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

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, bidding-price-secrecy and receipt-freeness, in online auction protocols. Privacy-type 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 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. In order to protect the privacy of users, the following basic privacy-type properties are required, e.g., for the protocol proposed by Abe and Suzuki [4] :

Bidding-price-secrecy: Intuitively, a protocol preserves bidding-price-secrecy if an adversary cannot derive the bidding price of any bidder.

Receipt-freeness: Intuitively, 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. 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 formally verify whether bidding-price-secrecy and receipt-freeness hold in their protocol. We

* Naipeng Dong is supported by a grant from the Fonds National de la Recherche (Luxembourg).

model the protocol AS02 using the applied π calculus [9]. The applied π calculus provides an intuitive way to model concurrent systems, especially security protocols. Moreover, the applied π calculus is supported by a verification tool ProVerif [10], which can be used to verify some 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, formalising of privacy-type properties have already been successfully applied in the domain of voting [11] (similar ideas were developed in a different formal framework [12]). We formally verify whether bidding-price-secrecy hold for the protocol AS02 automatically using ProVerif, while receipt-freeness is verified manually. In the protocol, the two properties hold except for the winning bidding price and the winning bidder, respectively.

2 The applied π calculus

The applied π calculus is a language for modelling concurrent systems, especially security protocols. The advantages of applied π are that it provides an intuitive way to describe a protocol and cryptographic primitives can be defined by users.

Syntax. We briefly introduce the syntax of the applied π calculus (for more details, see [9]). The calculus assumes an infinite set of names (which are used to represent communication channels or other constants), an infinite set of variables and a signature Σ consisting of a finite set of function symbols, which are used to capture the cryptographic primitives. Terms are defined as names, variables, and function symbols applied to other terms. An equational theory E is defined as a set of equations on terms constructed over the signature Σ. The equivalence relation induced by E is denoted as $=_E$.

Systems are described as processes: plain processes and extended processes. Plain processes are defined similar to processes in the π calculus. Extended processes are new in variable restrictions and active substitutions. By giving restrictions to names and variables, we bound 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. It is usually generated when a process outputs a term M. 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 0 (null process, process 0 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 behavior. We say an extended process is closed if all its variables are either bounded or defined by an active substitution.

Here are other notations used later. The domain of a frame φ, denoted as $\mathsf{dom}(\varphi)$, is defined as 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 replication, a condition, an input, or an output. A context $C[\ ]$ closes A when $C[A]$ is closed.

Semantics. The operational semantics in the applied π including: structural equivalence, denoted as $\equiv$, internal reductions, denoted as $\rightarrow$, labelled reductions, denoted as $\xrightarrow{\alpha}$. Structural equivalence is defined as the smallest equivalence relation on extended processes. Intuitively, if two processes model the same thing, the two processes are structural equivalent. Internal reductions mean a process can be executed without contacting its environment, for example, two parallel processes communicate with each other, or an if- statement is evaluated, and then- or else- branch is taken. Labelled reductions are used to reason about processes that interact with the context. The transition $A \xrightarrow{\alpha} B$ means process A performs an α action and the resulting process is B. Action α is either reading a term M from the context, or outputting a name or a variable of base type to the context. Especially, when the output is a term M, $out(u, M).P$ is rewritten into $\nu x.(\{^{M}/_{x}\} \mid P)$ by using structural equivalence rules, and then use $\nu x.out(u, x)$ as the action label.

In order to represent security protocols, adversaries need to be taken into consideration. Following the Dolev-Yao model [13], an adversary has full control of the network. An adversary can eavesdrop, replay, block and inject new messages. An adversary can be modelled as an arbitrary process running in parallel with the protocol, and can interact with the protocol in order to gain information. Adversaries are represented as a context in which a process is running.

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.*

However, the definition of observational equivalence is hard to use in practice, because of the quantification over contexts. Therefore, labelled bisimilarity is introduced, which is easier to reason 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 the processes (represented by their frames), while labelled bisimilarity examines the processes' dynamic behavior. Labelled bisimilarity and observational equivalence coincide [9].

3 AS02 sealed-bid online auction protocol

Sealed-bid auctions are a type of auction in which all bidders submit their bids simultaneously 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 in 2002 [4]. This protocol involves n bidders $B_1, \ldots, B_n$ and k auctioneers $A_1, \ldots, A_k$. Before starting the protocol, a price list is published. During the protocol, each bidder commits 'yes' to the price he wants to bid for, and commits 'no' to other prices in the price list. Auctioneers work together to open the commitments of all bidders from the highest price to the lower in the price list until find the winning bid(s).

