\mathtt{in}(synch, vb_1) \cdot\ \ldots \cdot\ \mathtt{in}(synch, vb_n)}_{n} \cdot \\
\quad \mathtt{if}\ cmt^{\mathsf{p}_m}_{\mathsf{b}_1} = \mathsf{commit}(ss^{\mathsf{p}_m}_{\mathsf{b}_1}, pk_{\mathsf{b}_1}, \mathsf{M}_{yes}) \\
\quad \mathtt{then\ out}(winnerch, (\mathsf{p}_m, \mathsf{b}_1)) \cdot \\
\qquad \mathtt{if}\ \mathsf{nextbidder}(\mathsf{b}_1) = \bot \\
\qquad \mathtt{then}\ stop \\
\qquad \mathtt{else\ let}\ b = \mathsf{nextbidder}(\mathsf{b}_1)\ \mathtt{in\ let}\ p = \mathsf{p}_m\ \mathtt{in}\ checknextb \\
\quad \mathtt{else\ if}\ \mathsf{nextbidder}(\mathsf{b}_1) = \bot \\
\qquad \mathtt{then\ if}\ \mathsf{nextprice}(\mathsf{p}_m) = \top \\
\qquad\quad \mathtt{then}\ stop \\
\qquad\quad \mathtt{else\ let}\ b = \mathsf{b}_1\ \mathtt{in\ let}\ p = \mathsf{nextprice}(\mathsf{p}_m)\ \mathtt{in}\ checknextbnp \\
\qquad \mathtt{else\ let}\ b = \mathsf{nextbidder}(\mathsf{b}_1)\ \mathtt{in\ let}\ p = \mathsf{p}_m\ \mathtt{in}\ checknextbnp
\end{array}
$$

Fig. 4. The auctioneer process.

Main process. The main process is represented in Fig. 1. This process first generates private channels: $privch_{\mathsf{b}_j}$ for each bidder b_j to receive secret keys, $untapch_{\mathsf{b}_j}$ shared between each bidder b_j and the auctioneer, *synch* used by the auctioneer to collect all necessary information before moving to the opening phase. Note that *ch* is a public channel, and $p_{\mathsf{b}_1}, \ldots, p_{\mathsf{b}_n}$ are parameters, each of these parameters has to be instantiated with a constant in the published price list $\mathsf{p}_1, \ldots, \mathsf{p}_m$. Then the main process launches the key distribution sub-process, n (number of bidders) copies of bidder sub-processes and one auctioneer sub-process.

Key distribution process. The key distribution process P_K, presented in Fig. 2, generates a signature key ssk_{b_j} for each bidder B_j, sends it to the bidder through the private channel $privch_{\mathsf{b}_j}$, and publishes the corresponding public signature key. Therefore, only a bidder knows his own secret key, and everyone including the adversary knows each bidder's public signature key.

Bidder process. First, a bidder receives his secret signature key from his private channel. Next, the bidder generates his secret key sk_b, and chooses a series of random numbers $r_1 \ldots r_m$ as secret seeds. The bidder then computes each bit-commitment cmt^{p_ℓ} as described in Sect. 3. Finally, the bidder publishes the series of bit-commitments $cmt^{\mathsf{p}_1}, \ldots, cmt^{\mathsf{p}_m}$ with his signature, and sends the series of secret seeds to the auctioneer through the untappable channel. As we assume there is only one honest auctioneer in the model, we do not need to model secret shares. The applied π calculus process for a bidder P_B is given in Fig. 3.

