Formal Modelling and Analysis of Receipt-Free Auction Protocols in Applied Pi

Naipeng Dong[a,*], Hugo Jonker[b], Jun Pang[c]

[a]*School of Computing, National University of Singapore, 21 Lower Kent Ridge Rd, 119077, Singapore*
[b]*School of Computer Science, Open University of the Netherlands, Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands*
[c]*Faculty of Science, Technology and Communication & Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, 6 rue Richard Coudenhove-Kalergi, L-1359, Luxembourg*

Abstract

We formally study two privacy-type properties for e-auction protocols: bidding-price-secrecy and receipt-freeness. These properties are formalised as observational equivalences in the applied pi calculus. We analyse two receipt-free auction protocols: one proposed by Abe and Suzuki in 2002 (AS02) and the other by Howlader et al. in 2014 (HRM14). Bidding-price-secrecy of the AS02 protocol is verified using the automatic verifier ProVerif, whereas receipt-freeness of the two protocols, as well as bidding-price-secrecy of the HRM14 protocol, are proved manually.

Keywords: e-auction, security protocol, formal verification, bidding-price-secrecy, receipt-freeness

1. Introduction

Auctions are ways to negotiate exchange of goods and services. We use *e-auctions* to refer to auctions over the Internet. A typical (e-)auction works as follows: a seller offers items to bid, then bidders submit bids, finally auctioneers decide the winner. In a traditional auction, bidders attend the auction in person. Compared to the traditional auctions, e-auctions attract more participants, as users with the Internet can join an auction. Real-life examples are well-known websites like *eBay, eBid, Yahoo!auctions*

*Corresponding author
Email addresses: dcsdn@nus.edu.sg (Naipeng Dong), hugo.jonker@ou.nl (Hugo Jonker), jun.pang@uni.lu (Jun Pang)

and so on. E-auction protocols are also the subject of an active field of research [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].

There are different types of (e-)auctions. For instance, depending on whether the bids are public, there are sealed-bid auctions and open-bid auctions;

- *Sealed-bid auctions*: There are two phases in an auction: the bidding phase and the opening phase. Bidders can only submit bids in the bidding phase. All bids are sealed in the bidding phase and opened in the opening phase.
- *Open-bid auctions*: Bids are broadcast to all participants.

Other criteria to classify (e-)auctions exist as well. For example, depending on the bidding price increases or decreases, there are English auctions (a bid needs to be higher than the previous one; the winning bid is the final bid) and Dutch auctions (the bidding price decreases until a bid is submitted); depending on the calculation of payment, there are first-price auctions (the winner pays for the price he bid (highest price)) and Vickrey auctions (the winner pays for the second highest price). Different auctions are suitable for different types of negotiations, e.g., English auctions are often used in real estate, Dutch auctions are often used in flower selling, and Vickrey auctions are favoured by economists as they are better at encouraging bidders to express their real estimation on the value of the items to bid on [11].

Many security issues have been identified in e-auctions, such as, a bidder may falsely claim or forge bids, the auctioneer may corrupt with other bidders [12]. Beside security issues, an important problem with existing e-auction systems is privacy. The link between a bidder and his bids needs to be protected as such information can be used to target a bidder with unsolicited junk mails or other malicious purposes, e.g., *bid shielding*.[1] A major challenge of designing a protocol is to ensure the functionality of the protocol. In addition to that, a challenge for designing a privacy preserving e-auction protocol is that too much anonymity may allow bidders to repudiate bids, whereas insufficient anonymity allows bidders to be profiled.

[1] A dishonest bidder submits a higher price to deter other bidders with lower valuations, when it approaches the close time of the auction, the dishonest bidder withdraws his bid in order to win with another lower bid from him.

Depending on different types of auctions, privacy may have varying levels. For instance, in sealed-bid auctions, all bids are sealed until the winner is determined. Therefore, if auctioneers can decide the winners without knowing the non-winning bidder's bids, sealed-bid auctions can offer bidding-price secrecy for non-winning bidders; while in open-bid auctions, all the bids are published. Some auctions require that the auctioneer cannot link a bidder to his bids, whereas some others do not. The arguments for this requirement are made according to the following lines. In Vickery auctions, a bidder's bid reflects the bidder's valuation of the item bid on. Knowing a bidder's bid, an auctioneer knows the bidder's valuation. Since the winning bidder pays for the second highest price, the auctioneer could enter a bid just slightly lower than the bidder's valuation, to increase the auction's revenue [11]. Contrarily in English auctions, a bidder's previous bids reveal less information of the bidder's future bid, thus, that the auctioneer knows the link between a bidder and his previous bids is less harmful [11]. In general, sealed-bid e-auctions require that the non-winning bidders' bidder-bid relation should be kept secret.

In addition to the above privacy notions, a stronger privacy notion – enforced privacy – has also been identified. In sealed-bid e-auctions, a bidder may be coerced to bid a low price, so that the coercer can win an auction with an unreasonably low price. The phenomenon that a coercer tries to control the winning price by coercion is called *bid-rigging*. Note that the traditional auctions do not suffer from bid-rigging, as the bidders do not have receipts on submitting a bid [13]. Inspired by the requirement of receipt-freeness in e-voting that a voter should not be able to prove his vote to a voter-buyer, the requirement of receipt-freeness for fighting against bid-rigging has been identified [14].

In general, the following two privacy notions are required in sealed-bid e-auctions:

Bidding-price-secrecy: A sealed-bid e-auction protocol preserves bidding-price-secrecy for non-winning bidders if the adversary cannot determine the bidding price of any non-winning bidder.

Receipt-freeness: A sealed-bid e-auction protocol is receipt-free for non-winning bidders if a non-winning bidder cannot prove how he bids to the adversary.

In this paper, we first formalise these two privacy notions in the applied pi calculus (Section 4). Without a precise definition, many protocols claimed to satisfy a property were later found flawed (see examples in [15]). For example, the Okamoto e-voting protocol [16] which claimed to satisfy receipt-freeness expressed in natural language, was later shown flawed with respect to a rigorous definition [17]; and according to the author, one important reason is the lack of formal definition of receipt-freeness in e-voting. To validate our formalisation, we model and study privacy properties of the AS02 protocol proposed by Abe and Suzuki [4] (Section 5) and the HRM14 protocol proposed by Howlader et al. [18] (Section 6). The authors of both papers 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 [19, 20, 15, 21]. 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 protocols. We model both protocols using the applied pi calculus [22] (Section 2). The applied pi calculus provides an intuitive way to model concurrent systems, especially security protocols. Moreover, it is supported by ProVerif [23], a verification tool which can be used to verify a number of security properties automatically (Section 3). As suggested in [15], we use observational equivalence to express bidding-price-secrecy and receipt-freeness in the applied pi calculus. Previously, formalisation of privacy-type properties has already been successfully executed in the domain of voting [24, 15] (similar ideas were developed in a different formal framework [25]). Bidding-price-secrecy for the AS02 protocol is verified automatically using ProVerif, whereas receipt-freeness, as well as bidding-price-secrecy for the HRM14, are proven manually. Related work is discussed in Section 7 and Section 8 concludes the paper with a few future works.

Note that an extended abstract of our work has appeared in the proceedings of 7th International Workshop on Formal Aspects in Security and Trust [26], where we have formally analysed the AS02 protocol. In the current paper, we have included the

full details of our analysis the AS02 protocol, and extended our method to analyse the recently published HRM14 protocol. For the HRM14 protocol, we showed that it may not satisfy receipt-freeness and proposed a fix, and then we proved that the fixed protocol satisfies receipt-freeness.

2. The applied pi calculus

The applied pi calculus is a language for modelling and analysing concurrent systems, in particular cryptographic protocols. It assumes the Dolev-Yao model [27] for adversaries which have full control of the network. Namely, an adversary can eavesdrop, replay, block and inject messages. The 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.

The following briefly introduces its syntax, semantics and equivalence relations. It is mainly based on [22, 28].

2.1. Syntax

The calculus assumes an infinite set of *names* (which are used to model communication channels or other atomic data), an infinite set of *variables* (which are used to model received messages) and a signature Σ consisting of a finite set of *function symbols* (which are used to model cryptographic primitives). Each function symbol has an arity. A function symbol with arity zero is a constant.

Example 1. *In cryptographic protocols, typical function symbols are* enc *with arity 2 for encryption and* dec *with arity 2 for decryption.*

Terms (which are used to model messages) are defined as names, variables, or function symbols applied to terms (see Figure 1).

$M,\ N,\ T ::=$	terms
$\mathtt{a},\ \mathtt{b},\ \mathtt{m},\ \mathtt{n}, \ldots$	names
$x,\ y,\ z$	variables
$\mathsf{f}(M_1, \ldots, M_\ell)$	function application

Figure 1: Terms in the applied pi calculus.

The applied pi calculus assumes a sort system for terms. Terms can be of a base type (e.g., KEY or a universal base type DATA) or type $\mathsf{Channel}\langle\omega\rangle$ where ω is a type. A variable and a name can have any type. A function symbol can only be applied to and return, terms of base type. Terms are assumed to be well-sorted and substitutions preserve types.

Terms are often equipped with an equational theory E – a set of equations on terms. The equational theory is normally used to capture features of cryptographic primitives. The equivalence relation induced by E is denoted as $=_E$.

Example 2. *The behaviour of symmetrical encryption and decryption can be captured by the following equation:* $\mathsf{dec}(\mathsf{enc}(x,y),y) =_E x$*, where* x,y *are variables.*

Systems are described as processes: plain processes and extended processes (see Figure 2). In Figure 2, M and N are terms, $\mathtt{n}$ is a name, x is a variable and u is a

$P,\ Q,\ R ::=$	plain processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu\, \mathtt{n}.\, P$	name restriction
if $M =_E N$ then P else Q	conditional
$\mathsf{in}(u,x).\, P$	message input
$\mathsf{out}(u,M).\, P$	message output
$A,\ B,\ C ::=$	extended processes
P	plain process
$A \mid B$	parallel composition
$\nu\, \mathtt{n}.\, A$	name restriction
$\nu\, x.\, A$	variable restriction
$\{M/x\}$	active substitution

Figure 2: Processes in the applied pi calculus.

metavariable, standing either for a name or a variable. The null process 0 does nothing. The parallel composition $P \mid Q$ represents the sub-process P and the sub-process Q running in parallel. The replication $!P$ represents an infinite number of process P running in parallel. The name restriction $\nu\, \mathtt{n}.\, P$ binds the name $\mathtt{n}$ in the process P, which means the name $\mathtt{n}$ is secret to the adversary. The conditional evaluation $M =_E N$ represents equality over the equational theory rather than strict syntactic identity. The

message input $\mathsf{in}(u,x)$. P reads a message from channel u, and bounds the message to the variable x in the following process P. The message output $\mathsf{out}(u,M)$. P sends the message M on the channel u, and then runs the process P. Extended processes add variable restrictions and active substitutions. The variable restriction $\nu\, x.\, A$ bounds the variable x in the process A. The active substitution $\{M/x\}$ replaces variable x with term M in any process that it contacts with. We also write "let $x = m$ in P" to represent $P\{M/x\}$.

Names and variables have scopes. A name is *bound* if it is under restriction. A variable is *bound* by restrictions or inputs. Names and variables are *free* if they are not delimited by restrictions or by inputs. The sets of free names, free variables, bound names and bound variables of a process A are denoted as $\mathsf{fn}(A)$, $\mathsf{fv}(A)$, $\mathsf{bn}(A)$ and $\mathsf{bv}(A)$, respectively. A term is *ground* when it does not contain variables. A process is *closed* if it does not contain free variables. A *frame* is defined as an extended process built up from 0 and active substitutions by parallel composition and restrictions. The active substitutions in extended processes allow us to map an extended process A to its frame $\mathsf{frame}(A)$ by replacing every plain process in A with 0. The *domain* of a frame B, denoted as $\mathsf{domain}(B)$, is the set of variables for which the frame defines a substitution. A *context* $\mathscr{C}[\_]$ is defined as a process with a hole, which may be filled with any process. An evaluation context is a context whose hole is not under a replication, a condition, an input or an output. Finally, we abbreviate the process $\nu\, \mathtt{n}_1.\cdots\nu\, \mathtt{n}_n.\, P$ as $\nu\, \tilde{\mathtt{n}}.\, P$.

2.2. Operational semantics

The operational semantics of the applied pi calculus is defined by: 1) structural equivalence ($\equiv$), 2) internal reduction ($\rightarrow$), and 3) labelled reduction ($\xrightarrow{\alpha}$) of processes.

1) Informally, two processes are structurally equivalent if they model the same thing but differ in structure. Formally, structural equivalence of processes is the smallest equivalence relation on extended process that is closed by α-conversion on names and variables, by application of evaluation contexts as shown in Figure 3.

2) Internal reduction is the smallest relation on extended processes closed under structural equivalence, application of evaluation of contexts as shown in Figure 4.

3) The labelled reduction models the environment interacting with the processes. It

$$
\begin{array}{lrcll}
\texttt{PAR}-0 & A \mid 0 & \equiv & A & \\
\texttt{PAR}-\texttt{A} & A \mid (B \mid C) & \equiv & (A \mid B) \mid C & \\
\texttt{PAR}-\texttt{C} & A \mid B & \equiv & B \mid A & \\
\texttt{REPL} & !P & \equiv & P \mid !P & \\
\texttt{SUBST} & \{M/x\} \mid A & \equiv & \{M/x\} \mid A\{M/x\} & \\
\texttt{NEW}-0 & \nu\, u.\, 0 & \equiv & 0 & \\
\texttt{NEW}-\texttt{C} & \nu\, u.\, \nu\, v.\, A & \equiv & \nu\, v.\, \nu\, u.\, A & \\
\texttt{NEW}-\texttt{PAR} & A \mid \nu\, u.\, B & \equiv & \nu\, u.\, (A \mid B) & \texttt{if } u \notin \mathsf{fn}(A) \cup \mathsf{fv}(A) \\
\texttt{ALIAS} & \nu\, x.\, \{M/x\} & \equiv & 0 & \\
\texttt{REWRITE} & \{M/x\} & \equiv & \{N/x\} & \texttt{if } M =_E N
\end{array}
$$

Figure 3: Structural equivalence in the applied pi calculus.

$$
\begin{array}{ll}
\texttt{COMM} & \mathsf{out}(c,x).\, P \mid \mathsf{in}(c,x).\, Q \quad \to \quad P \mid Q \\
\texttt{THEN} & \texttt{if } N =_E N \texttt{ then } P \texttt{ else } Q \quad \to \quad P \\
\texttt{ELSE} & \texttt{if } M =_E N \texttt{ then } P \texttt{ else } Q \quad \to \quad Q \\
 & \qquad \texttt{for ground terms } M,\ N \texttt{ where } M \neq_E N
\end{array}
$$

Figure 4: Internal reduction in the applied pi calculus.

defines a relation $A \xrightarrow{\alpha} A'$ as in Figure 5. The label α is either reading a term from the process's environment, or sending a name or a variable of base type to the environment.

$$
\begin{array}{ll}
\texttt{IN} & \mathsf{in}(c,x).\, P \xrightarrow{\mathsf{in}(c,M)} P\{M/x\} \\
\texttt{OUT}-\texttt{ATOM} & \mathsf{out}(c,u).\, P \xrightarrow{\mathsf{out}(c,u)} P \\
\texttt{OPEN}-\texttt{ATOM} & \dfrac{A \xrightarrow{\mathsf{out}(c,u)} A' \quad u \neq c}{\nu\, u.\, A \xrightarrow{\nu\, u.\, \mathsf{out}(c,u)} A'} \\
\texttt{SCOPE} & \dfrac{A \xrightarrow{\alpha} A' \quad u \texttt{ does not occur in } \alpha}{\nu\, u.\, A \xrightarrow{\alpha} \nu\, u.\, A'} \\
\texttt{PAR} & \dfrac{A \xrightarrow{\alpha} A' \quad \mathsf{bv}(\alpha) \cup \mathsf{fv}(B) = \mathsf{bn}(\alpha) \cap \mathsf{fn}(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B} \\
\texttt{STRUCT} & \dfrac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad A' \equiv B'}{A \xrightarrow{\alpha} A'}
\end{array}
$$

Figure 5: Labelled reduction in the applied pi calculus.

2.3. Equivalences

The applied pi calculus defines *observational equivalence* and *labelled bisimilarity* to model the indistinguishability of two processes by the adversary. It is proved that the two relations coincide when active substitutions are of base type [22, 29]. We mainly use the labelled bisimilarity for the convenience of proofs. Labelled bisimilarity is based on *static equivalence*: labelled bisimilarity compares the dynamic behaviour of processes, while static equivalence compares their static states (as represented by their frames).

Definition 1 (static equivalence)**.** *Two terms M and N are equal in the frame B, written as $(M =_E N)B$, iff there exists a set of restricted names $\tilde{n}$ and a substitution σ such that $B \equiv \nu\, \tilde{n}.\, \sigma$, $M\sigma =_E N\sigma$ and $\tilde{n} \cap (\mathsf{fn}(M) \cup \mathsf{fn}(N)) = \emptyset$.*

Closed frames B and B' are statically equivalent, denoted as $B \approx_s B'$, if
(1) $\mathsf{domain}(B) = \mathsf{domain}(B')$*;*
(2) $\forall$ *terms* M, N*:* $(M =_E N)B$ *iff* $(M =_E N)B'$*.*

Extended processes A, A' are statically equivalent, denoted as $A \approx_s A'$, if their frames are statically equivalent: $\mathsf{frame}(A) \approx_s \mathsf{frame}(A')$.

Definition 2 (labelled bisimilarity)**.** Labelled bisimilarity $(\approx_\ell)$ *is the largest symmetric relation $\mathscr{R}$ on closed extended processes, such that $A \mathscr{R} B$ implies:*
(1) $A \approx_s B$*;*
(2) if $A \to A'$ *then* $B \to^* B'$ *and* $A' \mathscr{R} B'$ *for some* B'*;*
(3) if $A \xrightarrow{\alpha} A'$ *and* $\mathsf{fv}(\alpha) \subseteq \mathsf{domain}(A)$ *and* $\mathsf{bn}(\alpha) \cap \mathsf{fn}(B) = \emptyset$*; then* $B \to^* \xrightarrow{\alpha} \to^* B'$ *and* $A' \mathscr{R} B'$ *for some* B'*, where * denotes zero or more.*

3. ProVerif

The verification of protocols modelled in the applied pi calculus is supported by an automatic verification tool ProVerif [23, 30, 31]. The tool has been used to verify many security and privacy properties, e.g., see [32, 33, 34, 35, 36, 9, 37, 21].