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

Before starting the protocol, one auctioneer publishes an increasing price list $p_1, \ldots, p_m$, a message M_{yes} for "I bid", a message M_{no} for "I do not bid", a large primes $p = 2q+1$, q, a generator g of order q subgroup of $\mathbb{Z}_p^*$. The protocol consists of two phases: bidding phase and opening phase.

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 as secret seeds, $r_1, \ldots, r_m$, one random number for each price, and decides a price p to bid for. Then he generates a bit-commitment for each price, using the following formula:

$$Commit_\ell = \begin{cases} g^{M_{yes}} h^{r_\ell} \ (\ell = p) & \text{Bidder bids for price } \ell \\ g^{M_{no}} h^{r_\ell} \ (\ell = p) & \text{Bidder does not bid for price } \ell \end{cases}$$

Next, he 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 $\log_g Commit_\ell$ by the interactive zero-knowledge proof. Finally, he computes t-out-of-k[1] secret shares r_ℓ^i for each secret seed r_ℓ, and sends the secret share r_ℓ^i with his signature through the one-way untappable channel to the auctioneer A_i.

Opening phase. Auctioneers together iterate the following steps for each price $\ell = m, m-1, \ldots, 1$ until the winning bid is determined.

Each auctioneer A_i publishes secret shares r_ℓ^i (the ℓ-th secret share of bidder) of all bidders. All auctioneers work together to reconstruct the secret seed r_ℓ for each bidder, and check the equivalence of

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

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

for all bidders. If there exists some bidders that the 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 exists, which means there is no bidder bids for the price ℓ, the auctioneers repeat the above process on the next lower price.

Informal reasoning of receipt-freeness. The protocol is claimed to be receipt-free, relying on the chosen cryptographic primitive, chameleon bit commitment and interactive zero-knowledge proof.

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

$$Commit_\ell = g^M \cdot h^{r_\ell}$$

Since $h = g^x$, we have

$$Commit_\ell = g^M \cdot (g^x)^{r_\ell} = g^{M+xr_\ell} \quad \text{and} \quad \log Commit_\ell = M + xr_\ell.$$

By using interactive zero-knowledge proof, a bidder B is guaranteed to know his secret key x and discrete logs $\log Commit_\ell$. An interesting property of chameleon bit commitment is that if bidder B bids for price ℓ,

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

he can easily cheat coercers by calculating a fake secret seed r'_ℓ such that:

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

Using the fake secret seed, 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 bit-commitment which he did not bid, as he did bid. Since bidders have the ability to cheat coercers, and a coercer cannot distinguish whether a bidder cheats or not, the protocol is receipt-free.

4 Modelling

We use the applied π calculus to model the AS02 protocol. In the protocol, auctioneers working together to find the winning bid, and less than t auctioneers cannot decide the winner, that guarantees t-out-of-k secrecy. In our modelling, we consider all auctioneers together as one honest auctioneer. Thus, we simplify the protocol to have only one honest auctioneer. The AS02 protocol uses interactive zero knowledge proof to guarantee each bidder knows his secret key and discrete logs of bit-commitments. However, it is not clear how a bidder and auction interact with each other. We simply zero-knowledge proof by assuming that each bidder knows his secret key and discrete logs of bit-commitments.

Signature and equational theory. Signatures and equational theory capture cryptographic primitives used in the protocol. Chameleon bit commitment is modeled as function commit with three arities (random number for that price, public key of bidder and message M to show whether bidder bids for that price).

$$\mathsf{fun\ commit}/3\ (*\text{bit commitment}*)$$

The desired property of chameleon bit commitment is captured by the following equational theory

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

M_{no} and M_{yes} are constants which represent "I do not bid" and "I bid", respectively. $\mathsf{pk}(sk_b)$ is the public key of a bidder. r is the secret seed the bidder chooses. $\mathsf{f}(r)$ is a function – given the secret seed, using function f, a bidder can calculate a fake secret seed $\mathsf{f}(r)$. Because we assume 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. We can model the function f by just give one parameter, the real secret seed.

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 the opposite situation also holds. A bidder can also open a bit-commitment as he bids for that price, but actually he does not.

In addition, we use function nextbidder to find the next bidder. The next bidder of the current bidder is defined in the equations. Function nextprice is used to find the next lower price in the price list. Function checksignature 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. Detailed equations could be found in the ProVerif code in [17].