Auctioneer process. During the bidding phase, the auctioneer launches n copies of sub-process *readinfo* to gather information from each bidder b_j, consisting of public signature key spk_{b_j}, signed public key $\mathsf{sign}(\mathsf{pk}(sk_{b_j}), ssk_{b_j})$, series of bit-commitments $cmt^{\mathsf{p}_1}_{\mathsf{b}_j}, \ldots, cmt^{\mathsf{p}_m}_{\mathsf{b}_j}$, and secret seeds $ss^{\mathsf{p}_1}_{\mathsf{b}_j}, \ldots, ss^{\mathsf{p}_m}_{\mathsf{b}_j}$. The auctioneer also needs to synchronise with all bidders. The auctioneer process is not allowed to continue until all bidders reach the end of the bidding phase. During the opening phase, the auctioneer evaluates $cmt^{\mathsf{p}_m}_{\mathsf{b}_j} \stackrel{?}{=} \mathsf{commit}(ss^{\mathsf{p}_m}_{\mathsf{b}_j}, pk_{\mathsf{b}_j}, \mathsf{M}_{yes})$ for each bidder. If the two values are equivalent, bidder b_j has bid for that price, otherwise, bidder b_j does not bid for that price. Once the auctioneer has determined the set of the winning bids, he publishes the set of the winning bidders together with the winning price through the public channel *winnerch*, and stops the process. Otherwise, the auctioneer repeats the evaluation steps for each bidder at the next lower price. In a similar way, the sub-process *checknextb* is used to evaluate the bid of a bidder b at price p, if there are already some winners before bidder b. And the sub-process *checknextbnp* is used to check the next bidder at price p, if there is no winner before that bidder. We use $\perp$ and $\top$ to represent the end of the bidder list and price list, respectively.

5 Analysis

After modelling the protocol in the previous section, now we formalise and analyse the two privacy-type properties: bidding-price-secrecy and receipt-freeness.

5.1 ProVerif

ProVerif is a tool for verifying security properties in cryptographic protocols. Given a security property as a query, ProVerif can take a protocol modelled as a process in the applied π calculus as input, and returns whether the protocol satisfies the security property.

In ProVerif, standard secrecy of a term M is defined as "an adversary cannot derive the term M". To check standard secrecy, we use query "*not attacker*: M". The positive query result means, no matter how the adversary interacts with the protocol, M will never be part of adversary's knowledge. Otherwise, ProVerif gives a counterexample to show how an adversary derives the term M.

In ProVerif, strong secrecy is defined as: for all closed substitutions σ and σ' of free variables in a process P, the process satisfies $P_\sigma \approx P_{\sigma'}$ (where $\approx$ denotes observational equivalence). To check strong secrecy of variable x, we can use the query "*noninterf* x". Intuitively, by instantiating x with different values, we obtain different versions of the given process. A protocol satisfies strong secrecy iff these different versions of the given process are observationally equivalent. The fundamental idea of observational equivalence checking in ProVerif is to focus on pairs of processes sharing the same structure and differing only in terms or destructors. ProVerif's reasoning about strong secrecy is sound but incomplete. If ProVerif reports that a process does not satisfy strong secrecy, there are two possibilities: either the process does not satisfy strong secrecy, or the process satisfies strong secrecy, but ProVerif cannot prove it.

5.2 Bidding-price-secrecy

Bidding-price-secrecy guarantees the anonymity of the link between a bidder and the price he bids for. In the AS02 protocol, the winning bid is published, and thus bidding-price-secrecy for the winning bidder is not satisfied. Particularly, if all bidders bid for the same price, then all bidders are winners, i.e. no bidder is a non-winning bidder, thus bidding-price-secrecy is not satisfied in this case. From here on, when we refer to bidding-price-secrecy, we mean only w.r.t. non-winning bids. There are two notions of secrecy: standard bidding-price-secrecy and strong bidding-price-secrecy.

Standard bidding-price-secrecy. Standard bidding-price-secrecy is defined as no matter how an adversary interacts with the protocol, he cannot determine which price in the price list a non-winning bidder has bid for.

In order to show that an adversary cannot determine the bidding price of a non-winning bidder, we can use the standard secrecy query in ProVerif. We have a winning bidder process in which a bidder submits the highest bid. And we have several other bidder processes. Each of these processes has a variable p_b representing the price the bidder bids for. The variable p_b can be instantiated by any price in the price list, except the highest price. By inquiring "*not attacker*: p_b", we check whether an adversary can derive the bidding price of a non-winning bidder. ProVerif gives us positive results, which means the protocol satisfies the property of standard bidding-price-secrecy.

Strong bidding-price-secrecy. Strong bidding-price-secrecy means an adversary cannot distinguish between the case when a bidder bids for price a and the case when a bidder bids for price c. In other words, an adversary cannot tell whether a bidder changes his bidding price from a to c. We use observational equivalence in the applied π calculus to formalise strong bidding-price-secrecy.