ProVerif takes a protocol and a property modelled in the applied pi calculus as input, returns a proof of correctness or flaws as output. A protocol modelled in the applied pi calculus is translated to Horn clauses [38]. The adversary ability is interpreted

as Horn clauses as well. Using these clauses, the verification of secrecy (e.g., secrecy of M) is to determine whether a predicate (e.g., "$attack : M$" meaning that attack knows M) can be deduced. However, not all properties can be expressed as such predicates. Many of such properties can be expressed as equivalences of processes, for example, strong secrecy which is defined as the adversary's inability to distinguish when the secret changes. Therefore, in addition, ProVerif provides automatic verification of labelled bisimilarity of two processes which differ only in the choice of some terms [39]. Strong secrecy of a variable x can be verified by querying "noninterf x", meaning that no matter how the variable x is instantiated, the adversary cannot detect any difference between these instantiations. An operation "$choice[a,b]$" is also used to model the different choices of a term in the two processes. Using this operation, the two processes can be written as one process – a *bi-process*. Using the first parameter of all "*choice*" operations in a bi-process P, we obtain one side of the equivalence (denoted as $\mathsf{fst}(P)$); using the second parameters, we obtain the other side (denoted as $\mathsf{snd}(P)$). Given a bi-process P, ProVerif determines whether $\mathsf{fst}(P)$ is labelled bisimilar to $\mathsf{snd}(P)$.

4. Formalisation of privacy notions in e-auctions

We formalise the two identified privacy notions, bidding-price-secrecy and receipt-freeness, using the applied pi calculus in the context of sealed-bid e-auctions.

An auction protocol is essentially a specification of the behaviour of the roles participating in the protocol. A protocol normally involves two roles: bidders and auctioneers, e.g., the AS02 protocol. Some protocols may involve other roles, such as role sealers in the HRM14 protocol. The behaviour of each role is a sequence of message inputs, message outputs and conditional evaluations on messages. Recall that each message is modelled as a term – names, variables, or function symbols applied on other terms, in the applied pi calculus; and the message inputs, outputs and conditional evaluations are modelled as atomic events in the applied pi calculus. Thus, the behaviour of each role specified in a protocol is formally defined as a process. Therefore, an auction protocol with n roles (including the role bidder defined as process P_b and auctioneer defined as process P_a) is formally defined as a tuple $(P_b, P_a, P_{role_1}, \ldots, P_{role_{n-2}})$ where

P_{role_i} defines the behaviour of role i. These processes are composed using parallel operator with communication channels and auxiliary data. The composed process is then the whole model of the entire protocol. For instance, an e-auction protocol with n_b bidders and n_a auctioneers can be modelled as:

$$P_{bid} := \nu\, chandata.\, (P_K \mid P_{b1} \mid \cdots \mid P_{bn_b} \mid P_{a1} \mid \cdots \mid P_{an_a}),$$

where P_{bi} is an instance of a bidder process, P_{aj} is an instance of an auctioneer process, P_K is the key distribution process, and *chandata* models private data and private channels.

4.1. Bidding-price-secrecy

Bidding-price-secrecy for non-winning bidders can be formalised in two levels: standard bidding-price-secrecy and strong bidding-price-secrecy. Standard bidding-price-secrecy is formalised as the adversary cannot derive the bidding price of a non-winning bidder. Strong bidding-price-secrecy is formalised as the adversary cannot even distinguish between the case when a bidder bids for price a and the case when the bidder bids for price c. In other words, the adversary cannot tell whether a bidder changes his bidding price from a to c.

Formalisation similar to strong bidding-price-secrecy has been used, e.g., vote-privacy [15]: a process in which voter v_A votes for a ($P_{vA}\{a/vote\}$) and voter v_B votes for c ($P_{vB}\{c/vote\}$) is observationally equivalent to a process where v_A votes for c ($P_{vA}\{c/vote\}$) and v_B votes for a ($P_{vB}\{a/vote\}$). The idea is that even if all other voters reveal how they voted, the adversary cannot deduce the votes of voter v_A and voter v_B, given voter v_A and voter v_B counterbalance each other. Different from privacy in voting where the voting result is published, in sealed-bid e-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 a higher price so that non-winning bids are not revealed in the opening phase. Therefore, strong bidding-price-secrecy is formalised as follows:

Definition 3 (strong bidding-price-secrecy for non-winning bidders)**.** *An auction pro-*

tocol P_{bid}, with a bidder sub-process represented as P_b, satisfies strong bidding-price-secrecy for non-winning bidders, if for all possible bidders b_A and b_B we have:

$$\mathscr{C}_b[P_{bA}\{^a/_{p_b}\} \mid P_{bB}\{^d/_{p_b}\}] \approx_\ell \mathscr{C}_b[P_{bA}\{^c/_{p_b}\} \mid P_{bB}\{^d/_{p_b}\}]$$

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

The context $\mathscr{C}_b[\_]$ is used to capture the assumption made on the checked protocol, usually it includes the other honest participants in the protocol, i.e., $\mathscr{C}_b[\_] := \nu\ chandata.\ (P_K \mid P_{b1} \mid \cdots \mid P_{b(n_b-2)} \mid \_ \mid P_{a1} \mid \cdots \mid P_{an_a})$. The process P_{bA} is a bidder process executed by a non-winning bidder b_A. The process P_{bB} is a bidder process executed by another bidder b_B who bids for a higher price. The variable p_b indicates the bidding price in a process. Hence, the processes $P_{bA}\{^a/_{p_b}\}$, $P_{bA}\{^c/_{p_b}\}$, and $P_{bB}\{^d/_{p_b}\}$ capture bidder b_A bidding for price a, bidder b_A bidding for price c, and bidder b_B bidding for price d, respectively. The intuition is that the adversary cannot determine whether a non-winning bidder bids for price a or price c, provided there exists another bidder who bids for a higher price d.

4.2. 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 private information, while receipt-freeness protects a bidder's privacy when the bidder is willing (or coerced) to reveal this.

In voting, receipt-freeness can be formalised as an observational equivalence [15]. A voting protocol satisfies receipt-freeness if the adversary cannot distinguish (observational equivalence) whether a voter genuinely did his voting or that voter claimed to do so, but voted for another candidate. In order to model observational equivalence, the situation that a voter provides his secret information to the adversary is modelled first:

Definition 4 (process $P^{\mathtt{chc}}$ [15])**.** *Let P be a plain process and* `chc` *a channel name. $P^{\mathtt{chc}}$, the process that shares all of P's secrets, is defined as:*

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

Delaune *et al.* also define process transformation $A^{\backslash \mathsf{out}(\mathtt{chc},\cdot)}$, which can be considered as a version of process A that hides all outputs on public channel $\mathtt{chc}$.

Definition 5 (process $A^{\backslash \mathsf{out}(\mathtt{chc},\cdot)}$ [15])**.** *Let A be an extended process. The process $A^{\backslash \mathsf{out}(\mathtt{chc},\cdot)}$ is defined as $\nu\, \mathtt{chc}.\, (A \mid !\mathsf{in}(\mathtt{chc},x))$.*

When modelling online auction protocols, we also need to model the situation in which a bidder shares his secret information with the adversary. We use the above definition directly in our model. Intuitively, a bidder who shares information with the adversary sends all input of base type and all freshly generated names of base type to the adversary over a public channel $\mathtt{chc}$. It is assumed that public channels are under the adversary's control.

Now, we can define receipt-freeness for sealed-bid e-auction protocols. Again, we need a bidder process P_{bB} in which bidder b_B bids for a higher price d, so that non-winning bids are not revealed. Intuitively, if a non-winning bidder 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 for non-winning bidders)**.** *An auction protocol P_{bid}, with a bidder sub-process P_b, satisfies receipt-freeness for non-winning bidders, if there exists a closed plain process P_f such that:*

1. $P_f{}^{\backslash \mathsf{out}(\mathtt{chc},\cdot)} \approx_\ell P_{bA}\{^c/_{p_b}\}$,
2. $\mathscr{C}_b[P_{bA}\{^a/_{p_b}\}^{\mathtt{chc}} \mid P_{bB}\{^d/_{p_b}\}] \approx_\ell \mathscr{C}_b[P_f \mid P_{bB}\{^d/_{p_b}\}]$

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

Process P_f is a bidder process in which bidder b_A bids for price c but communicates with the adversary and tells the adversary that he bids for price a. Process $P_{bA}\{^c/_{p_b}\}$ is a bidder process in which bidder b_A bids for price c. Process $P_{bA}\{^a/_{p_b}\}^{\mathtt{chc}}$ is a bidder process in which bidder b_A bids for price a and shares his secrets with the adversary. Process $P_{bB}\{^d/_{p_b}\}$ is a bidder process in which bidder b_B bids for a higher price d. The first equivalence says that ignoring the outputs bidder b_A makes on the channel chc to the adversary, P_f looks like a normal process in which b_A bids for price c. The second equivalence says that the adversary cannot tell the difference between the situation in which b_A obeys the adversary's commands and bids for price a, and the situation in which b_A pretends to cooperate but actually bids for price c, provided there is a bidding process P_{bB} that bids higher, ensuring that bidding processes P_{bA} and P_f are not winners. Receipt-freeness is a stronger property than bidding-price-secrecy, for the same reason as receipt-freeness in e-voting is stronger than vote-privacy (as shown in [15]).

5. Case study: the AS02 protocol

After receipt-freeness has been identified in sealed-bid e-auctions. Abe and Suzuki proposed the first protocol which aims to prevent bid-rigging – the AS02 protocol [4]. In this section, we analyse both *bidding-price-secrecy* and *receipt-freeness* for non-winning bidders in the AS02 protocol. The main steps of the protocol are depicted in Figure 6.

5.1. Introduction

This protocol is a sealed-bid e-auction protocol. The 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.[2]

5.2. Physical assumptions

In order to ensure privacy of bidders, the protocol has two physical assumptions:

a1: a bidding booth for the bidders, and

a2: 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 the adversary. The untappable channels ensure no adversary can see messages sent.

5.3. Settings

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 $p = 2q + 1$.

5.4. Description of the protocol

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_b to bid for. Then he generates a bit-commitment for each price p_ℓ ($1 \leq \ell \leq m$), using the following formula:

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

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

[2] The protocol does not specify how to resolve the case where there are fewer bidding items than winners.

discrete logs ($\log_g cmt^{\mathsf{p}_1}, \ldots, \log_g cmt^{\mathsf{p}_m}$) using interactive zero-knowledge proofs. Finally, he computes t-out-of-k[3] 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.

Figure 6: The AS02 protocol.

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

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

struct the secret seed r_ℓ, and check for each bidder whether

$$cmt^{\mathsf{p}_\ell} \stackrel{?}{=} g^{M_{yes}} h^{r_\ell}.$$

If there exist some bidders for which the above equivalences are satisfied, the auctioneers finish checking the current price and then stop. In this case, the price p_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 p_ℓ, the auctioneers repeat the above process on the next lower price.

5.5. Claimed properties

The authors claim the following properties: bidding-price-secrecy and receipt-freeness for non-winning bidders. Intuitively, the bidding price of each bidder is sealed in the bidding phase, and only the winning bidder's bidding price is revealed in the opening phase, thus the adversary does not know the bidding price for non-winning bidders, thus standard bidding-price-secrecy is satisfied. The strong bidding-price-secrecy is satisfied mainly due to the random number used in calculating the bit-commitments.

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

$$cmt^{\mathsf{p}_\ell} = g^M \cdot h^{r_\ell} = g^M \cdot (g^x)^{r_\ell} = g^{M+xr_\ell},$$

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

$$\log_g cmt^{\mathsf{p}_\ell} = M_{yes} + xr_\ell$$

he can calculate a fake r'_ℓ such that:

$$\log_g cmt^{\mathsf{p}_\ell} = M_{no} + xr'_\ell \quad \text{and} \quad r'_\ell = (M_{yes} + xr_\ell - M_{no})/x.$$

Using the fake r'_ℓ, the bidder can show that the bit-commitment cmt^{p_ℓ} is opened as message M_{no}, which means the bidder did not bid for price p_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, i.e., receipt-freeness is satisfied.

5.6. Modelling

We model the AS02 protocol in applied pi, using two simplifications:

s1: one honest auctioneer; and

s2: perfect zero knowledge proofs.

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 simplify 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 the zero knowledge proofs are perfect, that is, 1) we assume each bidder knows his secret key and discrete logs of bit-commitments and 2) non-eligible bids are not allowed (modelled as the adversary is not able to generate eligible bids), since the zero knowledge proofs are used to prevent non-eligible bidders from submitting bids.

In addition, the AS02 does not specify how the auctioneers tell the signed public key from the signed commitments generated by the same bidder. In order for the auctioneer to distinguish the two messages, in our modelling,

s3: we use a symbol k in the signed public key messages.

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 nextbidder to find the next bidder in the bidder list, and function nextprice to find the next lower price in the price list.

$$\begin{array}{rclcrcl}
\mathsf{nextbidder}(\mathsf{b}_1) & = & \mathsf{b}_2 & \qquad & \mathsf{nextprice}(\mathsf{p}_m) & = & \mathsf{p}_{m-1} \\
\ldots & & & & \ldots & & \\
\mathsf{nextbidder}(\mathsf{b}_{n-1}) & = & \mathsf{b}_n & & \mathsf{nextprice}(\mathsf{p}_2) & = & \mathsf{p}_1 \\
\mathsf{nextbidder}(\mathsf{b}_n) & = & \bot & & \mathsf{nextprice}(\mathsf{p}_1) & = & \top
\end{array}$$

Function 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 modelled as a function commit with arity 3 (a 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.

$$\begin{array}{rcll}
\mathsf{commit}(r, \mathsf{pk}(sk_b), \mathsf{M}_{yes}) & =_E & \mathsf{commit}(\mathsf{f}(r), \mathsf{pk}(sk_b), \mathsf{M}_{no}) & \textbf{et1} \\
\mathsf{commit}(r, \mathsf{pk}(sk_b), \mathsf{M}_{no}) & =_E & \mathsf{commit}(\mathsf{f}(r), \mathsf{pk}(sk_b), \mathsf{M}_{yes}) & \textbf{et2} \\
\mathsf{open}(\mathsf{commit}(r, \mathsf{pk}(sk_b), m), r, \mathsf{pk}(sk_b)) & =_E & m &
\end{array}$$

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 a bidder, 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[4]. In fact, from the formula in Section 5.5, $\mathsf{f}(r)$ returns the alternative secret seed of r, which leads to the opposite opening result of a bit-commitment. Thus, given $\mathsf{f}(r)$, which opens a bit-commitment as $\mathsf{M}_{yes}(\mathsf{M}_{no})$, the bidder can also compute r which leads to $\mathsf{M}_{no}(\mathsf{M}_{yes})$,

[4]The bidder proves that he knows his secret key and discrete logs of bit-commitments, using zero-knowledge proofs. Due to the perfect zero-knowledge assumption, the bidder is assumed to have that knowledge; and the adversary is assume not to have the knowledge and thus cannot apply f function. Hence, f is defined as *private* in Figure 7, meaning that the adversary cannot apply it.

$$\textit{fun}\ \mathsf{b}_1/0,\ \ldots,\ \textit{fun}\ \mathsf{b}_n/0,\ \textit{fun}\ \mathsf{p}_1/0,\ \ldots,\ \textit{fun}\ \mathsf{p}_m/0,\ \textit{fun}\ \mathsf{M}_{yes}/0,\ \textit{fun}\ \mathsf{M}_{no}/0,$$
$$\textit{fun}\ \mathsf{true}/0,\ \textit{fun}\ \mathsf{pk}/1,\ \textit{fun}\ \mathsf{commit}/3,\ \textit{fun}\ \mathsf{sign}/2,\ \textit{private fun}\ \mathsf{f}/1,\ \textit{fun}\ \mathsf{k}/0$$

Figure 7: Functions.

reduc	$\mathsf{checksign}(\mathsf{sign}(m,sk),\mathsf{pk}(sk))$	$=$	true
reduc	$\mathsf{getmsg}(\mathsf{sign}(m,sk))$	$=$	m
equation	$\mathsf{commit}(r,\mathsf{pk}(sk_b),\mathsf{M}_{no})$	$=$	$\mathsf{commit}(\mathsf{f}(r),\mathsf{pk}(sk_b),\mathsf{M}_{yes})$
equation	$\mathsf{f}(\mathsf{f}(r))$	$=$	r
reduc	$\mathsf{open}(\mathsf{commit}(r,\mathsf{pk}(sk_b),m),r,\mathsf{pk}(sk_b))$	$=$	m

Figure 8: Equational theory.

i.e., $\mathsf{f}(\mathsf{f}(r)) =_E r$. The first equivalence (**et1**) 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 (**et2**) shows that the opposite situation also holds. The third equivalence models that a bidder can open a bit-commitment with the corresponding public key and secret seed (potentially being fake). These three equivalences allow a bidder to open a bit-commitment as if he bids for that price, when actually he does not; and vice versa. All functions defined in this model are shown in Figure 7 and the equational theory is shown in Figure 8. Note that the functions and equational theory are defined in the ProVerif untyped style (for details, see [40]), which slightly differs from applied pi [5]. In particular, *fun* is used to denote function in ProVerif, the numerical number following a function symbol is the arity of the function, and *reduc* and *equation* are used to denote the equational theory in ProVerif (instead of using $=_E$ in applied pi) [6].

Main process. For each bidder b_j, the main process (see Figure 9) generates two private channels $\mathtt{privch}_{\mathsf{b}_j}$ (**m1**) and $\mathtt{privcha}_{\mathsf{b}_j}$ (**m2**). These channels are used for

[5] In the untyped ProVerif style, function nextbidder and nextprice cannot be used as in Figure 12. In the ProVerif code, we consider them as predefined. Additionally, the two equations **et1** and **et2** can be unified into one, due to the equation $\mathsf{f}(\mathsf{f}(r)) =_E r$, e.g., by replacing r with $\mathsf{f}(r)$ in **et1**, we obtain $\mathsf{commit}(\mathsf{f}(r),\mathsf{pk}(sk_b),\mathsf{M}_{yes}) =_E \mathsf{commit}(\mathsf{f}(\mathsf{f}(r)),\mathsf{pk}(sk_b),\mathsf{M}_{no})$. Since $\mathsf{f}(\mathsf{f}(r)) =_E r$, the equation coincides with **et2**.

[6] The ProVerif code is available at `http://satoss.uni.lu/projects/epriv`, under title 'Formal analysis of a receipt-free auction protocol in the applied pi'.

instantiating a bidder process. In particular, a bidder receives his secret signing key from channel $\texttt{privch}_{\mathsf{b}_j}$; and the auctioneer receives the corresponding public key from channel $\texttt{privcha}_{\mathsf{b}_j}$. In addition, the main process generates an untappable channel $\texttt{untapch}_{\mathsf{b}_j}$ for bidders b_j (**m3**). The untappable channel is shared between each bidder and the auctioneer. The private channels $\texttt{synch}_{\mathsf{b}_1}, \ldots, \texttt{synch}_{\mathsf{b}_n}$ are generated for modelling convenience (**m4**). These channels are used by the auctioneer to collect all necessary information before moving to the opening phase. The main process launches a key generating process P_K (**m5**), n instantiations of the bidder process (**m5-m8**) and an instance of the auctioneer process (**m8**). Four variables need to be instantiated in an instance of bidder process: the bidding price p_b, the untappable channel *untapch*, the private channel *privch* and the public channel for that bidder *ch*. For the simplicity of modelling, each bidder b_j has a distinct public channel $\texttt{ch}_{\mathsf{b}_j}$. The correspondence between $\texttt{privcha}_{\mathsf{b}_j}$, $\texttt{untapch}_{\mathsf{b}_j}$ and $\texttt{ch}_{\mathsf{b}_j}$ allows the auctioneer to distinguish messages from the same bidder. In this way, we avoid modelling the auctioneer classifying messages by bidders (by checking signatures). Note that $p_{\mathsf{b}_1}, \ldots, p_{\mathsf{b}_n}$ are parameters, each of these parameters has to be instantiated with a constant in the published price list $\mathsf{p}_1, \ldots, \mathsf{p}_m$.