Main process. The main process is represented in Fig. 1. This process first generates private channels: a private channel $privch_{\mathsf{b}_j}$ for each bidder B_j, a private channel $untapch_{\mathsf{b}_j}$ for each bidder, a private channel *synch* shared between bidders and the auctioneer. Note that *ch* and *winnerch* are public channels, $p_{\mathsf{b}_1}, \ldots, p_{\mathsf{b}_n}$ are parameters, each of these parameters has to be instantiated with a constant in the publish price list $\mathsf{p}_1, \ldots, \mathsf{p}_m$. Secondly, the main process generates a signature key ssk_{b_j} for each bidder, distributes the secret signature key to each bidder through private channel $privch_{\mathsf{b}_j}$, and publishes the public signature key of each secret signature key. Therefore, only each bidder knows his own secret key, and everyone including intruder knows each bidder's public signature key. Finally, the main process launches n (number of bidders) copies of bidder sub-processes and one auctioneer sub-process.

Fig. 1. The main process.

$$
\begin{aligned}
P \triangleq\ & \nu\ privch_{\mathsf{b}_1} \cdot\ \nu\ privch_{\mathsf{b}_2} \cdot\ \ldots \cdot\ \nu\ privch_{\mathsf{b}_n} \cdot \\
& \nu\ untapch_{\mathsf{b}_1} \cdot\ \nu\ untapch_{\mathsf{b}_2} \cdot\ \ldots \cdot\ \nu\ untapch_{\mathsf{b}_n} \cdot \\
& \nu\ synch \cdot \\
& \nu\ ssk_{\mathsf{b}_1} \cdot\ \nu\ ssk_{\mathsf{b}_2} \cdot\ \ldots \cdot\ \nu\ ssk_{\mathsf{b}_n} \cdot \\
& \mathtt{let}\ spk_{\mathsf{b}_1} = \mathsf{pk}(ssk_{\mathsf{b}_1})\ \mathtt{in} \\
& \ldots \\
& \mathtt{let}\ spk_{\mathsf{b}_n} = \mathsf{pk}(ssk_{\mathsf{b}_n})\ \mathtt{in} \\
& (\mathtt{out}(privch_{\mathsf{b}_1}, ssk_{\mathsf{b}_1}) \mid \ldots \mid \mathtt{out}(privch_{\mathsf{b}_n}, ssk_{\mathsf{b}_n}) \mid \\
& \ \mathtt{out}(ch, spk_{\mathsf{b}_1}) \mid \ldots \mid \mathtt{out}(ch, spk_{\mathsf{b}_n}) \mid \\
& \ \mathtt{let}\ p_b = p_{\mathsf{b}_1}\ \mathtt{in\ let}\ untapch = untapch_{\mathsf{b}_1}\ \mathtt{in} \\
& \ \mathtt{let}\ privch = privch_{\mathsf{b}_1}\ \mathtt{in}\ P_B \mid \\
& \ \ldots \mid \\
& \ \mathtt{let}\ p_b = p_{\mathsf{b}_n}\ \mathtt{in\ let}\ untapch = untapch_{\mathsf{b}_n}\ \mathtt{in} \\
& \ \mathtt{let}\ privch = privch_{\mathsf{b}_n}\ \mathtt{in}\ P_B \mid P_A)
\end{aligned}
$$

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

Fig. 2. The bidder process.

Bidder process. Firstly, a bidder receives his secret signature key from his private channel. Secondly, the bidder generates his secret key sk_b, and chooses a series of random numbers $r_1 \ldots r_m$ as secret seeds. Thirdly, the bidder 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 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 process for a bidder is given in Fig. 2. The keyword *sync* represents a sub-process in which a bidder sends a token to the auctioneer and waits the auctioneer's response to synchronise with other bidders. Therefore, *sync* represents the end of the bidder phase.

Auctioneer process. During the bidding phase, the auctioneer launches n copies of sub-process *readinfo* to gather information of each bidder B_j, including series of bit-commitments $cmt^{\mathsf{p}_1}_{\mathsf{b}_j}, \ldots, cmt^{\mathsf{p}_m}_{\mathsf{b}_j}$, secret seeds $ss^{\mathsf{p}_1}_{\mathsf{b}_j}, \ldots, ss^{\mathsf{p}_m}_{\mathsf{b}_j}$, public signa-