Similar formalisation has been used in the domain of voting. In [11], a similar property called vote-privacy is formalised as a process in which V_A votes for a and V_B votes for c is observationally equivalent to a process where V_A votes for c and V_B votes for a. The idea is that even all other voters reveal how they voted, an adversary cannot deduce the votes of V_A and V_B, given V_A and V_B counterbalance each other. Different from privacy in voting that the result is published, in auction protocols, normally a non-winning bidder's bidding price is not published. Therefore, we do not need a counterbalancing process. Instead, we need a process in which a bidder bids for higher price to stop the auctioneer process opening non-winning bids, so that non-winning bids are not revealed in the opening phase. Therefore, strong bid-price-secrecy is formalised as follows:

Definition 3 (Strong bidding-price-secrecy). *An auction protocol P, with a bidder sub-process represented as P_B, is strong bidding-price-secret if for all possible bidders b_1 and b_2 we have:*

$$S[P_B^1\{^a/_{p_b}\} \mid P_B^2\{^d/_{p_b}\}] \approx_\ell S[P_B^1\{^c/_{p_b}\} \mid P_B^2\{^d/_{p_b}\}]$$

with $a < d$ and $c < d$.

The context S is used to capture the assumption made on the checked protocol, usually it includes the other honest participants in the protocol. The process P_B^1 is a non-winning bidder process executed by bidder b_1. The process P_B^2 is a bidder process in which the bidder b_2 bids for a higher price d. The intuition is that an adversary cannot determine whether a non-winning bidder bids for price a or price c, provided there exists another bidder bids for a higher price d.

We define the context S as $\nu\tilde{r} \cdot (P_K \mid P_B\sigma_1 \mid \ldots \mid P_B\sigma_{n-2} \mid P_A \mid \_\,)$ in the AS02 protocol, where $\tilde{r}$ are channel names, P_K is the key distribution process, $P_B\sigma_i$ are some other honest bidder processes ($1 \leq i \leq n-2$), and P_A is the auctioneer process. The context is as the auction process with a hole instead of two bidder processes. We assume all the participants in the context are honest. In order to make it possible to check strong bidding-price-secrecy in ProVerif, we need to modify the presented auctioneer process. Note that ProVerif is sensitive to evaluations of statements in the if-then-else constructs. ProVerif reports false attacks, when using these constructions [15]. We can simplify the process by halting the process after checking price d, i.e. if-then-else constructs beyond the checking of price d are cut off. Since we assume there is a process bidding a high price d in the equivalence in the definition of strong bidding-price-secrecy, the auctioneer process will stop after checking price d (or even sooner), and the remaining part of the process will not be executed. Therefore, we can cut the remaining part of the auctioneer process without affecting the verification result. To be able to check *noninterf* in ProVerif, we modify the bidder process

by replacing if-then-else constructions with choice[ ] constructions (see [15] for more explanation). By querying "*noninterf* p_b *among* $\mathsf{p}_1, \ldots, \mathsf{p}_{d-1}$", the variable p_b is replaced with p_1 up to p_{d-1}, resulting into $d-1$ different versions of the process. ProVerif gives a positive result, which means that these process versions are all observationally equivalent. In this way, we prove that the protocol satisfies strong bidding-price-secrecy.

5.3 Receipt-freeness

Receipt-freeness means a bidder cannot prove to an adversary that he has bid in a certain way. It is useful to protect bidders from being coerced to show how they bid. Intuitively, bidding-price-secrecy protects a bidder's privacy when the bidder does not want to reveal his information, while receipt-freeness protects a bidder's privacy when the bidder is willing to reveal his information or the bidder is coerced to reveal his information.