$P_{AS02} :=$
m1. $\quad \nu\, \texttt{privch}_{\mathsf{b}_1}.\, \nu\, \texttt{privch}_{\mathsf{b}_2}.\, \cdots.\, \nu\, \texttt{privch}_{\mathsf{b}_n}.$
m2. $\quad \nu\, \texttt{privcha}_{\mathsf{b}_1}.\, \nu\, \texttt{privcha}_{\mathsf{b}_2}.\, \cdots.\, \nu\, \texttt{privcha}_{\mathsf{b}_n}.$
m3. $\quad \nu\, \texttt{untapch}_{\mathsf{b}_1}.\, \nu\, \texttt{untapch}_{\mathsf{b}_2}.\, \cdots.\, \nu\, \texttt{untapch}_{\mathsf{b}_n}.$
m4. $\quad \nu\, \texttt{synch}_{\mathsf{b}_1}.\, \nu\, \texttt{synch}_{\mathsf{b}_2}.\, \cdots.\, \nu\, \texttt{synch}_{\mathsf{b}_n}.$
m5. $\quad (P_K \mid (\texttt{let}\ \ p_b = p_{\mathsf{b}_1}\ \ \texttt{in let}\ \ \mathit{untapch} = \texttt{untapch}_{\mathsf{b}_1}\ \ \texttt{in}$
m6. $\qquad\quad \texttt{let}\ \ \mathit{privch} = \texttt{privch}_{\mathsf{b}_1}\ \ \texttt{in let}\ \ \mathit{ch} = \texttt{ch}_{\mathsf{b}_1}\ \ \texttt{in}\ \ P_b) \mid$
m7. $\quad \cdots \mid (\texttt{let}\ \ p_b = p_{\mathsf{b}_n}\ \ \texttt{in let}\ \ \mathit{untapch} = \texttt{untapch}_{\mathsf{b}_n}\ \ \texttt{in}$
m8. $\qquad\quad \texttt{let}\ \ \mathit{privch} = \texttt{privch}_{\mathsf{b}_n}\ \ \texttt{in let}\ \ \mathit{ch} = \texttt{ch}_{\mathsf{b}_n}\ \ \texttt{in}\ \ P_b) \mid P_a)$

Figure 9: The main process.

Key distribution process. This process generates and distributes keying material modelling a PKI – public key infrastructure (Figure 10). This process first generates n secret keys (**k1**). Each bidder b_j has one secret key $\texttt{ssk}_{\mathsf{b}_j}$ for signing messages. Each secret key corresponds to a public key (**k2-k4**). Each secret key is assigned to a bidder pro-

cess by being sent to the bidder over the private channel $\mathtt{privch}_{\mathsf{b}_j}$ corresponding to that bidder (**k5**). The corresponding public key is sent to the auctioneer over the private channel $\mathtt{privcha}_{\mathsf{b}_j}$ (**k6**) and is published over the public channel $\mathtt{ch}_{\mathsf{b}_j}$ such that the adversary knows the keys (**k7**). Therefore, only a bidder knows his own secret key, and everyone, including the adversary, knows each bidder's public key. Sending each public key to the auctioneer over a private channel, models the following protocol setting: There are fix number of bidders in sealed-bid auctions, and the auctioneer knows each bidder's public signing key as predetermined knowledge. This setting also disallows the adversary to generate an eligible bid (to capture perfect zero knowledge proof), as the adversary does not know any secret key which is needed to sign a bid.

$P_K :=$
k1. $\quad \nu\, \mathtt{ssk}_{\mathsf{b}_1}.\, \nu\, \mathtt{ssk}_{\mathsf{b}_2}.\, \cdots.\, \nu\, \mathtt{ssk}_{\mathsf{b}_n}.$
k2. $\quad \mathtt{let}\ \ spk_{\mathsf{b}_1} = \mathsf{pk}(\mathtt{ssk}_{\mathsf{b}_1})\ \ \mathtt{in}$
k3. $\quad \cdots$
k4. $\quad \mathtt{let}\ \ spk_{\mathsf{b}_n} = \mathsf{pk}(\mathtt{ssk}_{\mathsf{b}_n})\ \ \mathtt{in}$
k5. $\quad (\mathsf{out}(\mathtt{privch}_{\mathsf{b}_1}, \mathtt{ssk}_{\mathsf{b}_1}) \mid \cdots \mid \mathsf{out}(\mathtt{privch}_{\mathsf{b}_n}, \mathtt{ssk}_{\mathsf{b}_n}) \mid$
k6. $\quad \mathsf{out}(\mathtt{privcha}_{\mathsf{b}_1}, spk_{\mathsf{b}_1}) \mid \cdots \mid \mathsf{out}(\mathtt{privcha}_{\mathsf{b}_n}, spk_{\mathsf{b}_n}) \mid$
k7. $\quad \mathsf{out}(\mathtt{ch}_{\mathsf{b}_1}, spk_{\mathsf{b}_1}) \mid \cdots \mid \mathsf{out}(\mathtt{ch}_{\mathsf{b}_n}, spk_{\mathsf{b}_n}))$

Figure 10: The key distribution process.

Bidder process. The applied pi calculus process for a bidder P_b is given in Figure 11. First, a bidder receives his secret signature key from his private channel (**b1**). Next, the bidder generates his secret key $\mathtt{sk}_b$ (i.e., the secret key x in Section 5.4), signs the corresponding public key (i.e., $h = g^x$ in Section 5.4) and publishes the signed message (**b2**). To indicate that this message contains a key, we add k into the message (see s3). In addition, the bidder chooses a series of random numbers $\mathtt{r}_1, \ldots, \mathtt{r}_m$ as secret seeds (**b3**). The bidder then computes each bit-commitment cmt^{p_ℓ} as described in Section 5.4. For each price, the bidder computes a commitment: if the price is the bidding price, then the bidder commits 'yes' with M_{yes}, otherwise, the bidder commits 'no' with M_{no} (**b4-b6** when he bids for p_1). Finally, the bidder publishes the series of bit-commitments $cmt^{\mathsf{p}_1}, \ldots, cmt^{\mathsf{p}_m}$ with his signature (**b7**), and sends the signed series of secret seeds to the auctioneer through the untappable channel (**b8**). The process of

bidding for other prices is similar (**b9-b13** when bidding for p_m). As we assume there is only one honest auctioneer in the model, we do not need to model secret shares.

$P_b :=$
b1. $\mathsf{in}(privch, ssk_b).$
b2. $\nu\ \mathtt{sk}_b.\ \mathsf{out}(ch, \mathsf{sign}((\mathsf{pk}(\mathtt{sk}_b), \mathsf{k}), ssk_b)).$
b3. $\nu\ \mathtt{r}_1.\ \cdots.\ \nu\ \mathtt{r}_m.$
b4. $\mathtt{if}\ \ p_b = \mathsf{p}_1\ \ \mathtt{then}$
b5. $(\mathtt{let}\ \ cmt^{\mathsf{p}_1} = \mathsf{commit}(\mathtt{r}_1, \mathsf{pk}(\mathtt{sk}_b), \mathsf{M}_{yes})\ \ \mathtt{in}$
$\ldots$
b6. $\mathtt{let}\ \ cmt^{\mathsf{p}_1} = \mathsf{commit}(\mathtt{r}_1, \mathsf{pk}(\mathtt{sk}_b), \mathsf{M}_{no})\ \ \mathtt{in}$
b7. $\mathsf{out}(ch, \mathsf{sign}((cmt^{\mathsf{p}_1}, \cdots, cmt^{\mathsf{p}_m}), ssk_b)).$
b8. $\mathsf{out}(untapch, \mathsf{sign}((\mathtt{r}_1, \cdots, \mathtt{r}_m), ssk_b)))$
$\ldots$
b9. $\mathtt{if}\ \ p_b = \mathsf{p}_m\ \ \mathtt{then}$
b10. $(\mathtt{let}\ \ cmt^{\mathsf{p}_m} = \mathsf{commit}(\mathtt{r}_m, \mathsf{pk}(\mathtt{sk}_b), \mathsf{M}_{no})\ \ \mathtt{in}$
$\ldots$
b11. $\mathtt{let}\ \ cmt^{\mathsf{p}_m} = \mathsf{commit}(\mathtt{r}_m, \mathsf{pk}(\mathtt{sk}_b), \mathsf{M}_{yes})\ \ \mathtt{in}$
b12. $\mathsf{out}(ch, \mathsf{sign}((cmt^{\mathsf{p}_1}, \cdots, cmt^{\mathsf{p}_m}), ssk_b)).$
b13. $\mathsf{out}(untapch, \mathsf{sign}((\mathtt{r}_1, \cdots, \mathtt{r}_m), ssk_b)))$

Figure 11: The bidder process.

Auctioneer process. During the bidding phase, the auctioneer launches n copies of sub-process *readinfo* to gather information from each bidder b_j (**a1**).

In details, the auctioneer collects public signature key *spk* (**r1**) and the signed committing public key *signedpk* (supposed to be $\mathsf{sign}((\mathsf{pk}(sk_{\mathsf{b}_j}), \mathsf{k}), ssk_{\mathsf{b}_j})$ for bidder b_j) (**r2**) of each bidder. The auctioneer verifies whether the committing public key is signed with the right signature (**r3**) and obtains the committing public key *pk* from *signedpk* (**r4**). Next, the auctioneer reads in the signed commitments *signedcommit* of the bidder (**r5**) and verifies the signature (**r6**). If the commitments are correctly signed, the auctioneer obtains the series of bit-commitments $cmt^{\mathsf{p}_1}, \ldots, cmt^{\mathsf{p}_m}$ (**r7**), then the auctioneer reads in the secret seeds *sr* from the untappable channel of the bidder (**r8**). The auctioneer verifies the signature (**r9**). If the secret seeds are correctly signed, the auctioneer obtains the secret seeds $ss^{\mathsf{p}_1}, \ldots, ss^{\mathsf{p}_m}$ (**r10**). Finally, the auctioneer sends the signal that information collecting for the bidder has finished, over the channel *synch* (**r9**). In addition, the collected information (the committing public key, the commitments, the

$P_a :=$

a1. `let` $ch = \mathtt{ch}_{\mathsf{b}_1}$ `in let` $privcha = \mathtt{privcha}_{\mathsf{b}_1}$ `in`
`let` $synch = \mathtt{synch}_{\mathsf{b}_1}$ `in let` $untapch = \mathtt{untapch}_{\mathsf{b}_1}$ `in` $readinfo \mid$
$\cdots \mid$
`let` $ch = \mathtt{ch}_{\mathsf{b}_n}$ `in let` $privcha = \mathtt{privcha}_{\mathsf{b}_n}$ `in`
`let` $synch = \mathtt{synch}_{\mathsf{b}_n}$ `in let` $untapch = \mathtt{untapch}_{\mathsf{b}_n}$ `in` $readinfo \mid$
a2. $\mathsf{in}(\mathtt{synch}_{\mathsf{b}_1}, (pk_{\mathsf{b}_1}, cmt^{\mathsf{p}_1}_{\mathsf{b}_1}, \ldots, cmt^{\mathsf{p}_m}_{\mathsf{b}_1}, ss^{\mathsf{p}_1}_{\mathsf{b}_1}, \ldots, ss^{\mathsf{p}_m}_{\mathsf{b}_1}))$.
$\cdots$.
$\mathsf{in}(\mathtt{synch}_{\mathsf{b}_n}, (pk_{\mathsf{b}_n}, cmt^{\mathsf{p}_1}_{\mathsf{b}_n}, \ldots, cmt^{\mathsf{p}_m}_{\mathsf{b}_n}, ss^{\mathsf{p}_1}_{\mathsf{b}_n}, \ldots, ss^{\mathsf{p}_m}_{\mathsf{b}_n}))$.
a3. `if` $cmt^{\mathsf{p}_m}_{\mathsf{b}_1} = \mathsf{commit}(ss^{\mathsf{p}_m}_{\mathsf{b}_1}, pk_{\mathsf{b}_1}, \mathsf{M}_{yes})$
a4. `then` $\mathsf{out}(\mathtt{winnerch}, (\mathsf{p}_m, \mathsf{b}_1))$.
a5. `if` $\mathsf{nextbidder}(\mathsf{b}_1) = \bot$
a6. `then` 0
a7. `else` $checknextb^{\mathsf{p}_m}_{\mathsf{nextbidder}(\mathsf{b}_1)}$
a8. `else if` $\mathsf{nextbidder}(\mathsf{b}_1) = \bot$
a9. `then if` $\mathsf{nextprice}(\mathsf{p}_m) = \top$
a10. `then` 0
a11. `else` $checknextbnp^{\mathsf{nextprice}(\mathsf{p}_m)}_{\mathsf{b}_1}$
a12. `else` $checknextbnp^{\mathsf{p}_m}_{\mathsf{nextbidder}(\mathsf{b}_1)}$

Figure 12: The auctioneer process.

secret seeds) is sent to the sub-process in which the winning bidder is determined.

Next the auctioneer needs to synchronise with all bidders (**a2**). The auctioneer process is not allowed to continue until all bidders reach the end of the bidding phase. In the opening phase, the auctioneer evaluates whether the following holds $cmt^{\mathsf{p}_m}_{\mathsf{b}_j} \stackrel{?}{=} \mathsf{commit}(ss^{\mathsf{p}_m}_{\mathsf{b}_j}, pk_{\mathsf{b}_j}, \mathsf{M}_{yes})$ for each bidder (**a3, a7, a12**). If the two values are equivalent for the first bidder b_1 (**a3**), bidder b_1 has bid for that price, otherwise, bidder b_1 has not bid for that price. When bidder b_1 has bid for that price, the auctioneer publishes the bidder together with the price over the public channel `winnerch` (**a4**), then the auctioneer checks the evaluation for the next bidder (if exists) (**a7**). Once the auctioneer has evaluated for every bidder (**a5** when b_1 is the only bidder) and has determined the set of winning bidders (**a4**), he stops the process (**a6**). When bidder b_1 has not bid for that price, the auctioneer checks the evaluation for the next bidder (if exists) (**a12**). Once the auctioneer has evaluated for every bidder and no winner has been found (**a8** when b_1 is the only bidder), the auctioneer repeats the evaluation steps for each bidder at the next lower price (**a11**). If the next lower price does not exist (**a9** when p_m is

readinfo :=

r1. $\mathsf{in}(privcha, spk)$.
r2. $\mathsf{in}(ch, signedpk)$.
r3. `if` $\mathsf{checksign}(signedpk, spk) = \mathsf{true}$
r4. `then let` $(pk, = \mathsf{k}) = \mathsf{getmsg}(signedpk)$ `in`
r5. $\mathsf{in}(ch, signedcommit)$.
r6. `if` $\mathsf{checksign}(signedcommit, spk) = \mathsf{true}$
r7. `then let` $(cmt^{\mathsf{p}_1}, \ldots, cmt^{\mathsf{p}_m}) = \mathsf{getmsg}(signedcommit)$ `in`
r8. $\mathsf{in}(untapch, sr)$.
r9. `if` $\mathsf{checksign}(sr, spk) = \mathsf{true}$
r10. `then let` $(ss^{\mathsf{p}_1}, \ldots, ss^{\mathsf{p}_m}) = \mathsf{getmsg}(sr)$ `in`
r11. $\mathsf{out}(synch, (pk, cmt^{\mathsf{p}_1}, \ldots, cmt^{\mathsf{p}_m}, ss^{\mathsf{p}_1}, \ldots, ss^{\mathsf{p}_m}))$

Figure 13: The process *readinfo*.

the only price in the price list), the process stops (**a10**) and no bidder has bid for any price. In a similar way, the sub-process $checknextb_{\mathsf{b}_i}^{\mathsf{p}_j}$ is used to evaluate the bid of a bidder b_i at price p_j, if there are already some winners before bidder b_i. And the sub-process $checknextbnp_{\mathsf{b}_i}^{\mathsf{p}_j}$ is used to check the next bidder at price p_j, 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.

In the sub-process $checknextb_{\mathsf{b}_i}^{\mathsf{p}_j}$, the auctioneer checks whether the bidder b_i has bid for price p_j (**n1**). If the bidder b_i has bid for p_j, b_i is a winning bidder. The auctioneer publishes the winning bidder b_i and the winning price p_j (**n2**). Note that since there already exists one or more winning bidders, b_i is not the first winner. The auctioneer checks whether the bidder b_i is the last bidder (**n3**). If b_i is the last bidder, the auctioneer has found all winning bidders, thus stops the opening process (**n4**); otherwise, the auctioneer checks the evaluation for the next bidder at the same price (i.e., whether the next bidder is also a winner) (**n5**).

In the sub-process $checknextbnp_{\mathsf{b}_i}^{\mathsf{p}_j}$, the auctioneer first checks whether the bidder b_i has bid for price p_j (**p1**). If the bidder b_i has bid for p_j, b_i is a winner. The auctioneer publishes the bidder b_i and the winning price p_j (**p2**). Since there is no winning bidder found before, b_i is the first winner. Then the auctioneer checks whether the bidder b_i is the last bidder (**p3**). If b_i is the last bidder, bidder b_i is the only winner. Since the auctioneer has found all winners, he stops the opening process (**p4**). Otherwise, the

$checknextb_{\mathsf{b}_i}^{\mathsf{p}_j} :=$

n1. `if` $cmt_{\mathsf{b}_i}^{\mathsf{p}_j} = \mathsf{commit}(ss_{\mathsf{b}_i}^{\mathsf{p}_j}, pk_{\mathsf{b}_i}, \mathsf{M}_{yes})$
n2. `then` $\mathsf{out}(\mathtt{winnerch}, (\mathsf{p}_j, \mathsf{b}_i))$.
n3. `if` $\mathsf{nextbidder}(\mathsf{b}_i) = \bot$
n4. `then` 0
n5. `else` $checknextb_{\mathsf{nextbidder}(\mathsf{b}_i)}^{\mathsf{p}_j}$

Figure 14: The process $checknextb_{\mathsf{b}_i}^{\mathsf{p}_j}$.