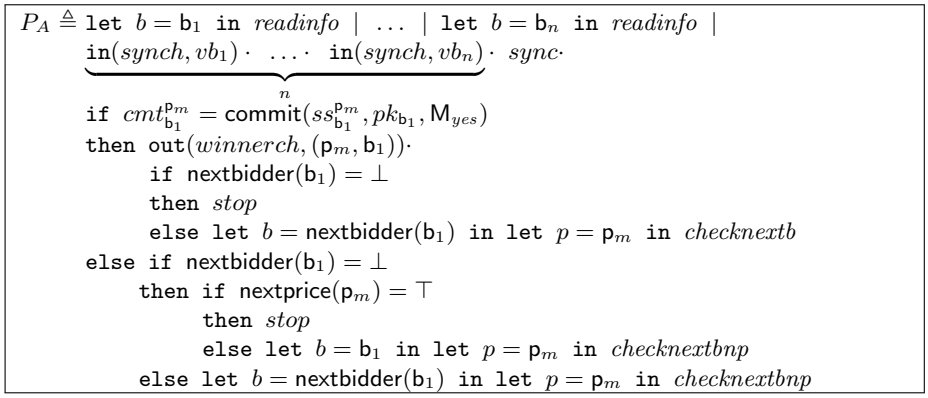

Fig. 3. The auctioneer process.

ture key spk_{b_j}, and public key pk_{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 $cmt^{\mathsf{p}_m}_{\mathsf{b}_j} = \mathsf{commit}(ss^{\mathsf{p}_m}_{\mathsf{b}_j}, pk_{\mathsf{b}_j}, \mathsf{M}_{yes})$ for each bidder. If the two values are equivalent, it means bidder B_j bids for that price, otherwise, bidder B_j does not bid for that price. If the auctioneer finds some winners, he publishes the wining bid, and stops the process. Otherwise, the auctioneer repeats the evaluation steps for each bidder at the next lower price. The sub-process *checknextb* is used to check the evaluation of a bidder b at price p, if there are already some winners before bidder b. The sub-process *checknextbnp* is used to check the evaluation of a bidder b at price p, if there is no winner before bidder b. We use $\bot$ 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 gives whether the protocol satisfies the security property as output. There are three possible outputs: ProVerif proves the required property of the protocol, ProVerif finds an attack as a counter example, ProVerif can neither prove the property nor disprove the property.

In ProVerif, the 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, the strong secrecy is defined as: for all closed substitutions σ and σ' of free variables in a process P, the process satisfies $P_\sigma \approx P_{\sigma'}$ ($\approx$ denotes observational equivalence). To check strong secrecy of variables $x_1, \ldots, x_n$, we can use the query: *noninterf* $x_1, \ldots, x_n$.
Intuitively, by replacing $x_1, \ldots, x_n$ 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. 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. 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.

5.2 Bidding-price-secrecy

Bidding-price-secrecy guarantees the anonymity of the link between a bidder to the price he bids for. According to the AS02 protocol, the winning bid is published, therefore, bidding-price-secrecy for the winning bidder is not satisfied. In the later sections, we refer to bidding-price-secrecy with respect to non-winning bids. There are two levels of secrecy: standard bidding-price-secrecy and strong bidding-price-secrecy.

Standard bidding-price-secrecy. Intuitively, standard bidding-price-secrecy means no matter how an adversary interacts with the protocol, an adversary cannot derive the bidding price of a non-winning bidder. Since all the prices are published in the price list, standard bidding-price-secrecy is defined as an adversary cannot derive the link between a non-winning bidder and the price he bids. In order to show that an adversary cannot derive the bidding price of a non-winning bidder, we can use 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 representing the price the bidder bids for. The variable p can be any price in the price list, but cannot be the highest price. By inquiring *not attacker* : p, we check whether an adversary can derive the bidding price of a non-winning bidder. ProVerif gives us positive result, 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 a bidder bids for price a and the case for price c. In other words, if a bidder swaps his bidding price a to c, an adversary cannot tell the difference. We use an observational equivalence property in applied π 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 process V_A votes for a and process 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 intruder cannot deduce the votes of V_A and V_B, given V_A and V_B as a counterbalance to each other. Different from privacy in voting, in AS02 protocol, even all other bidders reveal how they bid, a bidder's bidding price is not revealed. Therefore, we do not need a counterbalance 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 2 (Strong bidding-price-secrecy). *An auction protocol is strong bidding-price-secret if for all bidders except the winning bidder satisfy:*

$$S[B_A\{^a/_{p_b}\} \mid B_B\{^d/_{p_b}\} \mid A] \approx_\ell S[B_A\{^c/_{p_b}\} \mid B_B\{^d/_{p_b}\} \mid A]$$

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

S is defined as a context $S \triangleq \nu\tilde{n} \cdot (P_A\sigma_1 \mid \ldots \mid P_A\sigma_n \mid \_)$. The context is as an auction process with a hole instead of two bidder processes and the auctioneer process. In the context, $P_A\sigma_i$ are bidder processes, and $\tilde{n}$ are channel names. A is the auctioneer process. B_A is a non-winning bidder process. B_B is a bidder process in which a bidder bids for a higher price d. The intuition is that an adversary cannot distinguish between a non-winning bidder bids for price a or price c, given another bidder bidding for a higher price d.