In voting, receipt-freeness can be formalised as an observational equivalence [11]. A protocol satisfies receipt-freeness if there exists a process V', in which a voter votes for candidate a but feigns cooperation with the adversary by telling him that he votes for a different candidate c. The adversary cannot tell the difference between a situation in which a voter genuinely cooperates with him and the one in which the voter pretends to cooperate but actually votes for another candidate, as these two cases are observationally equivalent. In order to model observational equivalence, the situation that a voter is willing to provide his secret information to the adversary needs to be modelled first:

Definition 4 (Process P^{ch} [11]). *Let P be a plain process and ch a channel name. P^{ch}, the process that shares all of p's secrets, is defined as:*

- $0^{ch} \hat{=} 0$,
- $(P|Q)^{ch} \hat{=} P^{ch}|Q^{ch}$,
- $(\nu n.P)^{ch} \hat{=} \nu n.\mathsf{out}(ch, n).P^{ch}$ *when n is name of base type,*
- $(\nu n.P)^{ch} \hat{=} \nu n.P^{ch}$ *otherwise,*
- $(\mathsf{in}(u, x).P)^{ch} \hat{=} \mathsf{in}(u, x).\mathsf{out}(ch, x).P^{ch}$ *when x is a variable of base type,*
- $(\mathsf{in}(u, x).P)^{ch} \hat{=} \mathsf{in}(u, x).P^{ch}$ *otherwise,*
- $(\mathsf{out}(u, M).P)^{ch} \hat{=} \mathsf{out}(u, M).P^{ch}$,
- $(!P)^{ch} \hat{=} !P^{ch}$,
- $(\mathit{if}\ M =_E N\ \mathit{then}\ P\ \mathit{else}\ Q)^{ch} \hat{=} \mathit{if}\ M =_E N\ \mathit{then}\ P^{ch}\ \mathit{else}\ Q^{ch}$.

Delaune *et al.* also define process transformation $A^{\setminus out(ch,\cdot)}$, which can be considered as the process A hides $out(ch, \cdot)$ (the outputs on the public channel ch).

Definition 5 (Process $A^{\setminus out(ch,\cdot)}$ [11]). *Let A be an extended process. We define the process $A^{\setminus out(ch,\cdot)}$ as $\nu ch.(A|!in(ch, x))$.*

When modelling online auction protocols, we also need to model the situation a bidder wants to provide his secret information to an adversary. We use the above definition directly in our model. Intuitively, a bidder, who is willing to share information with the adversary, sends any input of base type, any freshly

generated names of base type to the adversary through a public channel ch. It is assumed that public channels are under adversaries' control.

Now, we can define receipt-freeness for online auction protocols. Again, we need a bidder process P_B^2 in which bidder b_2 bids for a higher price d, so that non-winning bids are not revealed. Intuitively, if a non-winning bid has a strategy to cheat the adversary, and the adversary cannot tell the difference between whether the bidder cheats or not, then the protocol is receipt-free.

Definition 6 (Receipt-freeness). *An auction protocol P, with a bidder sub-process P_B, is receipt-free if there exists a closed plain process ${P_B}'$ such that:*

1. ${P_B}'^{\backslash out(chc,\cdot)} \approx_\ell P_B^1\{^c/_{p_b}\}$,
2. $S[P_B^1\{^a/_{p_b}\}^{chc} \mid P_B^2\{^d/_{p_b}\}] \approx_\ell S[{P_B}' \mid P_B^2\{^d/_{p_b}\}]$

with $d > c$ and $d > a$.

Process ${P_B}'$ is a bidder process in which bidder b_1 bids for price c but communicates with the adversary and tells the adversary he bids for price a. Process $P_B^1\{^c/_{p_b}\}$ is a bidder process in which bidder b_1 bids for price c. Process $P_B^1\{^a/_{p_b}\}^{chc}$ is a bidder process in which bidder b_1 bids for price a and shares his secret with the adversary. Process P_B^2 is a bidder process in which bidder b_2 bids for a higher price d. The first equivalence says that ignoring the outputs bidder b_1 makes on the adversary channel chc, ${P_B}'$ looks like a normal process in which b_1 bidding for price c. The second equivalence says that the adversary cannot tell the difference between the situation in which b_1 obeys the adversary's commands and bids for price a, and the situation in which b_1 pretends to cooperate but actually bids for price c, provided there is a bidding process P_B^2 that bids higher, ensuring that bidding processes P_B^1 and ${P_B}'$ are not winner. Receipt-freeness is a stronger property than bidding-price-secrecy, as shown in [11].