$checknextbnp_{\mathsf{b}_i}^{\mathsf{p}_j} :=$

p1. `if` $cmt_{\mathsf{b}_i}^{\mathsf{p}_j} = \mathsf{commit}(ss_{\mathsf{b}_i}^{\mathsf{p}_j}, pk_{\mathsf{b}_i}, \mathsf{M}_{yes})$
p2. `then` $\mathsf{out}(\mathtt{winnerch}, (\mathsf{p}_j, \mathsf{b}_i))$.
p3. `if` $\mathsf{nextbidder}(\mathsf{b}_i) = \bot$
p4. `then` 0
p5. `else` $checknextb_{\mathsf{nextbidder}(\mathsf{b}_i)}^{\mathsf{p}_j}$
p6. `else if` $\mathsf{nextbidder}(\mathsf{b}_i) = \bot$
p7. `then if` $\mathsf{nextprice}(\mathsf{p}_j) = \top$
p8. `then` 0
p9. `else` $checknextbnp_{\mathsf{b}_1}^{\mathsf{nextprice}(\mathsf{p}_j)}$
p10. `else` $checknextbnp_{\mathsf{nextbidder}(\mathsf{b}_i)}^{\mathsf{p}_j}$

Figure 15: The process $checknextbnp_{\mathsf{b}_i}^{\mathsf{p}_j}$.

auctioneer checks whether the next bidder is also a winner (**p5**). Note that since there is already a winner b_i, the auctioneer use the process $checknextb_{\mathsf{nextbidder}(\mathsf{b}_i)}^{\mathsf{p}_j}$. If the bidder b_i has not bid for p_j, the auctioneer checks whether the bidder is the last bidder (**p6**). If b_i is the last bidder, since there is no bidder bid for price p_j before b_i and b_i has not bid for p_j, there is no bidder bid for price p_j. Thus, the auctioneer checks the evaluations for every bidder at the next lower price p_{j-1}. To do so, the auctioneer first checks whether p_{j-1} is the bottom (whether p_j is already the lowest price in the price list) (**p7**). If p_{j-1} is the bottom, since the auctioneer has not found a winner, there does not exist a winner. That is, the auctioneer has checked the evaluations for all bidders at all prices, and no one has bid for any price. Thus, the opening process stops (**p8**). If p_{j-1} is not the bottom, the auctioneer checks the evaluation for the first bidder at the next lower price p_{j-1}. Note that since b_1 is the first bidder checked for price p_{j-1}, there is no winning bidder found before, the process for checking b_1 is

$checknextbnp_{\mathsf{b}_1}^{\mathsf{nextprice}(\mathsf{p}_j)}$ (**p9**). If b_i has not bid for p_j and b_i is not the last bidder, the auctioneer checks the evaluation for the next bidder at the same price (**p10**). Note that since there is no winning bid found, the process is $checknextbnp_{\mathsf{nextbidder}(\mathsf{b}_i)}^{\mathsf{p}_j}$.

5.7. Analysis

After modelling the protocol in the previous section, we formally analyse bidding-price-secrecy and receipt-freeness for bidders. In the AS02 protocol, the winning bid is published, and thus bidding-price-secrecy and receipt-freeness for the winning bidders are 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 and receipt-freeness, we mean only with respect to non-winning bidders.

5.7.1. Bidding-price-secrecy

In general, bidding-price-secrecy can be formalised in two levels: standard bidding-price-secrecy and strong bidding-price-secrecy. Standard bidding-price-secrecy is defined as no matter how the adversary interacts with the protocol, he cannot derive a non-winning bidder's bidding price. Thus, it aims to keep the price secret. However, since the AS02 protocol publishes the bidding price list, the adversary initially knows all the prices. No matter which price a bidder bids for, the bidding price is not a secret to the adversary. Therefore, a bidder's bidding price is not a secret. In fact, what the AS02 protocol aims to protect is the link between bidders and the price he bid, instead of the price itself. Therefore, bidding-price-secrecy of the AS02 protocol is captured by strong bidding-price-secrecy.

Strong bidding-price-secrecy ensures the anonymity of the link between a non-winning bidder and the price he bids for. It is formalised as that the adversary cannot distinguish between the case when a bidder bids for price a and the case when the bidder bids for price c. This property is formally defined in Definition 3.

In the verification, we assume all the participants in the context are honest. Thus, the context $\mathscr{C}_{AS02}[\_]$ (see Figure 16) is defined as the auction process P_{AS02} with a hole (**c9**) instead of two bidder processes, P_{bA} and P_{bB}. Sub-process **c5** to **c8** models the

$\mathscr{C}_{AS02}[\_] :=$

c1. $\quad \nu\, \mathtt{privch}_{\mathtt{b}_1}.\, \nu\, \mathtt{privch}_{\mathtt{b}_2}.\, \cdots.\, \nu\, \mathtt{privch}_{\mathtt{b}_n}.$

c2. $\quad \nu\, \mathtt{privcha}_{\mathtt{b}_1}.\, \nu\, \mathtt{privcha}_{\mathtt{b}_2}.\, \cdots.\, \nu\, \mathtt{privcha}_{\mathtt{b}_n}.$

c3. $\quad \nu\, \mathtt{untapch}_{\mathtt{b}_1}.\, \nu\, \mathtt{untapch}_{\mathtt{b}_2}.\, \cdots.\, \nu\, \mathtt{untapch}_{\mathtt{b}_n}.$

c4. $\quad \nu\, \mathtt{synch}_{\mathtt{b}_1}.\, \nu\, \mathtt{synch}_{\mathtt{b}_2}.\, \cdots.\, \nu\, \mathtt{synch}_{\mathtt{b}_n}.$

c5. $\quad (P_K \mid \ (\mathtt{let}\ \ p_b = p_{\mathtt{b}_1}\ \ \mathtt{in\ let}\ \ \mathit{untapch} = \mathtt{untapch}_{\mathtt{b}_1}\ \ \mathtt{in}$

c6. $\quad\quad \mathtt{let}\ \ \mathit{privch} = \mathtt{privch}_{\mathtt{b}_1}\ \ \mathtt{in\ let}\ \ \mathit{ch} = \mathtt{ch}_{\mathtt{b}_1}\ \ \mathtt{in}\ \ P_b)\ \mid \cdots$

c7. $\quad\quad \mid\ (\mathtt{let}\ \ p_b = p_{\mathtt{b}_{n-2}}\ \ \mathtt{in\ let}\ \ \mathit{untapch} = \mathtt{untapch}_{\mathtt{b}_{n-2}}\ \ \mathtt{in}$

c8. $\quad\quad \mathtt{let}\ \ \mathit{privch} = \mathtt{privch}_{\mathtt{b}_{n-2}}\ \ \mathtt{in\ let}\ \ \mathit{ch} = \mathtt{ch}_{\mathtt{b}_{n-2}}\ \ \mathtt{in}\ \ P_b)\ \mid$

c9. $\quad\quad\quad \_\mid$

c10. $\quad P_a)$

Figure 16: The context $\mathscr{C}_{AS02}[\_]$.

other $n-2$ bidder processes. To verify strong bidding-price-secrecy is to verify the following equivalence:

$$
\begin{array}{rll}
 & \mathscr{C}_{AS02}[& (\text{let } p_b = \mathbf{a} \text{ in let } \mathit{untapch} = \mathtt{untapch}_{\mathtt{b}_A} \text{ in} \\
 & & \text{let } \mathit{privch} = \mathtt{privch}_{\mathtt{b}_A} \text{ in let } \mathit{ch} = \mathtt{ch}_{\mathtt{b}_A} \text{ in } P_b) \mid \\
 & & (\text{let } p_b = \mathbf{d} \text{ in let } \mathit{untapch} = \mathtt{untapch}_{\mathtt{b}_B} \text{ in} \\
 & & \text{let } \mathit{privch} = \mathtt{privch}_{\mathtt{b}_B} \text{ in let } \mathit{ch} = \mathtt{ch}_{\mathtt{b}_B} \text{ in } P_b)] \\
\approx_\ell & \mathscr{C}_{AS02}[& (\text{let } p_b = \mathbf{c} \text{ in let } \mathit{untapch} = \mathtt{untapch}_{\mathtt{b}_A} \text{ in} \\
 & & \text{let } \mathit{privch} = \mathtt{privch}_{\mathtt{b}_A} \text{ in let } \mathit{ch} = \mathtt{ch}_{\mathtt{b}_A} \text{ in } P_b) \mid \\
 & & (\text{let } p_b = \mathbf{d} \text{ in let } \mathit{untapch} = \mathtt{untapch}_{\mathtt{b}_B} \text{ in} \\
 & & \text{let } \mathit{privch} = \mathtt{privch}_{\mathtt{b}_B} \text{ in let } \mathit{ch} = \mathtt{ch}_{\mathtt{b}_B} \text{ in } P_b)]
\end{array}
$$

where a, c, d are from the list $\mathsf{p}_1, \ldots, \mathsf{p}_m$ with $a < d$ and $c < d$.

Normally, strong secrecy properties can be verified, using ProVerif, by querying *noninterf*. Note that ProVerif is sensitive to evaluations of statements in the *if-then-else* constructs [41]. ProVerif reports false attacks when directly querying the following predicate: *noninterf* p_b *among* $\mathsf{p}_1, \ldots, \mathsf{p}_{d-1}$. To be able to check the above equivalence in ProVerif, we use the operation *choice* instead [40], and modify the bidder process by replacing *if-then-else* constructions with choices of a list of variables $vp_1, \ldots, vp_{n-1}$ (see Figure 17). Each variable vp_i corresponds to a price p_i and can be assigned to two possible values, either M_{yes} or M_{no}. If the variable vp_i is assigned

$P_b :=$
b1. $\mathsf{in}(privch, ssk_b)$.
b2. $\nu\ \mathtt{sk}_b.\ \mathsf{out}(ch, \mathsf{sign}((\mathsf{pk}(\mathtt{sk}_b), \mathsf{k}), ssk_b))$.
b3. $\nu\ \mathtt{r}_1.\ \cdots.\ \nu\ \mathtt{r}_m$.
b4. $\mathtt{let}\ \ cmt^{\mathsf{p}_1} = \mathsf{commit}(\mathtt{r}_1, \mathsf{pk}(\mathtt{sk}_b), vp_1)\ \ \mathtt{in}$
$\ldots$
b5. $\mathtt{let}\ \ cmt^{\mathsf{p}_m} = \mathsf{commit}(\mathtt{r}_m, \mathsf{pk}(\mathtt{sk}_b), vp_m)\ \ \mathtt{in}$
b6. $\mathsf{out}(ch, \mathsf{sign}((cmt^{\mathsf{p}_1}, \cdots, cmt^{\mathsf{p}_m}), ssk_b))$.
b7. $\mathsf{out}(untapch, \mathsf{sign}((\mathtt{r}_1, \cdots, \mathtt{r}_m), ssk_b))$

Figure 17: The revised bidder process.

M_{yes}, the bidder bids that price, otherwise, not. Hence, a bidder specifies his bidding price by assigning M_{yes} or M_{no} to each variable $vp_1, \ldots, vp_m$ in his bidding process. For example, in process (P_{bB}) for bidder b_B in the above equivalence, "let $pb = d$ in" shall be replaced by "let $vp_1 = \mathsf{M}_{no}$ in $\ldots$ let $vp_d = \mathsf{M}_{yes}$ in $\ldots$ let $vp_m = \mathsf{M}_{no}$ in". The bidding price in the process (P_{bA}) for a non-winning bidder b_A shall be specified as follows, "let $vp_1 = \mathsf{M}_{no}$ in $\ldots$ let $vp_a = choice[\mathsf{M}_{yes}, \mathsf{M}_{no}]$ in $\ldots$ let $vp_c = choice[\mathsf{M}_{no}, \mathsf{M}_{yes}]$ in $\ldots$ let $vp_m = \mathsf{M}_{no}$ in". The *choice* operations capture the differences between two processes: in the first process, the bidder b_A bids for a ($P_{bA}\{^{a}/_{p_b}\}$), and in the second process, the bidder b_A bids for c ($P_{bA}\{^{c}/_{p_b}\}$). i.e., the non-winning bidder process on the left hand side and the right hand side of the above equivalence, respectively. To query strong bidding-price-secrecy, we specify the bidding price of each bidder in the main process, including the above P_{bB} and P_{bA} (**m6** and **m7** in Figure 18), which captures the above equivalence [7]. This process in Figure 18 is a bi-process due to the *choice* operations in the process (P_{bA}) for bidder b_A. Given the bi-process as input, ProVerif reports a positive result, which means that the above equivalence is satisfied[8]. In this way, we prove that the protocol satisfies strong bidding-price-secrecy.

5.7.2. Receipt-freeness

Receipt-freeness is formally defined in Definition 6. To prove receipt-freeness, we need to find a process P_f which satisfies both equivalences in the definition of receipt-

[7] The '$\cdots$' at the beginning of **m6, m7, m8** represents other bidders.
[8] The revised ProVerif code is available at `http://satoss.uni.lu/projects/epriv`.

$P_{AS02} :=$

m1. $\nu\ \mathtt{privch}_{\mathtt{b}_1}.\ \nu\ \mathtt{privch}_{\mathtt{b}_2}.\ \cdots.\ \nu\ \mathtt{privch}_{\mathtt{b}_n}.$

m2. $\nu\ \mathtt{privcha}_{\mathtt{b}_1}.\ \nu\ \mathtt{privcha}_{\mathtt{b}_2}.\ \cdots.\ \nu\ \mathtt{privcha}_{\mathtt{b}_n}.$

m3. $\nu\ \mathtt{untapch}_{\mathtt{b}_1}.\ \nu\ \mathtt{untapch}_{\mathtt{b}_2}.\ \cdots.\ \nu\ \mathtt{untapch}_{\mathtt{b}_n}.$

m4. $\nu\ \mathtt{synch}_{\mathtt{b}_1}.\ \nu\ \mathtt{synch}_{\mathtt{b}_2}.\ \cdots.\ \nu\ \mathtt{synch}_{\mathtt{b}_n}.$

m5. $(P_K \mid$

m6. $\cdots \mid (\mathtt{let}\ \ vp_1 = \mathsf{M}_{no}\ \mathtt{in}\ \ldots\ \mathtt{let}\ \ vp_d = \mathsf{M}_{yes}\ \mathtt{in}\ \ldots$
$\mathtt{let}\ \ vp_m = \mathsf{M}_{no}\ \mathtt{in}\ \mathtt{let}\ \ untapch = \mathtt{untapch}_{\mathtt{b}_B}\ \mathtt{in}$
$\mathtt{let}\ \ privch = \mathtt{privch}_{\mathtt{b}_B}\ \mathtt{in}\ \mathtt{let}\ \ ch = \mathtt{ch}_{\mathtt{b}_B}\ \mathtt{in}\ \ P_b) \mid$

m7. $\cdots \mid (\mathtt{let}\ \ vp_1 = \mathsf{M}_{no}\ \mathtt{in}\ \ldots\ \mathtt{let}\ \ vp_a = choice[\mathsf{M}_{yes}, \mathsf{M}_{no}]\ \mathtt{in}\ \ldots$
$\mathtt{let}\ \ vp_c = choice[\mathsf{M}_{no}, \mathsf{M}_{yes}]\ \mathtt{in}\ \ldots\ \mathtt{let}\ \ vp_m = \mathsf{M}_{no}\ \mathtt{in}$
$\mathtt{let}\ \ untapch = \mathtt{untapch}_{\mathtt{b}_A}\ \mathtt{in}$
$\mathtt{let}\ \ privch = \mathtt{privch}_{\mathtt{b}_A}\ \mathtt{in}\ \mathtt{let}\ \ ch = \mathtt{ch}_{\mathtt{b}_A}\ \mathtt{in}\ \ P_b) \mid$

m8. $\cdots \mid P_a)$

Figure 18: The bi-process.

freeness:

eq1:

$$\begin{aligned} & \text{let } untapch = \mathtt{untapch}_{\mathtt{b}_A} \text{ in} \\ & \text{let } privch = \mathtt{privch}_{\mathtt{b}_A} \text{ in let } ch = \mathtt{ch}_{\mathtt{b}_A} \text{ in } P_f{}^{\backslash \mathsf{out}(\mathtt{chc},\cdot)} \\ \approx_\ell\ & \text{let } p_b = \mathbf{c} \text{ in let } untapch = \mathtt{untapch}_{\mathtt{b}_A} \text{ in} \\ & \text{let } privch = \mathtt{privch}_{\mathtt{b}_A} \text{ in let } ch = \mathtt{ch}_{\mathtt{b}_A} \text{ in } P_b, \end{aligned}$$

eq2:

$$\begin{aligned} & \mathscr{C}_{AS02}[\ \ (\text{let } p_b = \mathbf{a} \text{ in let } untapch = \mathtt{untapch}_{\mathtt{b}_A} \text{in} \\ & \qquad \text{let } privch = \mathtt{privch}_{\mathtt{b}_A} \text{ in let } ch = \mathtt{ch}_{\mathtt{b}_A} \text{ in } P_b)^{\mathtt{chc}} \mid \\ & \qquad (\text{let } p_b = \mathbf{d} \text{ in let } untapch = \mathtt{untapch}_{\mathtt{b}_B} \text{ in} \\ & \qquad \text{let } privch = \mathtt{privch}_{\mathtt{b}_B} \text{ in let } ch = \mathtt{ch}_{\mathtt{b}_B} \text{ in } P_b)] \\ \approx_\ell\ & \mathscr{C}_{AS02}[\ \ P_f \mid (\text{let } p_b = \mathbf{d} \text{ in let } untapch = \mathtt{untapch}_{\mathtt{b}_B} \text{ in} \\ & \qquad \text{let } privch = \mathtt{privch}_{\mathtt{b}_B} \text{ in let } ch = \mathtt{ch}_{\mathtt{b}_B} \text{ in } P_b)] \end{aligned}$$

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

According to the properties of chameleon bit-commitments, 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,

$P_f :=$

f1. $\mathsf{in}(\mathit{privch}, \mathit{ssk}_b).\ \mathsf{out}(\mathtt{chc}, \mathit{ssk}_b)).$

f2. $\nu\ \mathtt{sk}_b.\ \mathsf{out}(\mathtt{chc}, \mathtt{sk}_b).$

f3. $\mathsf{out}(\mathit{ch}, \mathsf{sign}((\mathsf{pk}(\mathtt{sk}_b), \mathsf{k}), \mathit{ssk}_b)).$

f4. $\nu\ \mathtt{r}_1.\ \cdots.\ \nu\ \mathtt{r}_a.\ \cdots.\ \nu\ \mathtt{r}_c.\ \cdots.\ \nu\ \mathtt{r}_m.$

f5. $\mathsf{out}(\mathtt{chc}, (\mathtt{r}_1, \ldots, \mathsf{f}(\mathtt{r}_a), \ldots, \mathsf{f}(\mathtt{r}_c), \ldots, \mathtt{r}_m)).$

f6. $\mathtt{let}\ \ \mathit{cmt}^{\mathsf{p}_1} = \mathsf{commit}(\mathtt{r}_1, \mathsf{pk}(\mathtt{sk}_b), \mathsf{M}_{no})\ \ \mathtt{in}$

f7. ...

f8. $\mathtt{let}\ \ \mathit{cmt}^{\mathsf{p}_a} = \mathsf{commit}(\mathtt{r}_a, \mathsf{pk}(\mathtt{sk}_b), \mathsf{M}_{no})\ \ \mathtt{in}$

f9. ...

f10. $\mathtt{let}\ \ \mathit{cmt}^{\mathsf{p}_c} = \mathsf{commit}(\mathtt{r}_c, \mathsf{pk}(\mathtt{sk}_b), \mathsf{M}_{yes})\ \ \mathtt{in}$

f11. ...

f12. $\mathtt{let}\ \ \mathit{cmt}^{\mathsf{p}_m} = \mathsf{commit}(\mathtt{r}_m, \mathsf{pk}(\mathtt{sk}_b), \mathsf{M}_{no})\ \ \mathtt{in}$

f13. $\mathsf{out}(\mathit{ch}, \mathsf{sign}((\mathit{cmt}^{\mathsf{p}_1}, \ldots, \mathit{cmt}^{\mathsf{p}_m}), \mathit{ssk}_b)).$

f14. $\mathsf{out}(\mathit{untapch}, \mathsf{sign}((\mathtt{r}_1, \ldots, \mathtt{r}_a, \ldots, \mathtt{r}_c, \ldots, \mathtt{r}_m), \mathit{ssk}_b))$

Figure 19: The process P_f.

while the auctioneer opens the same bit-commitments as the bidder bids for price c, using the secret seeds the auctioneer received through an untappable channel. Thus, the bidder could execute the process P_f as shown in Figure 19 to lie to the adversary. The bidder in this process communicates with the adversary through channel chc, sending the adversary his secret signature key ssk_b (**f1**) and his secret key $\mathtt{sk}_b$ (**f2**). Later the bidder sends to the auctioneer $\mathtt{r}_1, \ldots, \mathtt{r}_m$ through an untappable channel (**f14**), and sends to the adversary the same list except changing $\mathtt{r}_a$ and $\mathtt{r}_c$ to $\mathsf{f}(\mathtt{r}_a)$ and $\mathsf{f}(\mathtt{r}_c)$, respectively (**f5**). The untappable channel ensures the adversary cannot learn anything about the differences.