In order to make it possible to check in ProVerif, we need to do some modifications to the way the auctioneer process is presented. Because ProVerif is sensitive with evaluations of statements in the if- then- else- constructs. Keeping these evaluations, ProVerif reports false attacks, more detailed reasons could be found in paper[10]. Therefore, in the ProVerif code, we do want the if- then- else- constructs. Fortunately, we can simplify the process by cutting the if- then- else- part of process, under our assumptions in the definition of strong bidding-price-secrecy. Since there is a higher bid in the equivalence, the auctioneer process will stop after checking the higher price, the remain process will not be executed, therefore we can cut the remain process without affecting the checking result.

In order to be able to check *noninterf* in ProVerif, we modify the bidder process by replacing if- then- else- instructions with choice[] instructions. Detailed reasons of the modification could be found in []. ProVerif code could be found on [17]. By requiring *noninterf* p_b *among* $\mathsf{p}_1, \ldots, \mathsf{p}_{d-1}$, we replace variable p_b with p_1 to p_{d-1} (expect the price p_d), and get $d-1$ different versions of the process. ProVerif giving a positive result means each pair of two versions are observational equivalent, meaning a bidder swapping his bidding price is indistinguishable to an adversary. In this way, we prove that the protocol satisfies strong bidding-price-secrecy.

5.3 Receipt-freeness

Receipt-freeness is defined as a bidder cannot prove to an adversary that he bids in a certain way. Receipt-freeness is a stronger privacy-type property than bidding-price-secrecy [11]. It is useful to protect bidders from coercers. 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.

Receipt-freeness in voting is formalised as an observational equivalence [11]. A protocol satisfies receipt-freeness if there exists a process V', in which a voter votes for a candidate a but communicates with the adversary in order to feign cooperation with him by telling the adversary he votes for the different candidate c the adversary wants him to vote. By proving the observational equivalence, 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. In order to model the observational equivalence, the situation that a voter is willing to provide his secret information to the adversary needs to be modelled first:

Definition 3 (Process P^{ch} [11]). *Let P be a plain process and ch a channel name. P^{ch} 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,*
- $(\mathsf{out}(u, M).P)^{ch} \hat{=} \mathsf{out}(u, M).P^{ch}$,
- $(!P)^{ch} \hat{=} !P^{ch}$,
- $(\mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q)^{ch} \hat{=} \mathit{if}\ M = N\ \mathit{then}\ P^{ch}\ \mathit{else}\ Q^{ch}$.

Delaune *et al.* also define process transformation $A^{\setminus out(ch,\cdot)}$, which is the process A hiding the output on the public channel ch.

Definition 4 (Process $A^{\setminus out(ch,\cdot)}$ [11]). *Let A be an extended process. We define the process $A^{\setminus out(ch,\cdot)}$) as $\nu ch.(A|!in(ch, x))$.*

In 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 in AS02 auction protocol. We need a winning bidder process B_B bids for a higher price d in modelling receipt-free as well, 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 5 (Receipt-freeness). *An auction protocol is receipt-free if there exists a closed plain process B' such that*

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

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

The context S' is slightly different with the context defined for the strong bidding-price-secrecy, as follows: $S' \triangleq \nu\tilde{n} \cdot (P_A\sigma_1 \mid \ldots \mid P_A\sigma_n \mid \_ \mid A)$. The context is as an auction process with a hole instead of two bidder processes. 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 the highest price and the winning bid. Due to the existence of higher bid ($B_B\{^d/_{p_b}\}$) on both of the left and right sides of the equivalence, the information the auctioneer process reveals is the same on both sides of the equivalence. The auctioneer process only reads information in bidding phase, and it processes the same on opening phase. The auctioneer process has no influence to the equivalence. Therefore, we can add the auctioneer process in the context S'. B' is a process in which bidder B_A bids for price a but communicates with the adversary and tells the adversary he bids for price c. The first equivalence says that ignoring the outputs B' makes on the adversary channel chc, B' looks like a normal process B_A bidding for price a. The second equivalence says that the adversary cannot tell the difference between a situation in which B_A obeys his order and bids for price c and the situation in which he pretends to cooperate but actually bids for price a, provided there is some bidding process B_B bids higher, such that bidding process B_A is not winner.