$$
\begin{aligned}
{P_B}' \triangleq\ & \mathtt{in}(privch, ssk_b) \cdot\ \mathtt{out}(chc, ssk_b)\cdot \\
& \nu\ sk_b \cdot\ \mathtt{out}(chc, sk_b)\cdot \\
& \mathtt{out}(ch, \mathsf{sign}(\mathsf{pk}(sk_b), ssk_b))\cdot \\
& \nu\ r_1 \cdot\ \ldots \cdot\ \nu\ r_a \cdot\ \ldots \cdot\ \nu\ r_c \cdot\ \ldots \cdot\ \nu\ r_m \cdot \\
& \mathtt{out}(chc, (r_1, \ldots, \mathsf{f}(r_a), \ldots, \mathsf{f}(r_c), \ldots, r_m))\cdot \\
& \mathtt{let}\ cmt^{\mathsf{p}_1} = \mathsf{commit}(r_1, \mathsf{pk}(sk_b), \mathsf{M}_{no})\ \mathtt{in} \\
& \ldots \\
& \mathtt{let}\ cmt^{\mathsf{p}_a} = \mathsf{commit}(r_a, \mathsf{pk}(sk_b), \mathsf{M}_{no})\ \mathtt{in} \\
& \ldots \\
& \mathtt{let}\ cmt^{\mathsf{p}_c} = \mathsf{commit}(r_c, \mathsf{pk}(sk_b), \mathsf{M}_{yes})\ \mathtt{in} \\
& \ldots \\
& \mathtt{let}\ cmt^{\mathsf{p}_m} = \mathsf{commit}(r_m, \mathsf{pk}(sk_b), \mathsf{M}_{no})\ \mathtt{in} \\
& \mathtt{out}(ch, \mathsf{sign}((cmt^{\mathsf{p}_1}, \ldots, cmt^{\mathsf{p}_m}), ssk_b))\cdot \\
& \mathtt{out}(untapch, (r_1, \ldots, r_a, \ldots, r_c, \ldots, r_m))\cdot
\end{aligned}
$$

Fig. 5. The process ${P_B}'$.

For the AS02 protocol, the context S is defined the same as in the analysis of the bidding-price-secrecy property. To prove receipt-freeness, we need to find a

process ${P_B}'$ which satisfies both equivalences in the definition of receipt-freeness. According to the properties of chameleon bit commitment, the bidder can send a sequence of fake secret seeds to the adversary, and sends the series of real secret seeds to the auctioneer through an untappable channel. The adversary opens the bit-commitments as the bidder bids for price a, using the fake secret seeds he received, while the auctioneer opens the same bit-commitments as the bidder bids for price c, using the secret seeds the auctioneer received through a untappable channel. The process ${P_B}'$ is shown in Fig. 5. The bidder in this process communicates with the adversary through channel *chc*, sending the adversary his secret signature key ssk_b and his secret key sk_b. Later the bidder sends the auctioneer $r_1, \ldots, r_m$ through an untappable channel, and send the adversary the same list except changing r_a and r_c to $\mathsf{f}(r_a)$ and $\mathsf{f}(r_c)$, respectively. The untappable channel ensures the adversary cannot learn anything about the differences.

To prove the first equivalence, we can simply consider ${P_B}'^{\setminus out(chc,\cdot)}$ as process ${P_B}'$ without communication on the channel *chc*. Since the process ${P_B}'^{\setminus out(chc,\cdot)}$ is exactly the same as the process $P_B^1\{^c/_{p_b}\}$, the first equivalence of Def. 6 is satisfied. To show the second equivalence of Def. 6, we need to consider all the executions of each side. On both sides, the process P_K only distributes keys, and all the bidder processes in the context follow the same process. For the sake of simplicity, we can ignore the outputs in the process P_K and those bidder processes. During the bidding phase the auctioneer process only reads information and synchronises on the private channel *synch*. There is no output on public channels in the auctioneer process. We denote the sequence of names $sk_b, r_1, \ldots, r_m, bsk_b, br_1, \ldots, br_m$ by $\tilde{n}$. After the key distribution, we want to see whether the behaviour of the process $P_B^1\{^a/_{p_b}\}^{chc} \mid P_B^2\{^d/_{p_b}\}$ is observationally equivalent to ${P_B}' \mid P_B^2\{^d/_{p_b}\}$. For this purpose, we need to consider all possible executions of these two processes. Here, we consider a particular execution and only show the interesting part of the two frames after each step of execution by the two processes. Let $P = P_B^1\{^a/_{p_b}\}^{chc} \mid P_B^2\{^d/_{p_b}\}$ and $Q = {P_B}' \mid P_B^2\{^d/_{p_b}\}$.