To prove the first equivalence, we can simply consider ${P_f}^{\setminus \mathsf{out}(\mathtt{chc}, \cdot)}$ as process P_f without communication on the channel chc. Since the process ${P_f}^{\setminus \mathsf{out}(\mathtt{chc}, \cdot)}$ works exactly the same as the process $P_b\{c/p_b\}$, the first equivalence (**eq1**) is satisfied. To show the second equivalence (**eq2**), we need to consider all the transitions of each side [9]. 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 ignore the out-

[9]The satisfaction of **eq2** is supported by ProVerif as well. ProVerif code is available at http://satoss.uni.lu/projects/epriv.

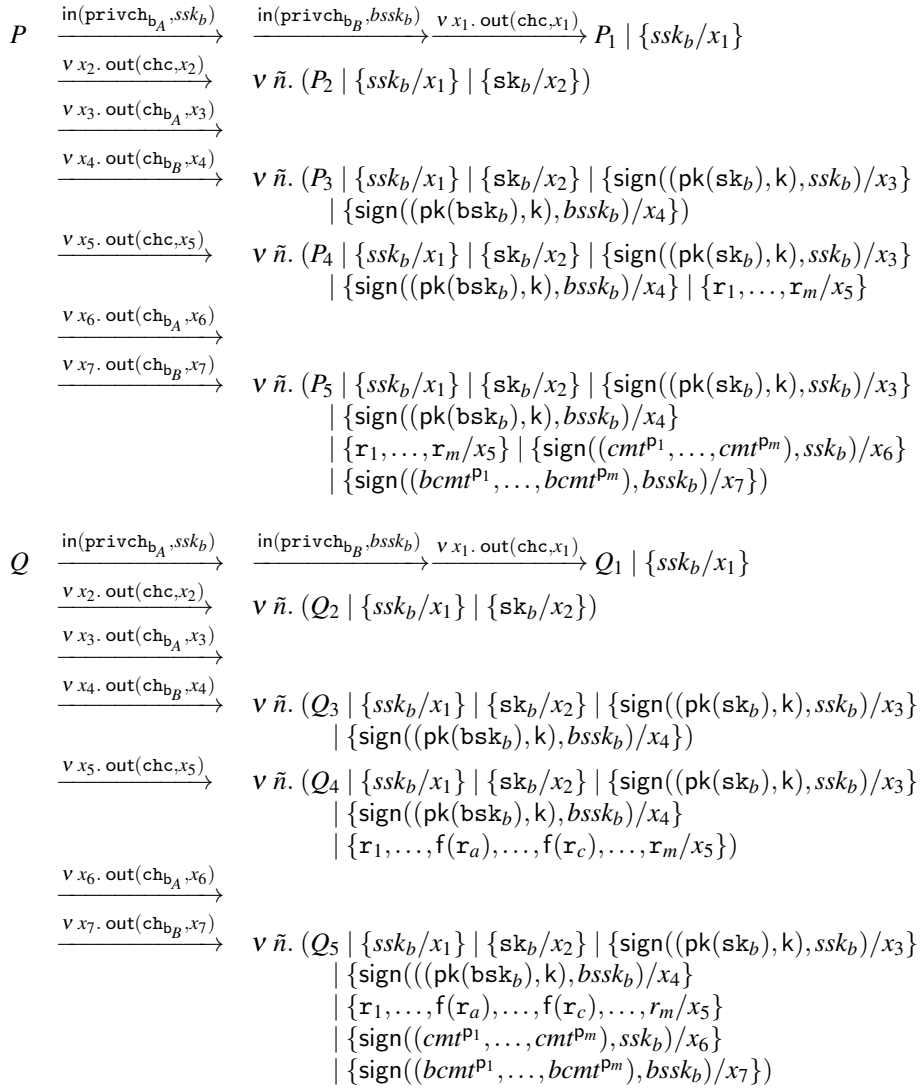

Figure 20: A brief proof of receipt-freeness in AS02.

puts in the process P_K and those bidder processes in the context. During the bidding phase the auctioneer process only reads information and synchronises on the private channels $\mathtt{synch}_{\mathtt{b}_1}, \ldots, \mathtt{synch}_{\mathtt{b}_n}$. There is no output on public channels in the auctioneer process. We denote the sequence of names $\mathtt{sk}_b, \mathtt{r}_1, \ldots, \mathtt{r}_m, \mathtt{bsk}_b, \mathtt{br}_1, \ldots, \mathtt{br}_m$ by $\tilde{n}$, i.e., $\mathtt{sk}_b, \mathtt{r}_1, \ldots, \mathtt{r}_m$ are names in the non-winning bidder processes P_{bA} and P_f, and

$\mathtt{bsk}_b, \mathtt{br}_1, \ldots, \mathtt{br}_m$ are names in the winning bidder process P_{bB}. After the key distribution, we want to see whether the behaviour of the process $P_{bA}\{a/p_b\}^{\mathtt{chc}} \mid P_{bB}\{d/p_b\}$ is observationally equivalent to $P_f \mid P_{bB}\{d/p_b\}$ $\big(P_{bA}\{a/p_b\}^{\mathtt{chc}} := (\text{let } p_b = \mathbf{a} \text{ in let } \mathit{untapch} = \mathtt{untapch}_{\mathtt{b}_A} \text{in let } \mathit{privch} = \mathtt{privch}_{\mathtt{b}_A} \text{ in let } \mathit{ch} = \mathtt{ch}_{\mathtt{b}_A} \text{ in } P_b)^{\mathtt{chc}}$, and $P_{bB}\{d/p_b\} := (\text{let } p_b = \mathbf{d} \text{ in let } \mathit{untapch} = \mathtt{untapch}_{\mathtt{b}_B} \text{ in let } \mathit{privch} = \mathtt{privch}_{\mathtt{b}_B} \text{ in let } \mathit{ch} = \mathtt{ch}_{\mathtt{b}_B} \text{ in } P_b)\big)$. 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 $P = P_{bA}\{a/p_b\}^{\mathtt{chc}} \mid P_{bB}\{d/p_b\}$ and $Q = P_f \mid P_{bB}\{d/p_b\}$, we have their labelled transitions as shown in Figure 20.

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 of the bidder. The only functions the adversary can use are getmsg and open. By applying these two functions, the adversary can get extra terms, the public key of the bidder represented as $x_{msg} = \mathsf{getmsg}(x_3, x_1)$ and a series of opened messages from bit-commitments. 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, $P_{bA}\{a/p_b\}$ bids for price a. The adversary opens the commitments $\mathit{cmt}^{p_a} = \mathsf{commit}(r_a, \mathsf{pk}(\mathit{sk}_b), \mathsf{M}_{yes})$ and $\mathit{cmt}^{p_c} = \mathsf{commit}(r_c, \mathsf{pk}(\mathit{sk}_b), \mathsf{M}_{no})$ as follows:

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

For the process Q, the process P_f 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 Section 5.6), the adversary opens $\mathit{cmt}^{p_a} = \mathsf{commit}(r_a, \mathsf{pk}(\mathit{sk}_b), \mathsf{M}_{no}) = \mathsf{commit}(\mathsf{f}(r_a), \mathsf{pk}(\mathit{sk}_b), \mathsf{M}_{yes})$ and opens $\mathit{cmt}^{p_c} = \mathsf{commit}(r_c, \mathsf{pk}(\mathit{sk}_b), \mathsf{M}_{yes}) = \mathsf{commit}(\mathsf{f}(r_c), \mathsf{pk}(\mathit{sk}_b), \mathsf{M}_{no})$ as follows:

$$\mathsf{open}(\mathit{cmt}^{p_a}, \mathsf{f}(r_a), \mathsf{pk}(\mathit{sk}_b)) = \mathsf{M}_{yes} \qquad \mathsf{open}(\mathit{cmt}^{p_c}, \mathsf{f}(r_c), \mathsf{pk}(\mathit{sk}_b)) = \mathsf{M}_{no}$$

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 bids. Therefore, non-winning bids are not revealed. Since we have assumed the auctioneer is honest, the information that the auctioneer process reveals is the opened bit-commitments of all bidders at prices no lower than the winning price, and the winning bidders. 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 $P_{b_B}\{d/p_b\}$) on both sides of the equivalence, the bid made by the bidder b_A will never be published, hence the information the auctioneer process reveals is the same on both sides. Now, we can conclude that the protocol satisfies receipt-freeness.

6. Case study: the HRM14 protocol

HRM14 is a also seal-bid auction protocol designed with receipt-freeness in mind. Similar to the AS02 protocol, the HRM14 protocol allows a bidder to lie to the adversary by providing fake bids. Unlike the AS02 protocol, which depends on the use of chameleon-bit-commitments and untappble channels, the HRM14 protocol uses *Plan-Ahead Deniable Encryption* (PDE) to achieve the same goal.

6.1. Description of the protocol

The protocol involves m bidders, k sealers (authorities who share the same public key and execute sealing operations together) and an auctioneer. It works as follows: each bidder encrypts his bid ('yes' or 'no') on each price with the public key of the auctioneer and the public key of the sealer. All bids of a bidder form a bidding vector. The bidder encrypts his bidding vector using PDE with the public key of the *Coercing Resistant Mix* (CRM) and sends the resulting ciphertext to the CRM. The CRM decrypts the ciphertext and obtains the bidding vector. The CRM collects all bidders' bidding vectors, permutes them and finally sends them to a group of sealers (the size of the group is more than a threshold t) via an anonymous channel. The sealers nullify their public keys in each bid, and seal each bid with two nonces. One of the nonce is used to

blind the bid, and the other is used to ensure the sealed message is not guessable. The sealed bids are published to the Bulletin Board (BB) and read by the auctioneer. The auctioneer signs each bidder's bids using a specific scheme and publishes the signature to the Bulletin Board, so that the bidders can check whether his bids are counted. Finally, the sealers and the auctioneer together open the bid from the highest price to the lowest price. If the winning bid is found, the opening procedure stops. The main steps are shown in Figure 21.

Intuitively, due to the use of PDE, a coerced bidder can prepare fake bidding vectors with a different bidding price, and the adversary cannot verify which bidding vectors (real or fake) are used, while the CRM can always get the real ones.

6.2. Settings

Bidding price. The protocol predefines a price list, represented as d ordered vectors, $ve_0, \ldots, ve_{d-1}$ from low to high, where each vector ve_k consists of 10 numbers from 0 to 9, denoted as p_{k0}, ..., p_{k9}, i.e., the price list is p_{00}, ..., p_{09},..., $\mathsf{p}_{(d-1)0}$,..., $\mathsf{p}_{(d-1)9}$, representing the price from 0 to $10^d - 1$. The bidder bids 'yes' or 'no' on each p_{ji} ($0 \leq i \leq 9$, $0 \leq j \leq d-1$). On each vector only one number is marked with 'yes'. The bidding price b is calculated as follows: for each p_{ji} in the list, if it is marked with 'no', bidding price remains unchanged ($b = b$), if it is marked with 'yes', $b = b + j * 10^i$.

'yes' and 'no' marks. Like the AS02 protocol, the 'no' mark is a constant in the HRM14 protocol, in particular the 'no' mark is the number 1 in the HRM14 protocol. Unlike the AS02 where the 'yes' is a constant, in the HRM14 protocol, the 'yes' mark is calculated as $\hat{r}_{i,(k,j)} G_i^{r_{i,(k,j)}}$ where $G_i = g_y^{x_{B_i}}$. It is not clear what g_y is in the paper. Since the private key of bidder B_i is x_{B_i} and the corresponding public key is $g^{x_{B_i}}$, we assume g_y is similar to g in the public key, because $G_i = g_y^{x_{B_i}}$ share the same patter as the public key $g^{x_{B_i}}$. Thus, we consider G_i as a special public key of the bidder B_i. Unlike the normal public key which is assumed to be publicly known and is used to identify a bidder, this special public key is not revealed, such that the adversary cannot use it to identify the bidder. Otherwise privacy is trivially broken. However, given a bidder's secret key, the adversary can construct and thus verify the special public key of the bidder. With the above assumptions, the 'yes' mark is modelled as a function with

Figure 21: The HRM14 protocol.

two nonces and the special public key of the bidder as parameters, formally '*fun* $\mathsf{B}/3$.'. Note that unlike in the AS02 protocol, in this protocol, the 'yes' marks differ for each bid.

6.3. Cryptographic primitives

6.3.1. PDE

The PDE enables a normal probabilistic encryption of a message m with a public key pk (known by the adversary) using a random number r, denoted as $\mathsf{denc}(r, pk, m)$. In the normal probabilistic encryption, the adversary can coerce for m and r, and thus be able to verify the encrypted message m. However, in PDE, given a fake message mf, it allows a user to generate a fake random number rf, such that $\mathsf{denc}(r, pk, m) = \mathsf{denc}(rf, pk, mf)$.

This can be modelled in a similar way as chameleon-bit-commitments. The difference is that the chameleon-bit-commitments are opened with the random number, whereas the PDE is opened with the designated receiver's secret key.

fun	$\mathsf{denc}/3$.
fun	$\mathsf{fake}/4$.
reduc	$\mathsf{ddec}(\mathsf{denc}(r, \mathsf{pk}(k), m), k) = m$.
equation	$\mathsf{denc}(r, k, m) = \mathsf{denc}(\mathsf{fake}(r, k, m, mf), k, mf)$.

The function symbol denc models the PDE encryption which takes three parameters (r, pk, m) as inputs and returns a deniable encryption as output. The function fake produces the fake random, given the real random number r, the public key k, the real message m and the fake message mf as parameters. The two equations ensure that that the designated receiver of the deniable cipher always interprets the plaintext in only one way – the real one. The adversary who does not have the secret key of the designated receiver cannot get the real plaintext. Given the fake random, the adversary may get a fake message mf. Due to the last equation, the adversary cannot distinguish whether the coerced random number and the plaintext are genuine or not.

Remark on ambiguities of PDE. In the original paper [18], the PDE outputs two types

of encryptions: one type with three parameters, $c = Enc^{-}(m_t, pk, r_t)$, and the other with four parameters, $c_d = Enc^{m_f}(m_t, pk, r_t)$, where m_t and r_t are real message and real random and m_f is the fake message. Both of them is decrypted as m_t. The fake random r_f is produced by a operation applied on c_d, m_t and m_f, such that $Enc^{-}(m_t, pk, r_t) = Enc^{-}(m_f, pk, r_f)$. However, it is not clear whether c equals c_d. Following the intuition of PDE, we consider them as the same (or 'look alike' [18]). More importantly, if c is not the same as c_d, the adversary would be able to distinguish whether the encryption used is a three parametrised version or four parametrised version, by coercing for m_t and r_t. When noticing that the four parametrised version is used, the adversary knows the user is trying to cheat. By additional coercing for m_f and r_f, the adversary can verify which message, m_f or m_t, is the genuine one. Hence, c needs to be the same as c_d.

6.3.2. Bidding encryption

Each bid is encrypted with both the public key of the auctioneer pk_A and the public key of the sealers pk_S (All sealers have the same public key). Only with the sum of both the secret key of the auctioneer and the secret key of the sealers, the ciphertext can be decrypted. Providing the secret key of the auctioneer or the secret key of the sealers, its corresponding public key component in the encryption can be nullified. Symbolically, it works the same as cascaded encryptions – first encrypting the bid with pk_A and then encrypting with pk_S (or the other way), but using the same random number. Hence, it is captured by the typical probabilistic encryption functions and equations.

$$\begin{array}{ll} \textit{fun} & \mathsf{penc}/3. \\ \textit{reduc} & \mathsf{pdec}(\mathsf{enc}(a, \mathsf{pk}(x), m), x) = m. \end{array}$$

An encrypted bid is thus $\mathsf{penc}(a, pk_S, \mathsf{penc}(a, pk_A, v))$, where v is either 'yes' or 'no', and a is a nonce.

6.3.3. Sealing operation

For each encrypted bid $\mathsf{penc}(a, pk_S, \mathsf{penc}(a, pk_A, v))$, after nullifying the sealers' public key component (decrypting using sk_S which remains $\mathsf{penc}(a, pk_A, v)$), the sealers

seal it with two nonces r and rs. The result of the sealing operation is the same as encrypting $v * rs$ with pk_A using nonce $a + r$, i.e, the message v is blinded. Thus the sealing operation is captured by the following functions.

$$
\begin{array}{ll}
\textit{fun} & \mathsf{seal}/4. \\
\textit{fun} & \mathsf{blind}/2. \\
\textit{equation} & \mathsf{seal}(\mathsf{penc}(a, \mathsf{pk}(sk_S), \mathsf{penc}(a, pk, m)), r, rs, sk_S) = \\
 & \qquad\qquad \mathsf{penc}(\mathsf{blind}(a, r), pk, \mathsf{blind}(rs, m)).
\end{array}
$$

6.3.4. Bid verification

The bid verification is performed on each price vector, i.e., 10 bids. For each price vector, the sealers publish the the response-vector – multiplication of the rs's. This is modelled as a hash function with $t * 10$ arities – the 10 rs's in the vector from all t sealers. Assume that from the multiplication result, the adversary cannot deduce any rs. That is, the sealers provide partial information on the blinding factor (rs's). Then the auctioneer provides partial information on his secret key and the blinded nonces ($\mathsf{blind}(a, r)$'s). The bidder knows the nonces (a's) used in encrypting the bids. With these partial information, the bidder can verify whether his bids in a vector is correctly computed using an equation. For simplicity, we assume only one sealer. And this bid verification can be formally captured as

$$
\begin{array}{ll}
\textit{fun} & \mathsf{hash}/10. \\
\textit{fun} & \mathsf{sign}/2. \\
\textit{fun} & \mathsf{combine}/11. \\
\textit{equation} & \mathsf{sign}((\mathsf{penc}(\mathsf{blind}(a_0, r_0), pk, \mathsf{blind}(rs_0, m_0)), \ldots, \\
 & \qquad \mathsf{penc}(\mathsf{blind}(a_9, r_9), pk, \mathsf{blind}(rs_9, m_9))), sk) \\
 & = \mathsf{combine}(\mathsf{blind}(a_0, r_0), \ldots, \mathsf{blind}(a_9, r_9), sk). \qquad \textbf{et3}
\end{array}
$$

$$
\begin{array}{ll}
\textit{reduc} & \mathsf{verify}((\mathsf{penc}(\mathsf{blind}(a_0, r_0), pk, \mathsf{blind}(rs_0, m_0)), \ldots, \\
 & \qquad \mathsf{penc}(\mathsf{blind}(a_9, r_9), pk, \mathsf{blind}(rs_9, m_9))), \\
 & \qquad \mathsf{combine}(\mathsf{blind}(a_0, r_0), \ldots, \mathsf{blind}(a_9, r_9), sk), \\
 & \qquad (a_0, \ldots, a_9), \mathsf{hash}(rs_0, \ldots, rs_9)) = \mathsf{true}. \qquad \textbf{et4}
\end{array}
$$

Remark on formula of sealing and bid verification. In the original paper [18], by applying Algorithm 2, a bid from bidder i at price represented by element (k, j) in the price list is sealed as $(X_{S_t i,(k,j)}, Y_{S_t i,(k,j)})$, where $S_1, \ldots, S_t$ are the sealers. We observe that the calculation of $Y_{S_t i,(k,j)}$ in the Algorithm 2 differs from the one in the appendix. In Algorithm 2,

$$Y_{S_l i,(k,j)} = \hat{r}_{S_l i,(k,j)} \cdot h_A^{r_{S_l i,(k,j)}} \cdot h_{S/S_1,\ldots,S_l}^{r_{S_l i,(k,j)}} \cdot (X_{S_{l-1} i,(k,d)})^{-x_{S_l}} \cdot Y_{S_{l-1} i,(k,d)}$$

whereas in the appendix of [18]

$$Y_{S_l i,(k,j)} = \hat{r}_{S_l i,(k,j)} \cdot h_A^{r_{S_l i,(k,j)}} \cdot h_{S/S_1,\ldots,S_l}^{r_{S_l i,(k,j)}} \cdot (X_{S_{l-1} i,(k,j)})^{-x_{S_l}} \cdot Y_{S_{l-1} i,(k,j)} \tag{1}$$

We suspect that there is a typo in Algorithm 2, and use the one in the appendix (formula (1)), since the following equation (2) is proved using formula (1).