Fig. 4. The process $B_A\{^c/_{p_b}\}$.

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

Fig. 5. The process $B'^{\setminus out(chc,\cdot)}$.

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

Fig. 6. The process $B_A\{^a/_{p_b}\}$.

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

Fig. 7. The process $B_A\{^a/_{p_b}\}^{chc}$.

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

Fig. 8. The process $B_B\{^d/_{p_b}\}$.

$$
\begin{aligned}
B_B\{^d/_{p_b}\} \triangleq\ & \mathtt{in}(privch_b, bssk_b) \cdot \\
& \nu\ bsk_b \cdot \\
& \mathtt{out}(ch, \mathsf{sign}(\mathsf{pk}(bsk_b), bssk_b)) \cdot \\
& \nu\ br_1 \cdot \ \ldots \cdot \nu\ br_a \cdot \ \ldots \cdot\ \nu\ br_c \cdot \ \ldots \cdot\ \nu\ br_d\ \cdot \ \ldots \cdot\ \nu\ br_m \cdot \\
& \mathtt{let}\ \ bcmt^{\mathsf{p}_1} = \mathsf{commit}(br_1, \mathsf{pk}(bsk_b), \mathsf{M}_{no})\ \ \mathtt{in} \\
& \ldots \\
& \mathtt{let}\ \ bcmt^{\mathsf{p}_a} = \mathsf{commit}(br_a, \mathsf{pk}(bsk_b), \mathsf{M}_{no})\ \ \mathtt{in} \\
& \ldots \\
& \mathtt{let}\ \ bcmt^{\mathsf{p}_c} = \mathsf{commit}(br_c, \mathsf{pk}(bsk_b), \mathsf{M}_{no})\ \ \mathtt{in} \\
& \ldots \\
& \mathtt{let}\ \ bcmt^{\mathsf{p}_d} = \mathsf{commit}(br_d, \mathsf{pk}(bsk_d), \mathsf{M}_{yes})\ \ \mathtt{in} \\
& \ldots \\
& \mathtt{let}\ \ bcmt^{\mathsf{p}_m} = \mathsf{commit}(br_m, \mathsf{pk}(bsk_b), \mathsf{M}_{no})\ \ \mathtt{in} \\
& \mathtt{out}(ch, \mathsf{sign}((bcmt^{\mathsf{p}_1}, \ldots, bcmt^{\mathsf{p}_m}), bssk_b)) \cdot \\
& \mathtt{out}(untapch, (br_1, \ldots, br_a, \ldots, br_c, \ldots, br_d, \ldots, br_m)) \cdot sync \cdot
\end{aligned}
$$

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

Fig. 9. The process B'.

To prove receipt-freeness of the protocol, we need to find a process B' which satisfies two equivalence in the definition of receipt-freeness. The idea is that a bidder bids for price c but tells the adversary that he bids for price a. According to the properties of chameleon bit commitment, the bidder can send a sequence of fake secret seeds to the adversary, which is used to open the bit-commitments as bidding for price c, and sends the series of real secret seeds to the auctioneer through an untapped channel, which is used to open the bit-commitments as bidding for price a, while the bit-commitments the bidder publishes are unchanged. The adversary opens the bit-commitments as the bidder bids for price c, while the auctioneer opens the bit-commitments as the bidder bids for price a. The sequence of secret seeds sent the adversary is $r_1, \ldots, \mathsf{f}(r_a), \ldots, \mathsf{f}(r_c), \ldots, r_m$, while the sequence of secret seeds sent to the auctioneer is $r_1, \ldots, r_a, \ldots, r_c, \ldots, r_m$. The process B' is represented in Fig. 9. The bidder in this process communicates with the adversary through channel chc, sends the adversary his secret signature key ssk_b, his secret key sk_b. The bidder sends the adversary the secret seeds $r_1, \ldots, \mathsf{f}(r_a), \ldots, \mathsf{f}(r_c), \ldots, r_m$, while sending the auctioneer the secret seeds $r_1, \ldots, r_a, \ldots, r_c, \ldots, r_m$ through an untappable channel. The untappable channel ensures the adversary knows nothing about the differences.