$$
\begin{aligned}
P \xrightarrow{in(privch,ssk_b)} & \xrightarrow{in(privchb,bssk_b)} \xrightarrow{\nu\ x_1 \cdot\ out(chc,x_1)} P_1 \mid \{^{ssk_b}/_{x_1}\} \\
& \xrightarrow{\nu\ x_2 \cdot\ out(chc,x_2)} \nu\tilde{n} \cdot\ (P_2 \mid \{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\}) \\
& \xrightarrow{\nu\ x_3 \cdot\ out(ch,x_3)} \\
& \xrightarrow{\nu\ x_4 \cdot\ out(chc,x_4)} \nu\tilde{n} \cdot\ (P_3 \mid \{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_3}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_4}\}) \\
& \xrightarrow{\nu\ x_5 \cdot\ out(chc,x_5)} \nu\tilde{n} \cdot\ (P_4 \mid \{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_3}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_4}\} \mid \{^{r_1,\ldots,r_m}/_{x_5}\}) \\
& \xrightarrow{\nu\ x_6 \cdot\ out(ch,x_6)} \\
& \xrightarrow{\nu\ x_7 \cdot\ out(chc,x_7)} \nu\tilde{n} \cdot\ (P_5 \mid \{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_3}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_4}\} \\
& \qquad\qquad \mid \{^{r_1,\ldots,r_m}/_{x_5}\} \mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_6}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}((bcmt^{\mathsf{p}_1},\ldots,bcmt^{\mathsf{p}_m}),bssk_b)}/_{x_7}\})
\end{aligned}
$$

$$
\begin{aligned}
Q \xrightarrow{in(privch,ssk_b)} & \xrightarrow{in(privchb,bssk_b)} \xrightarrow{\nu\ x_1 \cdot\ out(chc,x_1)} Q_1 \mid \{^{ssk_b}/_{x_1}\} \\
& \xrightarrow{\nu\ x_2 \cdot\ out(chc,x_2)} \nu\tilde{n} \cdot\ (Q_2 \mid \{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\}) \\
& \xrightarrow{\nu\ x_3 \cdot\ out(ch,x_3)} \\
& \xrightarrow{\nu\ x_4 \cdot\ out(ch,x_4)} \nu\tilde{n} \cdot\ (Q_3 \mid \{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_3}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_4}\}) \\
& \xrightarrow{\nu\ x_5 \cdot\ out(chc,x_5)} \nu\tilde{n} \cdot\ (Q_4 \mid \{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_3}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_4}\} \\
& \qquad\qquad \mid \{^{r_1,\ldots,\mathsf{f}(r_a),\ldots,\mathsf{f}(r_c),\ldots,r_m}/_{x_5}\}) \\
& \xrightarrow{\nu\ x_6 \cdot\ out(ch,x_6)} \\
& \xrightarrow{\nu\ x_7 \cdot\ out(ch,x_7)} \nu\tilde{n} \cdot\ (Q_5 \mid \{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_3}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_4}\} \\
& \qquad\qquad \mid \{^{r_1,\ldots,\mathsf{f}(r_a),\ldots,\mathsf{f}(r_c),\ldots,r_m}/_{x_5}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_6}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}((bcmt^{\mathsf{p}_1},\ldots,bcmt^{\mathsf{p}_m}),bssk_b)}/_{x_7}\})
\end{aligned}
$$