Later, the sealers publish a response-vector $R_{S_t i,k}$ for each vector k, and the auctioneer publishes $\mathbb{X}_{i,k}$.

$$R_{S_t i,k} = \prod_{j=0}^{9} \prod_{v=1}^{t} \hat{r}_{S_v i,(k,j)} \qquad \mathbb{X}_{i,k} = (\prod_{j=0}^{9} X_{S_t i,(k,j)})^{x_A} = h_A^{\sum_{j=0}^{9} (r_{i,(k,j)} + \sum_{v=1}^{t} r_{s_v i,(k,j)})}$$

The bidder verifies whether the following equation (2) holds:

$$\prod_{j=0}^{9} Y_{S_t i,(k,j)} = R_{S_t i,k} \cdot \mathbb{X}_{i,k} \cdot \prod_{j=0}^{9} G_{i,(k,j)} \tag{2}$$

where $G_{i,(k,j)}$ is the 'yes' mark computed by using the public key of the bidder i and the nonce used to generated the corresponding bid on element (k, j) in the price list.

The response-vector $R_{S_t i,k}$ is modelled by function hash. The signature $\mathbb{X}_{i,k}$ is modelled by function sign, combine and the equation **et3**, the equation (2) is formally captured by the reduction **et4**.

6.4. Modelling

As shown in Figure 22, the protocol has a private channel `privch` between the CRM and the sealers. The auctioneer has a secret key $\mathtt{sk}_A$ and a corresponding public key pk_A and follows the behaviour of process P_A (Figure 26). Since only more than a threshold t sealers together can nullify the sealers' public key and perform the sealing operation, we assume the sealers' operations are honest, and abstract the sealers as one honest sealer. Hence, the sealer has a secret key $\mathtt{sk}_S$ and a corresponding public key pk_S, and follows the behaviour modelled in process P_S (Figure 25). The CRM has a secret key $\mathtt{sk}_{CRM}$ and a corresponding public key pk_{CRM}; and its behaviour is modelled in process P_{CRM} (Figure 24). Each bidder b_i has a bidding price, represented as a vector $p_0^{\mathsf{b}_i}, \ldots, p_{d-1}^{\mathsf{b}_i}$, where each $p_j^{\mathsf{b}_i}$ is a number between 0 and 9. Lines **m4-m6** model that there are m bidders. The process $\mathtt{out}(\mathtt{ch}, pk_A) \mid \mathtt{out}(\mathtt{ch}, pk_S) \mid \mathtt{out}(\mathtt{ch}, pk_{CRM})$ ensures that the adversary knows the public keys.

$P_{HRM14} :=$
m1. $\nu\ \mathtt{privch}.\ \nu\ \mathtt{sk}_A.\ \nu\ \mathtt{sk}_S.\ \nu\ \mathtt{sk}_{CRM}.\mathtt{let}\ \ pk_A = \mathsf{pk}(\mathtt{sk}_A)\ \ \mathtt{in}$
m2. $\mathtt{let}\ \ pk_S = \mathsf{pk}(\mathtt{sk}_S)\ \ \mathtt{in}\ \ \mathtt{let}\ \ pk_{CRM} = \mathsf{pk}(\mathtt{sk}_{CRM})\ \ \mathtt{in}$
m3. $(\mathtt{out}(\mathtt{ch}, pk_A) \mid \mathtt{out}(\mathtt{ch}, pk_S) \mid \mathtt{out}(\mathtt{ch}, pk_{CRM}) \mid$
m4. $\mathtt{let}\ \ p_0^b = p_0^{\mathsf{b}_1}\ \ \mathtt{in}\ \ \ldots\mathtt{let}\ \ p_{d-1}^b = p_{d-1}^{\mathsf{b}_1}\ \ \mathtt{in}\ \ P_B \mid$
m5. $\cdots \mid$
m6. $\mathtt{let}\ \ p_0^b = p_0^{\mathsf{b}_m}\ \ \mathtt{in}\ \ \ldots\mathtt{let}\ \ p_{d-1}^b = p_{d-1}^{\mathsf{b}_m}\ \ \mathtt{in}\ \ P_B \mid$
m7. $P_{CRM} \mid P_S \mid P_A)$

Figure 22: The HRM14 main process.

A bidder's behaviour is shown in Figure 23. Each bidder has a private key $\mathtt{sk}_B$ and a corresponding public key pk_B (**b1**), which is used to calculate the 'yes' marks. The bidder first generates a nonce $\mathtt{a}_{kj}$ for each element in the price list, which is used for encrypting his bids and a nonce `r` for generating the deniable encryption (**b2**). Then according to the bidding price, the bidder chooses the branch to calculate his bids. For instance, lines **b3 -b14** model the bidder calculating the bidding vector and sending it out, when the bidding price is 0. Line **b15** models the bidder's behaviour when the bidding price is 1 and line **b16** models the bidder behaviour when the bidding price is 2, and finally, lines **b17-b28** model the bidder's behaviour when the bidding price is

$P_B :=$

b1. $\nu\ \mathtt{sk}_B.\ \mathtt{let}\ \ pk_B = \mathsf{pk}(\mathtt{sk}_B)\ \ \mathtt{in}$

b2. $\nu\ \mathtt{r}.\ \nu\ \mathtt{a}_{00}.\ \cdots.\ \nu\ \mathtt{a}_{09}.\ \nu\ \mathtt{a}_{10}.\ \cdots.\ \nu\ \mathtt{a}_{19}.\ \cdots.\ \nu\ \mathtt{a}_{(d-1)0}.\ \cdots.\ \nu\ \mathtt{a}_{(d-1)9}.$

b3. $\mathtt{if}\ \ p_0^b = \mathsf{p}_{00} \wedge p_1^b = \mathsf{p}_{10}\ \ \wedge \ldots \wedge p_{d-1}^b = \mathsf{p}_{(d-1)0}\ \ \mathtt{then}$

b4. $\big(\nu \mathtt{r}_0.$

b5. $\mathtt{let}\ \ b^{\mathsf{p}00} = \mathsf{penc}(\mathtt{a}_{00}, pk_S, \mathsf{penc}(\mathtt{a}_{00}, pk_A, \mathsf{B}(\mathtt{r}_0, pk_B, \mathtt{a}_{00})))\ \ \mathtt{in}$

b6. $\mathtt{let}\ \ b^{\mathsf{p}01} = \mathsf{penc}(\mathtt{a}_{01}, pk_S, \mathsf{penc}(\mathtt{a}_{01}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

$\ldots$

b7. $\mathtt{let}\ \ b^{\mathsf{p}09} = \mathsf{penc}(\mathtt{a}_{09}, pk_S, \mathsf{penc}(\mathtt{a}_{09}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

b8. $\mathtt{let}\ \ ve_0 = (b^{\mathsf{p}00}, \ldots, b^{\mathsf{p}09})\ \ \mathtt{in}$

$\ldots$

b9. $\nu \mathtt{r}_{d-1}.$

b10. $\mathtt{let}\ \ b^{\mathsf{p}(d-1)0} = \mathsf{penc}(\mathtt{a}_{(d-1)0}, pk_S, \mathsf{penc}(\mathtt{a}_{(d-1)0}, pk_A, \mathsf{B}(\mathtt{r}_{d-1}, pk_B, \mathtt{a}_{(d-1)0})))\ \ \mathtt{in}$

$\ldots$

b11. $\mathtt{let}\ \ b^{\mathsf{p}(d-1)9} = \mathsf{penc}(\mathtt{a}_{(d-1)9}, pk_S, \mathsf{penc}(\mathtt{a}_{(d-1)9}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

b12. $\mathtt{let}\ \ ve_{d-1} = (b^{\mathsf{p}(d-1)0}, \ldots, b^{\mathsf{p}(d-1)9})\ \ \mathtt{in}$

b13. $\mathtt{let}\ \ ve = (ve_1, \ldots, ve_{d-1})\ \ \mathtt{in}$

b14. $\mathsf{out}(\mathtt{ch}, \mathsf{denc}(\mathtt{r}, pk_{CRM}, ve))).\ P_{check}\big)$

b15. $\mathtt{else\ if}\ \ p_0^b = \mathsf{p}_{01}\ \ \wedge\ \ p_1^b = \mathsf{p}_{10}\ \ \wedge \ldots \wedge p_{d-1}^b = \mathsf{p}_{(d-1)0}\ \ \mathtt{then}$

$\ldots$

b16. $\mathtt{else\ if}\ \ p_0^b = \mathsf{p}_{02}\ \ \wedge\ \ p_1^b = \mathsf{p}_{10}\ \ \wedge \ldots \wedge p_{d-1}^b = \mathsf{p}_{(d-1)0}\ \ \mathtt{then}$

$\ldots$

b17. $\mathtt{else\ if}\ \ p_0^b = \mathsf{p}_{09}\ \ \wedge\ \ p_1^b = \mathsf{p}_{19}\ \ \wedge \ldots \wedge p_{d-1}^b = \mathsf{p}_{(d-1)9}\ \ \mathtt{then}$

b18. $\big(\nu \mathtt{r}_0.$

b19. $\mathtt{let}\ \ b^{\mathsf{p}00} = \mathsf{penc}(\mathtt{a}_{00}, pk_S, \mathsf{penc}(\mathtt{a}_{00}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

b20. $\mathtt{let}\ \ b^{\mathsf{p}01} = \mathsf{penc}(\mathtt{a}_{01}, pk_S, \mathsf{penc}(\mathtt{a}_{01}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

$\ldots$

b21. $\mathtt{let}\ \ b^{\mathsf{p}09} = \mathsf{penc}(\mathtt{a}_{09}, pk_S, \mathsf{penc}(\mathtt{a}_{09}, pk_A, \mathsf{B}(\mathtt{r}_0, pk_B, \mathtt{a}_{09})))\ \ \mathtt{in}$

b22. $\mathtt{let}\ \ ve_0 = (b^{\mathsf{p}00}, \ldots, b^{\mathsf{p}09})\ \ \mathtt{in}$

$\ldots$

b23. $\nu \mathtt{r}_{d-1}.$

b24. $\mathtt{let}\ \ b^{\mathsf{p}(d-1)0} = \mathsf{penc}(\mathtt{a}_{(d-1)0}, pk_S, \mathsf{penc}(\mathtt{a}_{(d-1)0}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

$\ldots$

b25. $\mathtt{let}\ \ b^{\mathsf{p}(d-1)9} = \mathsf{penc}(\mathtt{a}_{(d-1)9}, pk_S, \mathsf{penc}(\mathtt{a}_{(d-1)9}, pk_A, \mathsf{B}(\mathtt{r}_{d-1}, pk_B, \mathtt{a}_{(d-1)9})))\ \ \mathtt{in}$

b26. $\mathtt{let}\ \ ve_{d-1} = (b^{\mathsf{p}(d-1)0}, \ldots, b^{\mathsf{p}(d-1)9})\ \ \mathtt{in}$

b27. $\mathtt{let}\ \ ve = (ve_0, \ldots, ve_{d-1})\ \ \mathtt{in}$

b28. $\mathsf{out}(\mathtt{ch}, \mathsf{denc}(\mathtt{r}, pk_{CRM}, ve)).\ P_{check}\big)$

Figure 23: The HRM14 bidder process.

the maximum price $9\ldots9$ (=10^k-1). After sending out his bids, the bidder waits for the bids to be sealed and signed, and then verifies whether his bids are correctly sealed and signed. The verification behaviour is modelled in the subsequent process P_{check} (Figure 27).

On receiving the bidding vectors from each bidder (**c1**), the CRM decrypts the deniable encryption and obtains the real bidding vectors (**c2**), and then sends the bidding vectors to the sealer via private channel `privch` (**c3**), see Figure 24. The permutation is modelled as sending the bidding vectors in parallel, which captures all possible permutations.

$P_{CRM} :=$
c1. $\mathsf{in}(\mathtt{ch}, xev_1).\ \mathsf{in}(\mathtt{ch}, xev_2).\ \cdots.\ \mathsf{in}(\mathtt{ch}, xev_m).$
c2. $\mathtt{let}\ \ xv_1 = \mathsf{pdec}(xev_1, \mathsf{sk}_{CRM})\ \ \mathtt{in}\ \ \cdots$
$\mathtt{let}\ \ xv_m = \mathsf{pdec}(xev_1, \mathsf{sk}_{CRM})\ \ \mathtt{in}$
c3. $(\mathsf{out}(\mathtt{privch}, xv_1) \mid \cdots \mid \mathsf{out}(\mathtt{privch}, xv_m))$

Figure 24: The CRM process.

When the sealer receives bidding vectors of all m bidders (**s1**), he first gets each bids in the bidding vectors (**s2-s3**). For each bid of a bidder, the sealer generates two nonces, and seals the bid with the nonces together with the sealer's private key which is used to nullify the sealer's public key in the bid (Figure 25). For instance, line **s4** generates the nonces for bidder B_1; lines **s5-s11** seal all bids for bidder B_1; then the sealed bidding vectors for B_1 is published (**s12**). Other bidder's bidding vectors are treated in the same way. Lines **s13-s21** show the sealing of bids for bidder B_m. In addition, for each bidding vector (containing 10 elements), the sealer publishes the response-vector. For instance, for the first vector of bidder B_1 (sv_1^0), the sealer generates the hash of all rs's used in the vector as its response-vector (**s22**). Other B_1's bidding vectors' response-vectors are calculated in the same way (**s23**). Finally, B_1's bidding vectors and their response-vectors are published to the Bulletin Board (**s34**). Other bidders' bids are sealed in the same way (**s25-s27**).

The auctioneer reads in each bidder's bidding vectors together with their corresponding response-vectors (**a1-a2**). The auctioneer signs each bidding vector of all bidders (Figure 26). For instance, **a3** models that the auctioneer signs the first vector of

$P_S :=$

s1. $\mathsf{in}(\mathtt{privch}, xvs_1).\ \mathsf{in}(\mathtt{privch}, xvs_2).\ \cdots.\ \mathsf{in}(\mathtt{privch}, xvs_m).$

s2. $\mathtt{let}\ ((xvs_1^{\mathsf{p}00}, \ldots, xvs_1^{\mathsf{p}09}), \ldots, (xvs_1^{\mathsf{p}(d-1)0}, \ldots, xvs_1^{\mathsf{p}(d-1)9})) = xvs_1\ \mathtt{in}$

$\ldots$

s3. $\mathtt{let}\ ((xvs_m^{\mathsf{p}00}, \ldots, xvs_m^{\mathsf{p}09}), \ldots, (xvs_m^{\mathsf{p}(d-1)0}, \ldots, xvs_m^{\mathsf{p}(d-1)9})) = xvs_m\ \mathtt{in}$

s4. $\nu\, \mathtt{r}_1^{\mathsf{p}00}.\ \cdots.\ \nu\, \mathtt{r}_1^{\mathsf{p}(d-1)9}.\ \cdots \nu\, \mathtt{rs}_1^{\mathsf{p}00}.\ \cdots.\ \nu\, \mathtt{rs}_1^{\mathsf{p}(d-1)9}.$

s5. $\mathtt{let}\ sv_1^{\mathsf{p}00} = \mathsf{seal}(xvs_1^{\mathsf{p}00}, \mathtt{r}_1^{\mathsf{p}00}, \mathtt{rs}_1^{\mathsf{p}00}, \mathtt{sk}_S)\ \mathtt{in}$

$\ldots$

s6. $\mathtt{let}\ sv_1^{\mathsf{p}09} = \mathsf{seal}(xvs_1^{\mathsf{p}09}, \mathtt{r}_1^{\mathsf{p}09}, \mathtt{rs}_1^{\mathsf{p}09}, \mathtt{sk}_S)\ \mathtt{in}$

s7. $\mathtt{let}\ sv_1^0 = (sv_1^{\mathsf{p}00}, \cdots, sv_1^{\mathsf{p}09})\ \mathtt{in}$

$\ldots$

s8. $\mathtt{let}\ sv_1^{\mathsf{p}(d-1)0} = \mathsf{seal}(xvs_1^{\mathsf{p}(d-1)0}, \mathtt{r}_1^{\mathsf{p}(d-1)0}, \mathtt{rs}_1^{\mathsf{p}(d-1)0}, \mathtt{sk}_S)\ \mathtt{in}$

$\ldots$

s9. $\mathtt{let}\ sv_1^{\mathsf{p}(d-1)9} = \mathsf{seal}(xvs_1^{\mathsf{p}(d-1)9}, \mathtt{r}_1^{\mathsf{p}(d-1)9}, \mathtt{rs}_1^{\mathsf{p}(d-1)9}, \mathtt{sk}_S)\ \mathtt{in}$

s10. $\mathtt{let}\ sv_1^{d-1} = (sv_1^{\mathsf{p}(d-1)0}, \cdots, sv_1^{\mathsf{p}(d-1)9})\ \mathtt{in}$

s11. $\mathtt{let}\ sv_1 = (sv_1^1, \cdots, sv_1^{d-1})\ \mathtt{in}$

s12. $\mathsf{out}(\mathtt{ch}, sv_1).$

$\ldots$

s13. $\nu\, \mathtt{r}_m^{\mathsf{p}00}.\ \cdots.\ \nu\, \mathtt{r}_m^{\mathsf{p}(d-1)9}.\ \cdots \nu\, \mathtt{rs}_m^{\mathsf{p}00}.\ \cdots.\ \nu\, \mathtt{rs}_m^{\mathsf{p}(d-1)9}.$

s14. $\mathtt{let}\ sv_m^{\mathsf{p}00} = \mathsf{seal}(xvs_m^{\mathsf{p}00}, \mathtt{r}_m^{\mathsf{p}00}, \mathtt{rs}_m^{\mathsf{p}00}, \mathtt{sk}_S)\ \mathtt{in}$