To prove the first equivalence, we can simply consider that $B'^{\backslash out(chc,\cdot)}$ as process B' without communication on chc channel. The process $B'^{\backslash out(chc,\cdot)}$ can be considered as a process $B_A\{^c/_p\}$. Since the process $B'^{\backslash out(chc,\cdot)}$ is exactly the same as the process $B_A\{^c/_p\}$, the first equivalence is satisfied. To show the second equivalence, we can informally consider all the executions of each side. Let $P = S'[B_A\{^a/_{p_b}\}^{chc} \mid B_B\{^d/_{p_b}\}]$, let $Q = S'[B' \mid B_B\{^d/_{p_b}\}]$. We denote sequence

of names $sk_b, r_1, \ldots, r_m, bsk_b, br_1, \ldots, br_m$, by $\tilde{n}$.

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

$$
\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)} 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(chc,x_4)} \\
& \xrightarrow{\nu\ x_5 \cdot\ out(ch,x_5)} \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}(sk_b),ssk_b)}/_{x_4}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_5}\}) \\
& \xrightarrow{\nu\ x_6 \cdot\ out(chc,x_6)} \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}(sk_b),ssk_b)}/_{x_4}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_5}\} \\
& \qquad\qquad \mid \{^{r_1,\ldots,\mathsf{f}(r_a),\ldots,\mathsf{f}(r_c),\ldots,r_m}/_{x_6}\}) \\
& \xrightarrow{\nu\ x_7 \cdot\ out(ch,x_7)} \xrightarrow{\nu\ x_8 \cdot\ out(chc,x_8)} \\
& \xrightarrow{\nu\ x_9 \cdot\ out(ch,x_9)} \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}(sk_b),ssk_b)}/_{x_4}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_5}\} \\
& \qquad\qquad \mid \{^{r_1,\ldots,\mathsf{f}(r_a),\ldots,\mathsf{f}(r_c),\ldots,r_m}/_{x_6}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_7}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_8}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}((bcmt^{\mathsf{p}_1},\ldots,bcmt^{\mathsf{p}_m}),bssk_b)}/_{x_9}\}) \\
& \xrightarrow{\nu\ x_{10} \cdot\ out(ch,x_{10})} \nu\tilde{n} \cdot\ (Q_6 \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}(sk_b),ssk_b)}/_{x_4}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_5}\} \\
& \qquad\qquad \mid \{^{r_1,\ldots,\mathsf{f}(r_a),\ldots,\mathsf{f}(r_c),\ldots,r_m}/_{x_6}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_7}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_8}\} \\
& \qquad\qquad \mid \{^{\mathsf{sign}((bcmt^{\mathsf{p}_1},\ldots,bcmt^{\mathsf{p}_m}),bssk_b)}/_{x_9}\} \\
& \qquad\qquad \mid \{^{r_1,\ldots,\mathsf{f}(r_a),\ldots,\mathsf{f}(r_c),\ldots,r_m}/_{x_{10}}\})
\end{aligned}
$$

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 bit-commitments.

The adversary first checks whether the bidder is lying, by checking whether x_3 is equivalent to x_4 and whether x_7 is equivalent to x_8. The only functions an adversary can use are getmsg and open. By applying these two destructor functions, an adversary can get another two terms, the public key of the bidder and the series of opened messages, represented as x_{11} and x_{12}, respectively.

$$x_{11} = \mathsf{getmsg}(x_3, x_1) \qquad x_{12} = \mathsf{open}(x_7, x_6, x_{11})$$

The frame obtained from process P is:

$$\begin{aligned}\varphi = \nu\tilde{n}\cdot\ (&\{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_3}\}\\ &\mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_4}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_5}\}\\ &\mid \{^{r_1,\ldots,r_m}/_{x_6}\} \mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_7}\}\\ &\mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_8}\}\\ &\mid \{^{\mathsf{sign}((bcmt^{\mathsf{p}_1},\ldots,bcmt^{\mathsf{p}_m}),bssk_b)}/_{x_9}\} \mid \{^{r_1,\ldots,r_m}/_{x_{10}}\}\\ &\mid \{^{\mathsf{getmsg}(x_3,x_1)}/_{x_{11}}\} \mid \{^{\mathsf{open}(x_7,x_6,x_{11})}/_{x_{12}}\})\end{aligned}$$