The frames we obtained at the end of P and Q are statically equivalent. In particular, as the adversary knows the bit-commitments the bidder submits, the public key of the bidder, and the secret seeds, the adversary can open all the commitments. The only functions an adversary can use are getmsg and open. By applying these two functions, the adversary can get other terms, the public key of the bidder represented as $x_{msg} = \mathsf{getmsg}(x_3, x_1)$ and a series of opened messages. Since x_3 and x_1 are the same for both P and Q, x_{msg} is the same for both processes as well. Particularly, $P_B^1\{^a/_{p_b}\}$ bids for price a. The adversary opens the commitments $cmt^{p_a} = \mathsf{commit}(r_a, \mathsf{pk}(sk_b), \mathsf{M}_{yes})$ and $cmt^{p_c} = \mathsf{commit}(r_c, \mathsf{pk}(sk_b), \mathsf{M}_{no})$:

$$\mathsf{open}(cmt^{p_a}, r_a, \mathsf{pk}(sk_b)) = \mathsf{M}_{yes} \quad \mathsf{open}(cmt^{p_c}, r_c, \mathsf{pk}(sk_b)) = \mathsf{M}_{no}$$

For the process Q, the process ${P_B}'$ bids for price c. The adversary has a sequence of secret seeds, in which two of them are fake: $\mathsf{f}(r_a)$ and $\mathsf{f}(r_c)$. According to the equational theory of chameleon bit-commitments (see Sect. 4), the adversary opens $cmt^{p_a} = \mathsf{commit}(r_a, \mathsf{pk}(sk_b), \mathsf{M}_{no}) = \mathsf{commit}(\mathsf{f}(r_a), \mathsf{pk}(sk_b), \mathsf{M}_{yes})$ and $cmt^{p_c} = \mathsf{commit}(r_c, \mathsf{pk}(sk_b), \mathsf{M}_{yes}) = \mathsf{commit}(\mathsf{f}(r_c), \mathsf{pk}(sk_b), \mathsf{M}_{no})$ as follows:

$$\mathsf{open}(cmt^{p_a}, \mathsf{f}(r_a), \mathsf{pk}(sk_b)) = \mathsf{M}_{yes} \quad \mathsf{open}(cmt^{p_c}, \mathsf{f}(r_c), \mathsf{pk}(sk_b)) = \mathsf{M}_{no}$$

All other secret seeds and bit-commitments are the same in both P and Q, hence the adversary gets the same series of opened messages for both P and Q as well.

Next, we consider the opening phase, the auctioneer process is the only active process. According to the protocol, the auctioneer process stops after finding the winning bid. Therefore, non-winning bids are not revealed. Since we have assumed the auctioneer is honest, the information the auctioneer process reveals is the opened bit-commitments of all bidders at prices higher than the winning

price, and the winning bid. Only the winning bid is opened as M_{yes}, others are opened as M_{no}. Due to the existence of a higher bid (d in the process $P_B^2\{^d/_{p_b}\}$) on both sides of the equivalence, the bid made by the bidder b_1 will never be published, hence the information the auctioneer process reveals is the same. Now, we can conclude that the protocol satisfies receipt-freeness.

6 Conclusion

The main contribution of this paper is that we propose a formalisation of two privacy-type properties in auction protocols: bidding-price-secrecy and receipt-freeness, following definitions of vote privacy and receipt-freeness in voting [11]. There are two notions of bidding-price-secrecy: standard bidding-price-secrecy and strong bidding-price-secrecy. Standard bidding-price-secrecy is defined as that an adversary does not know a non-winning bidder's bidding price. Strong bidding-price-secrecy and receipt-freeness are modelled using observational equivalence. We have modelled the AS02 protocol in the applied π calculus, verified bidding-price-secrecy of the protocol automatically using ProVerif and receipt-freeness of the protocol manually.