$\ldots$

s15. $\mathtt{let}\ sv_m^{\mathsf{p}09} = \mathsf{seal}(xvs_m^{\mathsf{p}09}, \mathtt{r}_m^{\mathsf{p}09}, \mathtt{rs}_m^{\mathsf{p}09}, \mathtt{sk}_S)\ \mathtt{in}$

s16. $\mathtt{let}\ sv_m^0 = (sv_m^{\mathsf{p}00}, \cdots, sv_m^{\mathsf{p}09})\ \mathtt{in}$

$\ldots$

s17. $\mathtt{let}\ sv_m^{\mathsf{p}(d-1)0} = \mathsf{seal}(xvs_m^{\mathsf{p}(d-1)0}, \mathtt{r}_m^{\mathsf{p}(d-1)0}, \mathtt{rs}_m^{\mathsf{p}(d-1)0}, \mathtt{sk}_S)\ \mathtt{in}$

$\ldots$

s18. $\mathtt{let}\ sv_m^{\mathsf{p}(d-1)9} = \mathsf{seal}(xvs_m^{\mathsf{p}(d-1)9}, \mathtt{r}_m^{\mathsf{p}(d-1)9}, \mathtt{rs}_m^{\mathsf{p}(d-1)9}, \mathtt{sk}_S)\ \mathtt{in}$

s19. $\mathtt{let}\ sv_m^{d-1} = (sv_m^{\mathsf{p}(d-1)0}, \cdots, sv_m^{\mathsf{p}(d-1)9})\ \mathtt{in}$

s20. $\mathtt{let}\ sv_m = (sv_m^0, \cdots, sv_m^{d-1})\ \mathtt{in}$

s21. $\mathsf{out}(\mathtt{ch}, sv_m).$

s22. $\mathtt{let}\ response_1^0 = (sv_1^0, \mathsf{hash}(\mathtt{r}_1^{\mathsf{p}00}, \ldots, \mathtt{r}_1^{\mathsf{p}09}))\ \mathtt{in}$

$\ldots$

s23. $\mathtt{let}\ response_1^{d-1} = (sv_1^{d-1}, \mathsf{hash}(\mathtt{r}_1^{\mathsf{p}(d-1)0}, \ldots, \mathtt{r}_1^{\mathsf{p}(d-1)9}))\ \mathtt{in}$

s24. $\mathsf{out}(\mathtt{ch}, (response_1^0, \ldots, response_1^{d-1})).$

$\ldots$

s25. $\mathtt{let}\ response_m^0 = (sv_m^0, \mathsf{hash}(\mathtt{r}_m^{\mathsf{p}00}, \ldots, \mathtt{r}_m^{\mathsf{p}09}))\ \mathtt{in}$

$\ldots$

s26. $\mathtt{let}\ response_m^{d-1} = (sv_m^{d-1}, \mathsf{hash}(\mathtt{r}_{k0_m}^{\mathsf{p}(d-1)0}, \ldots, \mathtt{r}_m^{\mathsf{p}(d-1)9}))\ \mathtt{in}$

s27. $\mathsf{out}(\mathtt{ch}, (response_m^0, \ldots, response_m^{d-1}))$

Figure 25: The sealer process.

bidder B_1; **a4** models that the auctioneer signs the last vector of B_1. The signed vectors are appended to their corresponding vectors (**a5**) and then are published to the Bulletin Board (**a6**). Other bidders' sealed vectors are signed in a similar way (**a7-a10**).

$P_A :=$
a1. $\mathsf{in}(\mathtt{ch}, ((xva_1^0, xra_1^0), \ldots, (xva_1^{d-1}, xra_1^{d-1})))$.
...
a2. $\mathsf{in}(\mathtt{ch}, ((xva_m^0, xra_m^0), \ldots, (xva_m^{d-1}, xra_m^{d-1})))$.
a3. $\mathtt{let}\ \ av_1^0 = \mathsf{sign}(xva_1^0, \mathtt{sk}_A)\ \ \mathtt{in}$
...
a4. $\mathtt{let}\ \ av_1^{d-1} = \mathsf{sign}(xva_1^{d-1}, \mathtt{sk}_A)\ \ \mathtt{in}$
a5. $\mathtt{let}\ \ av_1 = ((xva_1^0, xra_1^0, av_1^0), \ldots, (xva_1^{d-1}, xra_1^{d-1}, av_1^{d-1}))\ \ \mathtt{in}$
a6. $\mathsf{out}(\mathtt{ch}, av_1)$.
...
a7. $\mathtt{let}\ \ av_m^0 = \mathsf{sign}(xva_m^0, \mathtt{sk}_A)\ \ \mathtt{in}$
...
a8. $\mathtt{let}\ \ av_m^{d-1} = \mathsf{sign}(xva_m^{d-1}, \mathtt{sk}_A)\ \ \mathtt{in}$
a9. $\mathtt{let}\ \ av_m = ((xva_m^0, xra_m^0, av_m^0), \ldots, (xva_m^{d-1}, xra_m^{d-1}, av_m^{d-1}))\ \ \mathtt{in}$
a10. $\mathsf{out}(\mathtt{ch}, va_m)$

Figure 26: The auctioneer process.

$P_{check} :=$
k1. $\mathsf{in}(\mathtt{ch}, ((xkv_1^0, xkr_1^0, xks_1^0), \ldots, (xkv_1^{d-1}, xkr_1^{d-1}, xks_1^{d-1})))$.
k2. $\mathtt{if}\ \ \mathsf{verify}(xkv_1^0, xkr_1^0, xks_1^0, (\mathsf{a}_{00}, \ldots, \mathsf{a}_{09})) = \mathsf{true} \wedge \ldots \wedge$
k3. $\mathsf{verify}(xkv_1^{d-1}, xkr_1^{d-1}, xks_1^{d-1}, (\mathsf{a}_{(d-1)0}, \ldots, \mathsf{a}_{(d-1)9})) = \mathsf{true}\ \ \mathtt{then}\ \ 0$
k4. $\mathtt{else}\ \ \mathsf{in}(\mathtt{ch}, ((xkv_2^0, xkr_2^0, xks_2^0), \ldots, (xkv_2^{d-1}, xkr_2^{d-1}, xks_2^{d-1})))$.
k5. $\mathtt{if}\ \ \mathsf{verify}(xkv_2^0, xkr_2^0, xks_2^0, (\mathsf{a}_{00}, \ldots, \mathsf{a}_{09})) = \mathsf{true} \wedge \ldots \wedge$
k6. $\mathsf{verify}(xkv_2^{d-1}, xkr_2^{d-1}, xks_2^{d-1}, (\mathsf{a}_{(d-1)0}, \ldots, \mathsf{a}_{(d-1)9})) = \mathsf{true}\ \ \mathtt{then}\ \ 0$
k7. $\mathtt{else}\ \ \mathsf{in}(\mathtt{ch}, ((xkv_3^0, xkr_3^0, xks_3^0), \ldots, (xkv_3^{d-1}, xkr_3^{d-1}, xks_3^{d-1})))$.
...
k8. $\mathtt{else}\ \ \mathsf{in}(\mathtt{ch}, ((xkv_m^0, xkr_m^0, xks_m^0), \ldots, (xkv_m^{d-1}, xkr_m^{d-1}, xks_m^{d-1})))$.
k9. $\mathtt{if}\ \ \mathsf{verify}(xkv_m^0, xkr_m^0, xks_m^0, (\mathsf{a}_{00}, \ldots, \mathsf{a}_{09})) = \mathsf{true} \wedge \ldots \wedge$
k10. $\mathsf{verify}(xkv_m^{d-1}, xkr_m^{d-1}, xks_m^{d-1}, (\mathsf{a}_{(d-1)0}, \ldots, \mathsf{a}_{(d-1)9})) = \mathsf{true}\ \ \mathtt{then}\ \ 0$
k11. $\mathtt{else}\ \ \mathsf{out}(\mathtt{ch}, \mathsf{e})$

Figure 27: The bidder verifying process.

Once the singed bids are published, the bidders can verify whether his bids are counted correctly (Figure 27). Since the bidders' bids are sealed and permuted, a bidder does not know which signature corresponds to his bids. Hence, the bidder reads in

signatures of an arbitrary bidder, (**k1**), and verifies the signatures using function verify (**k2-k3**). If for all d bidding vectors, the verification of their signatures are true, the bidder knows that the sealed-bids corresponding to the read-in signatures are the correct calculation of his bids. Otherwise, the bidder reads in another set of signatures and performs the verification again (**k4**). The bidder keeps checking until find his bids (**k5-k10**). If none of the sealed-bids are his bids, the bidder reports an error message represented by a constant e (**k11**).

6.5. Analysis

Receipt-freeness. We found that the protocol may not satisfy receipt-freeness due to that how the fake bidding vectors are generated is not clear. For instance, when generating the fake bidding vectors, if a set of fresh nonces are used, there will be a receipt for the adversary to verify the bidding price of a coerced bidder.

In details, if a bidder B_c claims that he bid w in vector v ((v,w) is marked as 'yes'), the adversary can coerce for the bid of (v,w), i.e., the adversary asks for the bid $\mathsf{penc}(\mathsf{a}_{vw}, pk_S, \mathsf{penc}(\mathsf{a}_{vw}, pk_A, \mathsf{B}(r_v, pk_{B_c}, a_{vw})))$. In addition, the adversary can ask for the bidder's private key $\mathtt{sk}_{B_c}$, the g_y to calculate the public key pk_{B_c} and the two nonces that are used to form the 'yes' mark for the bid, i.e., r_v and a_{vw}. Using these information, the adversary can construct the 'yes' mark $\mathsf{B}(r_v, pk_{B_c}, a_{vw})$, i.e., $G_{c,(v,w)}$. Then, for each bidder's v-th signature, the adversary tests whether $\prod_{j=0}^{9} Y_{S_t i,(v,w)} = R_{S_t i,v} \cdot \mathbb{X}_{i,v} \cdot \prod_{j=0}^{9} G_{c,(v,w)}$ holds. If the bidder did not lie, there should exist exactly one signature satisfying the above equation. If the bidder lies to the adversary – the bidder bids for u, instead of w, in vector v, but claims that he bids for w. In order to cheat, according to the protocol, the bidder first calculates his real bid for (v,u), i.e., $b_c^{\mathsf{p}_{vu}} = \mathsf{penc}(\mathsf{a}_{vu}, pk_S, \mathsf{penc}(\mathsf{a}_{vu}, pk_A, \mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathsf{a}_{vu})))$, and his real bid for (v,w), i.e., $b_c^{\mathsf{p}_{vw}} = \mathsf{penc}(\mathsf{a}_{vw}, pk_S, \mathsf{penc}(\mathsf{a}_{vw}, pk_A, \mathsf{M}_{no}))$. Then he calculates the fake bid for (v,u), i.e., $b_f^{\mathsf{p}_{vu}} = \mathsf{penc}(\mathsf{a'}_{vu}, pk_S, \mathsf{penc}(\mathsf{a'}_{vu}, pk_A, \mathsf{M}_{no}))$, and the fake bid for (v,w), i.e., $b_f^{\mathsf{p}_{vw}} = \mathsf{penc}(\mathsf{a'}_{vw}, pk_S, \mathsf{penc}(\mathsf{a'}_{vw}, pk_A, \mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathsf{a'}_{vw})))$. The bids are encrypted with deniable encryption, so that the CRM reads the real bids ($b_c^{\mathsf{p}_{vu}}$ and $b_c^{\mathsf{p}_{vw}}$), whereas the adversary reads the fake bids ($b_f^{\mathsf{p}_{vu}}$ and $b_f^{\mathsf{p}_{vw}}$). Since the adversary can ask for the nonces

of each bid and verify whether the nonces are those used in the bids, the bidder cannot lie about the nonces, i.e., the bidder has to send $\mathtt{a'}_{vu}$, $\mathtt{a'}_{vw}$, $\mathtt{r}_v$ to the adversary, from which the adversary calculates the 'yes' mark as $\mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathtt{a'}_{vw})$. Then the adversary uses this 'yes' mark to test whether the v-th signature of each bidder satisfies the equation $\prod_{j=0}^{9} Y_{S_t i,(v,w)} = R_{S_t i,v} \cdot \mathbb{X}_{i,v} \cdot \prod_{j=0}^{9} G_{\square}$, where $G_{\square} = \mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathtt{a'}_{vw})$. There is no vector satisfying the equation, because in B_c's v-th vector, the $G_{\square}$ equals $\mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathtt{a}_{vu})$, which satisfies the equation, instead of $\mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathtt{a'}_{vw})$. Hence, the adversary can tell that the bidder lied.

Similarly, when generating the fake bids, if the nonce used in each fake bid is exactly the same old nonce in the corresponding real bid, there exists a receipt as well. In this case, we have $b_f^{\mathsf{p}_{vu}} = \mathsf{penc}(\mathtt{a}_{vu}, pk_S, \mathsf{penc}(\mathtt{a}_{vu}, pk_A, \mathsf{M}_{no}))$ and $b_f^{\mathsf{p}_{vw}} = \mathsf{penc}(\mathtt{a}_{vw}, pk_S, \mathsf{penc}(\mathtt{a}_{vw}, pk_A, \mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathtt{a}_{vw})))$. The adversary coerces for $\mathtt{a}_{vw}$, $\mathtt{r}_v$ and $\mathtt{sk}_{B_c}$. The bidder cannot lie about them because the adversary can use them to construct $b_f^{\mathsf{p}_{vw}}$ and verify whether they are the real ones used in $b_f^{\mathsf{p}_{vw}}$. Since the bidder claims that he bids for (v,w), the adversary constructs the 'yes' mark as $\mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathtt{a}_{vw})$, which differs from the real 'yes' mark ($\mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathtt{a}_{vu})$) in the v-th vector. Thus, there is no v-th signature satisfying $\prod_{j=0}^{9} Y_{S_t i,(v,w)} = R_{S_t i,v} \cdot \mathbb{X}_{i,v} \cdot \prod_{j=0}^{9} G_{\square}$, where $G_{\square} = \mathsf{B}(\mathtt{r}_v, pk_{B_c}, \mathtt{a}_{vw})$. Hence, the adversary knows that the bidder lied.

Therefore, only using deniable encryption is not a guarantee of receipt-freeness.

Our fix. To ensure receipt-freeness, we additionally require that when the bidder calculates the fake bid, the bidder should use the real 'yes' bid's nonces for the fake 'yes' bid. That is, the bidder uses a_{vu} to calculate the fake bid for price (v,w), i.e., the fake bid for the price (v,w) shall be $\mathsf{penc}(\mathtt{a}_{vu}, pk_S, \mathsf{penc}(\mathtt{a}_{vu}, pk_A, \mathsf{B}(r_v, pk_{B_c}, a_{vu})))$, as shown in Figure 29.

Assuming the sealers and auctioneer are honest on the opening phase, we prove that after fixing the flaw on how to calculating the fake bids, the protocol satisfies receipt-freeness up to the bidding phase. We manually proved it because, the equations for PDE cannot be handled by ProVerif – ProVerif would not terminate. Differing from the chameleon-bit-commitments equations in the AS02 protocol, where the message is either a constant M_{yes} or a constant M_{no}, in the PDE equations, the message is not a

$P_B^{\mathtt{chc}} :=$

c1. $\nu\ \mathtt{sk}_B.\ \mathsf{out}(\mathtt{chc}, \mathtt{sk}_B).\mathtt{let}\ \ pk_B = \mathsf{pk}(\mathtt{sk}_B)\ \ \mathtt{in}$

c2. $\nu\ \mathtt{r}.\ \mathsf{out}(\mathtt{chc}, \mathtt{r}).\nu\ \mathtt{a}_{00}.\ \mathsf{out}(\mathtt{chc}, \mathtt{a}_{00}).\cdots.\ \nu\ \mathtt{a}_{09}.\ \mathsf{out}(\mathtt{chc}, \mathtt{a}_{09}).\cdots.$
$\nu\ \mathtt{a}_{v0}.\ \mathsf{out}(\mathtt{chc}, \mathtt{a}_{v0}).\cdots.\ \nu\ \mathtt{a}_{vw}.\ \mathsf{out}(\mathtt{chc}, \mathtt{a}_{vw}).\cdots.\ \nu\ \mathtt{a}_{vu}.\ \mathsf{out}(\mathtt{chc}, \mathtt{a}_{vu}).$
$\cdots.\ \nu\ \mathtt{a}_{v9}.\ \mathsf{out}(\mathtt{chc}, \mathtt{a}_{v9}).\cdots.\ \nu\ \mathtt{a}_{(d-1)0}.\ \mathsf{out}(\mathtt{chc}, \mathtt{a}_{(d-1)0}).\cdots.$
$\nu\ \mathtt{a}_{(d-1)9}.\ \mathsf{out}(\mathtt{chc}, \mathtt{a}_{(d-1)9}).$

c3. $\big(\nu \mathtt{r}_0.\ \mathsf{out}(\mathtt{chc}, \mathtt{r}_0).\mathtt{let}\ \ b^{\mathsf{p}_{00}} = \mathsf{penc}(\mathtt{a}_{00}, pk_S, \mathsf{penc}(\mathtt{a}_{00}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$
$\ldots$

c4. $\mathtt{let}\ \ b^{\mathsf{p}_{0i}} = \mathsf{penc}(\mathtt{a}_{0i}, pk_S, \mathsf{penc}(\mathtt{a}_{0i}, pk_A, \mathsf{B}(\mathtt{r}_0, pk_B, \mathtt{a}_{0i})))\ \ \mathtt{in}$
$\ldots$

c5. $\mathtt{let}\ \ b^{\mathsf{p}_{09}} = \mathsf{penc}(\mathtt{a}_{09}, pk_S, \mathsf{penc}(\mathtt{a}_{09}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

c6. $\mathtt{let}\ \ ve_0 = (b^{\mathsf{p}_{00}}, \ldots, b^{\mathsf{p}_{0i}}, \ldots, b^{\mathsf{p}_{09}})\ \ \mathtt{in}$
$\ldots$

c7. $\nu \mathtt{r}_v.\ \mathsf{out}(\mathtt{chc}, \mathtt{r}_v).\mathtt{let}\ \ b^{\mathsf{p}_{v0}} = \mathsf{penc}(\mathtt{a}_{v0}, pk_S, \mathsf{penc}(\mathtt{a}_{v0}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$
$\ldots$

c8. $\mathtt{let}\ \ b^{\mathsf{p}_{vw}} = \mathsf{penc}(\mathtt{a}_{vw}, pk_S, \mathsf{penc}(\mathtt{a}_{vw}, pk_A, \mathsf{B}(\mathtt{r}_v, pk_B, \mathtt{a}_{vw})))\ \ \mathtt{in}$
$\ldots$

c9. $\mathtt{let}\ \ b^{\mathsf{p}_{vu}} = \mathsf{penc}(\mathtt{a}_{vu}, pk_S, \mathsf{penc}(\mathtt{a}_{vu}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$
$\ldots$

c10. $\mathtt{let}\ \ b^{\mathsf{p}_{v9}} = \mathsf{penc}(\mathtt{a}_{v9}, pk_S, \mathsf{penc}(\mathtt{a}_{v9}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

c11. $\mathtt{let}\ \ ve_v = (b^{\mathsf{p}_{v0}}, \ldots, b^{\mathsf{p}_{vw}}, \ldots, b^{\mathsf{p}_{v9}})\ \ \mathtt{in}$
$\ldots$

c12. $\nu \mathtt{r}_{d-1}.\ \mathsf{out}(\mathtt{chc}, \mathtt{r}_{d-1}).$
$\mathtt{let}\ \ b^{\mathsf{p}_{(d-1)0}} = \mathsf{penc}(\mathtt{a}_{(d-1)0}, pk_S, \mathsf{penc}(\mathtt{a}_{(d-1)0}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$
$\ldots$

c13. $\mathtt{let}\ \ b^{\mathsf{p}_{(d-1)j}} = \mathsf{penc}(\mathtt{a}_{(d-1)j}, pk_S, \mathsf{penc}(\mathtt{a}_{(d-1)j}, pk_A, \mathsf{B}(\mathtt{r}_k, pk_B, \mathtt{a}_{(d-1)j})))\ \ \mathtt{in}$
$\ldots$

c14. $\mathtt{let}\ \ b^{\mathsf{p}_{(d-1)9}} = \mathsf{penc}(\mathtt{a}_{(d-1)9}, pk_S, \mathsf{penc}(\mathtt{a}_{(d-1)9}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

c15. $\mathtt{let}\ \ ve_{d-1} = (b^{\mathsf{p}_{(d-1)0}}, \ldots, b^{\mathsf{p}_{(d-1)j}}, \ldots, b^{\mathsf{p}_{(d-1)9}})\ \ \mathtt{in}$

c16. $\mathtt{let}\ \ ve = (ve_1, \ldots, ve_v, \ldots, ve_{d-1})\ \ \mathtt{in}$

c17. $\mathsf{out}(\mathtt{ch}, \mathsf{denc}(\mathtt{r}, pk_{CRM}, ve)).\ P_{check}\big)$