The frame obtained from process Q is:

$$\begin{aligned}\varphi' = \nu\tilde{n}\cdot\ (&\{^{ssk_b}/_{x_1}\} \mid \{^{sk_b}/_{x_2}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_3}\}\\ &\mid \{^{\mathsf{sign}(\mathsf{pk}(sk_b),ssk_b)}/_{x_4}\} \mid \{^{\mathsf{sign}(\mathsf{pk}(bsk_b),bssk_b)}/_{x_5}\}\\ &\mid \{^{r_1,\ldots,\mathsf{f}(r_a),\ldots,\mathsf{f}(r_c),\ldots,r_m}/_{x_6}\}\\ &\mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_7}\}\\ &\mid \{^{\mathsf{sign}((cmt^{\mathsf{p}_1},\ldots,cmt^{\mathsf{p}_m}),ssk_b)}/_{x_8}\}\\ &\mid \{^{\mathsf{sign}((bcmt^{\mathsf{p}_1},\ldots,bcmt^{\mathsf{p}_m}),bssk_b)}/_{x_9}\}\\ &\mid \{^{r_1,\ldots,\mathsf{f}(r_a),\ldots,\mathsf{f}(r_c),\ldots,r_m}/_{x_{10}}\})\\ &\mid \{^{\mathsf{getmsg}(x_3,x_1)}/_{x_{11}}\} \mid \{^{\mathsf{open}(x_7,x_6,x_{11})}/_{x_{12}}\})\end{aligned}$$

The process B_A bids for price a. The adversary opens the bit-commitments $cmt^{p_a} = \mathsf{commit}(r_a, \mathsf{pk}(sk_b), \mathsf{M}_{yes})$ and $cmt^{p_c} = \mathsf{commit}(r_c, \mathsf{pk}(sk_b), \mathsf{M}_{no})$ as:

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

On the process Q, the process B_A 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, the adversary open the bit-commitments $cmt^{p_a} = \mathsf{commit}(r_a, \mathsf{pk}(sk_b), \mathsf{M}_{no}) = \mathsf{commit}(\mathsf{f}(r_a), \mathsf{pk}(sk_b), \mathsf{M}_{yes})$ 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:

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

The frame φ and the frame φ' are statically equivalent. Other secret seeds are the same in both P and Q. The adversary gets the same opening results on both P and Q. The frames we obtained are statically equivalent for every step. And the process P and process Q can both do the same action at every sate. Therefore, the process P and Q satisfies the definition of labelled bisimilarity. Now, we 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 the definition of privacy and receipt-freeness in paper[11]. We have modelled the AS02 protocol in the applied π calculus, formalised two privacy-type properties: bidding-price-secrecy and receipt-freeness, and verified that the AS02 protocol satisfies the two properties. Bidding-price-secrecy is considered at two levels: standard bidding-price-secrecy and strong bidding-price-secrecy. Standard bidding-price-secrecy is modelled as a non-winning bidder's bidding price is not in an adversary's knowledge. It is verified automatically using ProVerif. Strong bidding-price-secrecy is modelled as an observational equivalence, which is also verified automatically in ProVerif. Receipt-freeness is modelled as observational equivalences as well and verified manually.

Coercion-resistance is a stronger privacy property, saying 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 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 coercee can do as he wants, in spite of the coercer, and the coercer cannot distinguish whether the coercee 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 zero-knowledge proof ensures the bidder knows his own secret key and the discrete logs of bit-commitments, a bidder can figure out which price the coercer wants him to bid, and then calculate the fake secrete seeds $\mathsf{f}(r_a)$ and $\mathsf{f}(r_c)$ to change the price the coercer calculated, and sends secret seeds $r_1, \ldots, \mathsf{f}(r_a), \ldots, \mathsf{f}(r_c), \ldots r_m$ to the auctioneer.

Coercion-resistance is a complicated property to formalise. Several different formalisations have been given [14–16], in addition to Delaune, Kremer and Ryan's work [11]. In the future, we would like to study coercion-resistance in online auction protocols. Moreover, the winning bid is revealed in the AS02 protocol. Bidding-price-secrecy and receipt-freeness are only for non-winners. In [6], Chen et al. propose another auction protocol which can ensure the winner's privacy also. We are also interested in formally verifying this protocol.

Acknowledgement. We want to thank Zhengqin Luo 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. Jonker, H.L., Mauw, S., Pang, J.: A formal framework for quantifying voter-controlled privacy. J. Algorithms **64**(2-3) (2009) 89–105
13. Dolev, D., Yao, A.C.C.: On the security of public key protocols. IEEE Trans. Information Theory **29**(2) (1983) 198–207
14. 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
15. Küsters, R., Truderung, T.: An epistemic approach to coercion-resistance for electronic voting protocols. In: Proc. S&P'09, IEEE CS (2009) 251–266
16. Küsters, R., Truderung, T., Vogt, A.: A game-based definition of coercion-resistance and its applications. In: Proc. CSF'10. IEEE CS (2010) to appear
17. Dong, N.: http://satoss.uni.lu/members/naipeng/