Coercion-resistance in voting is a stronger privacy property than receipt-freeness [11], saying that a voter cannot cooperate with a coercer to prove to him that he voted in a certain way. It is modelled by giving the coercer the ability to communicate with the coercee and the ability to prepare information for the coercee to use [11]. In more details, coercion-resistance is formalised in the applied π calculus by requiring the existence of a process in which a voter can do as he wants, despite the presence of the coercer, and the coercer cannot tell whether the voter is cheating. According to this definition, it seems to us that the AS02 protocol is also coercion-resistant. The information a coercer can generate in the bidder process is: the bidder's secret key sk_b, the random number $r_1, \ldots, r_a, \ldots, r_c, \ldots r_m$, the bit-commitments $cmt^{\mathsf{p}_1}, \ldots, cmt^{\mathsf{p}_m}$. Since the zero-knowledge proof ensures the bidder knows his own secret key, as well as the discrete logs of bit-commitments, a bidder can figure out which price the coercer wants him to bid, and then calculate the fake secret seeds $\mathsf{f}(r_a)$ and $\mathsf{f}(r_c)$ to change the price the coercer calculated, and sends secret seeds $r_1, \ldots, r_{a-1}, \mathsf{f}(r_a), r_{a+1}, \ldots, r_{c-1}, \mathsf{f}(r_c), r_{c+1}, \ldots, r_m$ to the auctioneer.

Coercion-resistance is a complicated property to formalise. Several different formalisations have been given [16–18], in addition to Delaune, Kremer and Ryan's work. In the future, we intend to study coercion-resistance in online auction protocols.

The AS02 protocol reveals the winning bid. Bidding-price-secrecy and receipt-freeness only hold for non-winners. In [6], Chen et al. propose another auction protocol which can ensure the winner's privacy as well. We are also interested in formally verifying this protocol.

Acknowledgement. We thank Zhengqin Luo and Ben Smyth for helpful discussions and the anonymous referees for their valuable comments on a preliminary version of the paper.

References

1. Harkavy, M., Tygar, J.D., Kikuchi, H.: Electronic auctions with private bids. In: Proc. 3rd USENIX Workshop on Electronic Commerce. (1998) 6–6
2. Cachin, C.: Efficient private bidding and auctions with an oblivious third party. In: Proc. CCS'99, ACM Press (1999) 120–127
3. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: Proc. ACM-EC'99, ACM Press (1999) 129–139
4. Abe, M., Suzuki, K.: Receipt-free sealed-bid auction. In: Proc. ICISC'02. LNCS 2433., Springer (2002) 191–199
5. Lipmaa, H., Asokan, N., Niemi, V.: Secure vickrey auctions without threshold trust. In: Proc. FC'03. LNCS 2357., Springer (2003) 87–101
6. Chen, X., Lee, B., Kim, K.: Receipt-free electronic auction schemes using homomorphic encryption. In: Proc. ICISC'03. LNCS 2971., Springer (2003) 259–273
7. Lowe, G.: Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In: Proc. TACAS'96. LNCS 1055., Springer (1996) 147–166
8. Chadha, R., Kremer, S., Scedrov, A.: Formal analysis of multi-party contract signing. In: Proc. CSFW'04, IEEE CS (2004) 266–279
9. Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: Proc. POPL'01, ACM (2001) 104–115
10. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: Proc. CSFW'01, IEEE CS (2001) 82–96
11. Delaune, S., Kremer, S., Ryan, M.D.: Verifying privacy-type properties of electronic voting protocols. J. Computer Security **17**(4) (2009) 435–487
12. Kremer, S., Ryan, M.D.: Analysis of an electronic voting protocol in the applied pi calculus. In: Proc. ESOP'05. LNCS 3444., Springer (2005) 186–200
13. Jonker, H.L., Mauw, S., Pang, J.: A formal framework for quantifying voter-controlled privacy. J. Algorithms **64**(2-3) (2009) 89–105
14. Dolev, D., Yao, A.C.C.: On the security of public key protocols. IEEE Trans. Information Theory **29**(2) (1983) 198–207
15. Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. J. Log. Algebr. Program. **75**(1) (2008) 3–51
16. Backes, M., Hriţcu, C., Maffei, M.: Automated verification of remote electronic voting protocols in the applied pi-calculus. In: Proc. CSF'08, IEEE CS (2008) 195–209
17. Küsters, R., Truderung, T.: An epistemic approach to coercion-resistance for electronic voting protocols. In: Proc. S&P'09, IEEE CS (2009) 251–266
18. Küsters, R., Truderung, T., Vogt, A.: A game-based definition of coercion-resistance and its applications. In: Proc. CSF'10. IEEE CS (2010) 122-136