Figure 28: The $P_B^{\mathtt{chc}}$ process.

constant, when the message is a 'yes' mark. Thus, although the equations are similar, the chameleon-bit-commitments equations can be handled, whereas the PDE equations cannot be handled. The main proof steps are shown as follows: Let $P = P_B^{\mathtt{chc}}$ and let $Q = P_B^{fake}$. First, $Q^{\setminus \mathsf{out}(\mathtt{chc},\cdot)}$ is exactly the same as the process where the bidder bids for (v,u). Second, we show that the adversary cannot distinguish P from Q. In both of the two processes P and Q, the bidder sends to the adversary his secret key and the

$P_B^{fake} :=$

c1. $\nu\ \mathtt{sk}_B.\ \mathtt{let}\ \ pk_B = \mathsf{pk}(\mathtt{sk}_B)\ \ \mathtt{in}$

c2. $\nu\ \mathtt{r}.\ \nu\ \mathsf{a}_{00}.\ \cdots.\ \nu\ \mathsf{a}_{09}.\ \cdots.$
$\nu\ \mathsf{a}_{v0}.\ \cdots.\ \nu\ \mathsf{a}_{vw}.\ \cdots.\ \nu\ \mathsf{a}_{vu}.\ \cdots.\ \nu$
$\mathsf{a}_{v9}.\ \cdots.\ \nu\ \mathsf{a}_{(d-1)0}.\ \cdots.\ \nu\ \mathsf{a}_{(d-1)9}.$

c3. $\big(\nu \mathtt{r}_0.\ \mathtt{let}\ \ b^{\mathsf{p}_{00}} = \mathsf{penc}(\mathsf{a}_{00}, pk_S, \mathsf{penc}(\mathsf{a}_{00}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$
$\ldots$

c4. $\mathtt{let}\ \ b^{\mathsf{p}_{0i}} = \mathsf{penc}(\mathsf{a}_{0i}, pk_S, \mathsf{penc}(\mathsf{a}_{0i}, pk_A, \mathsf{B}(\mathtt{r}_0, pk_B, \mathsf{a}_{0i})))\ \ \mathtt{in}$
$\ldots$

c5. $\mathtt{let}\ \ b^{\mathsf{p}_{09}} = \mathsf{penc}(\mathsf{a}_{09}, pk_S, \mathsf{penc}(\mathsf{a}_{09}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

c6. $\mathtt{let}\ \ ve_0 = (b^{\mathsf{p}_{00}}, \ldots, b^{\mathsf{p}_{0i}}, \ldots, b^{\mathsf{p}_{09}})\ \ \mathtt{in}$
$\ldots$

c7. $\nu \mathtt{r}_v.\ \mathtt{let}\ \ b^{\mathsf{p}_{v0}} = \mathsf{penc}(\mathsf{a}_{10}, pk_S, \mathsf{penc}(\mathsf{a}_{10}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$
$\ldots$

c8. $\mathtt{let}\ \ b^{\mathsf{p}_{vw}} = \mathsf{penc}(\mathsf{a}_{vw}, pk_S, \mathsf{penc}(\mathsf{a}_{vw}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

c8′. $\mathtt{let}\ \ b^{\mathsf{p}_{vw}}_{fake} = \mathsf{penc}(\mathsf{a}_{vu}, pk_S, \mathsf{penc}(\mathsf{a}_{vu}, pk_A, \mathsf{B}(\mathtt{r}_v, pk_B, \mathsf{a}_{vu})))\ \ \mathtt{in}$
$\ldots$

c9. $\mathtt{let}\ \ b^{\mathsf{p}_{vu}} = \mathsf{penc}(\mathsf{a}_{vu}, pk_S, \mathsf{penc}(\mathsf{a}_{vu}, pk_A, \mathsf{B}(\mathtt{r}_v, pk_B, \mathsf{a}_{vu})))\ \ \mathtt{in}$

c9′. $\nu\ \mathsf{a}'_{vu}.\ \mathtt{let}\ \ b^{\mathsf{p}_{vu}}_{fake} = \mathsf{penc}(\mathsf{a}'_{vu}, pk_S, \mathsf{penc}(\mathsf{a}'_{vu}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$
$\ldots$

c10. $\mathtt{let}\ \ b^{\mathsf{p}_{v9}} = \mathsf{penc}(\mathsf{a}_{v9}, pk_S, \mathsf{penc}(\mathsf{a}_{v9}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

c11. $\mathtt{let}\ \ ve'_v = (b^{\mathsf{p}_{v0}}, \ldots, b^{\mathsf{p}_{vw}}, \ldots, b^{\mathsf{p}_{v9}})\ \ \mathtt{in}$

c11′. $\mathtt{let}\ \ ve_{v_{fake}} = (b^{\mathsf{p}_{v0}}, \ldots, b^{\mathsf{p}_{vw}}_{fake}, \ldots, b^{\mathsf{p}_{vu}}_{fake}, \ldots, b^{\mathsf{p}_{v9}})\ \ \mathtt{in}$
$\ldots$

c12. $\nu \mathtt{r}_{d-1}.$
$\mathtt{let}\ \ b^{\mathsf{p}_{(d-1)0}} = \mathsf{penc}(\mathsf{a}_{(d-1)0}, pk_S, \mathsf{penc}(\mathsf{a}_{(d-1)0}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$
$\ldots$

c13. $\mathtt{let}\ \ b^{\mathsf{p}_{(d-1)j}} = \mathsf{penc}(\mathsf{a}_{(d-1)j}, pk_S, \mathsf{penc}(\mathsf{a}_{(d-1)j}, pk_A,$
$\mathsf{B}(\mathtt{r}_{d-1}, pk_B, \mathsf{a}_{(d-1)j})))\ \ \mathtt{in}$
$\ldots$

c14. $\mathtt{let}\ \ b^{\mathsf{p}_{(d-1)9}} = \mathsf{penc}(\mathsf{a}_{(d-1)9}, pk_S, \mathsf{penc}(\mathsf{a}_{(d-1)9}, pk_A, \mathsf{M}_{no}))\ \ \mathtt{in}$

c15. $\mathtt{let}\ \ ve_{d-1} = (b^{\mathsf{p}_{(d-1)0}}, \ldots, b^{\mathsf{p}_{(d-1)j}}, \ldots, b^{\mathsf{p}_{(d-1)9}})\ \ \mathtt{in}$

c16. $\mathtt{let}\ \ ve = (ve_1, \ldots, ve_v, \ldots, ve_{d-1})\ \ \mathtt{in}$

c16′. $\mathtt{let}\ \ ve_{fake} = (ve_1, \ldots, ve_{v_{fake}}, \ldots, ve_{d-1})\ \ \mathtt{in}$

chc1. $\mathsf{out}(\mathtt{chc}, \mathtt{sk}_B).\mathsf{out}(\mathtt{chc}, \mathsf{fake}(\mathtt{r}, pk_{CRM}, ve, ve_{fake})).\mathsf{out}(\mathtt{chc}, \mathsf{a}_{00}).\cdots.$
$\mathsf{out}(\mathtt{chc}, \mathsf{a}_{09}).\ \ \cdots.\mathsf{out}(\mathtt{chc}, \mathsf{a}_{v0}).\cdots.\mathsf{out}(\mathtt{chc}, \mathsf{a}'_{vw}).\cdots.\mathsf{out}(\mathtt{chc}, \mathsf{a}_{vw}).$
$\cdots.\ \ \mathsf{out}(\mathtt{chc}, \mathsf{a}_{v9})).\ \ \cdots.\mathsf{out}(\mathtt{chc}, \mathsf{a}_{(d-1)0}).\cdots.\mathsf{out}(\mathtt{chc}, \mathsf{a}_{(d-1)9}).$

chc2. $\mathsf{out}(\mathtt{chc}, \mathtt{r}_1).\cdots.\mathsf{out}(\mathtt{chc}, \mathtt{r}_v).\cdots.\mathsf{out}(\mathtt{chc}, \mathtt{r}_{d-1}).$

c17. $\mathsf{out}(\mathtt{ch}, \mathsf{denc}(\mathtt{r}, pk_{CRM}, ve)).\ P_{check}\big)$

Figure 29: The fake process.

nonces used in calculating the bids and the deniable encryptions. The transition steps are shown is Figure 30. If the bidder claims that he bids $\mathsf{p}_{1i}, \ldots, \mathsf{p}_{vw}, \ldots, \mathsf{p}_{kj}$ in the price

$$
\begin{aligned}
P \quad & \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{sk}_B)}, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{r})}, \\
& \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{00})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{09})}, \cdots, \\
& \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{v0})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{vw})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{vu})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{v9})}, \cdots, \\
& \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{(d-1)0})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{(d-1)9})} \\
& \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{r}_0)}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{r}_v)}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{r}_{d-1})} \\
& \xrightarrow{\mathtt{out}(\mathtt{ch},\mathtt{denc}(\mathtt{r},pk_{CRM},ve))} \\
& P_{check} \mid \{\mathtt{sk}_B/x_{sk}\} \mid \{\mathtt{r}/x_r\} \\
& \quad \mid \{\mathtt{a}_{00}/x_{00}\} \mid \cdots \mid \{\mathtt{a}_{09}/x_{09}\} \mid \cdots \\
& \quad \mid \{\mathtt{a}_{v0}/x_{v0}\} \mid \cdots \mid \{\mathtt{a}_{vw}/x_{vw}\} \mid \cdots \mid \{\mathtt{a}_{vu}/x_{vu}\} \mid \cdots \mid \{\mathtt{a}_{v9}/x_{v9}\} \mid \cdots \\
& \quad \mid \{\mathtt{a}_{(d-1)0}/x_{(d-1)0}\} \mid \cdots \mid \{\mathtt{a}_{(d-1)9}/x_{(d-1)9}\} \\
& \quad \mid \{\mathtt{r}_0/y_0\} \mid \cdots \mid \{\mathtt{r}_v/y_v\} \mid \cdots \mid \{\mathtt{r}_{d-1}/y_{d-1}\} \\
& \quad \mid \{\mathtt{denc}(\mathtt{r},pk_{CRM},ve)/y\}
\end{aligned}
$$

$$
\begin{aligned}
Q \quad & \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{sk}_B)}, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{fake}(\mathtt{r},pk_{CRM}ve,ve_{fake}))}, \\
& \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{00})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{09})}, \cdots, \\
& \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{v0})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}'_{vw})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{vw})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{v9})}, \cdots, \\
& \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{(d-1)0})}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{a}_{(d-1)9})} \\
& \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{r}_0)}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{r}_v)}, \cdots, \xrightarrow{\mathtt{out}(\mathtt{chc},\mathtt{r}_{d-1})} \\
& \xrightarrow{\mathtt{out}(\mathtt{ch},\mathtt{denc}(\mathtt{r},pk_{CRM},ve))} \\
& P_{check} \mid \{\mathtt{sk}_B/x_{sk}\} \mid \{\mathtt{fake}(\mathtt{r},pk_{CRM},ve,ve_{fake})/x_r\} \\
& \quad \mid \{\mathtt{a}_{00}/x_{00}\} \mid \cdots \mid \{\mathtt{a}_{09}/x_{09}\} \mid \cdots \\
& \quad \mid \{\mathtt{a}_{v0}/x_{v0}\} \mid \cdots \mid \{\mathtt{a}'_{vw}/x_{vw}\} \mid \cdots \mid \{\mathtt{a}_{vw}/x_{vu}\} \mid \cdots \mid \{\mathtt{a}_{v9}/x_{v9}\} \mid \cdots \\
& \quad \mid \{\mathtt{a}_{(d-1)0}/x_{(d-1)0}\} \mid \cdots \mid \{\mathtt{a}_{(d-1)9}/x_{(d-1)9}\} \mid \\
& \quad \mid \{\mathtt{r}_0/y_0\} \mid \cdots \mid \{\mathtt{r}_v/y_v\} \mid \cdots \mid \{\mathtt{r}_{d-1}/y_{d-1}\} \\
& \quad \mid \{\mathtt{denc}(\mathtt{r},pk_{CRM},ve)/y\}
\end{aligned}
$$

Figure 30: A brief proof of receipt-freeness in fixed HRM14.

list, with the coerced information, the adversary can calculate each bid and obtains the vector ve, and then verifies the equation $\mathsf{denc}(x_r, pk_{CRM}, ve) =_E y$. This equation is obviously satisfied in process P, since the bidder did not lie. In process Q, the adversary calculates the vector ve_{fake} instead of ve. The fake vector ve_{fake} only differs from the real one ve on bid (v,w) and bid (v,u), due to that nonces x_{vw} and x_{vu} differ in P and Q but other nonces remain the same. Furthermore, in process Q, the adversary receives a fake nonce for the deniable encryption $\{\mathsf{fake}(\mathtt{r}, pk_{CRM}, ve, ve_{fake})/x_r\}$. The fake bid-

ding vector together with the fake nonce also satisfy $\mathsf{denc}(x_r, pk_{CRM}, ve_{fake}) =_E y$, due to that $\mathsf{denc}(r,k,m) = \mathsf{denc}(\mathsf{fake}(r,k,m,mf),k,mf)$. Hence, the equations that are satisfied in the frame $\{\mathtt{sk}_B/x_{sk}\} \mid \{\mathtt{r}/x_r\}\cdots \mid \{\mathtt{a}_{vw}/x_{vw}\} \mid \cdots \mid \{\mathtt{a}_{vu}/x_{vu}\} \mid \cdots \mid \{\mathsf{denc}(\mathtt{r}, pk_{CRM}, ve)/y\}$ (originated from process P) are also satisfied in the frame $\{\mathtt{sk}_B/x_{sk}\} \mid \{\mathsf{fake}(\mathtt{r}, pk_{CRM}, ve, ve_{fake})/x_r\} \mid \cdots \mid \{\mathtt{a'}_{vw}/x_{vw}\} \mid \cdots \mid \{\mathtt{a}_{vw}/x_{vu}\} \mid \cdots \mid \{\mathsf{denc}(\mathtt{r}, pk_{CRM}, ve)/y\}$ (originated from process Q). Hence, the adversary cannot tell whether the bidder lied.

In the subsequent steps, in both cases (P and Q), the CRM reads exactly the same bidding vector ve. For each bid, the sealer blinds them with newly generated nonces and publishes the blinded result. Hence, the adversary would not be able tell any difference. In particular,

$$
\begin{aligned}
& \nu\, \mathtt{r'}_{vw}.\nu\, \mathtt{rs}_{vw}.\{\mathsf{penc}(\mathsf{blind}(\mathtt{r}_v, \mathtt{r'}_{vw}), pk_A, \mathsf{blind}(\mathtt{rs}_{vw}, \mathsf{B}(\mathtt{r}_v, pk_B, \mathtt{a}_{vw})))/z\} \\
\approx_s \quad & \nu\, \mathtt{r'}_{vw}.\nu\, \mathtt{rs}_{vw}.\{\mathsf{penc}(\mathsf{blind}(\mathtt{r}_v, \mathtt{r'}_{vw}), pk_A, \mathsf{blind}(\mathtt{rs}_{vw}, \mathsf{M}_{no}))/z\} \\
\approx_s \quad & \nu\, \mathtt{r'}_{vw}.\nu\, \mathtt{rs}_{vw}.\{\mathsf{penc}(\mathsf{blind}(\mathtt{r}_v, \mathtt{r'}_{vw}), pk_A, \mathsf{blind}(\mathtt{rs}_{vw}, \mathsf{B}(\mathtt{r}_v, pk_B, \mathtt{a'}_{vw})))/z\}
\end{aligned}
$$

$$
\begin{aligned}
& \nu\, \mathtt{r'}_{vu}.\nu\, \mathtt{rs}_{vu}.\{\mathsf{penc}(\mathsf{blind}(\mathtt{r}_v, \mathtt{r'}_{vu}), pk_A, \mathsf{blind}(\mathtt{rs}_{vu}, \mathsf{M}_{no}))/z\} \\
\approx_s \quad & \nu\, \mathtt{r'}_{vu}.\nu\, \mathtt{rs}_{vu}.\{\mathsf{penc}(\mathsf{blind}(\mathtt{r}_v, \mathtt{r'}_{vu}), pk_A, \mathsf{blind}(\mathtt{rs}_{vu}, \mathsf{B}(\mathtt{r}_v, pk_B, \mathtt{a}_{vw}))))/z\} \\
\approx_s \quad & \nu\, \mathtt{r'}_{vu}.\nu\, \mathtt{rs}_{vu}.\{\mathsf{penc}(\mathsf{blind}(\mathtt{r}_v, \mathtt{r'}_{vu}), pk_A, \mathsf{blind}(\mathtt{rs}_{vu}, \mathsf{M}_{no}))/z\}
\end{aligned}
$$

where the first process in the equations is the case where the bidder did not lie and z is the real sealed bid, the second process is the case where the bidder lied and z is the real sealed bid, and the third process is the case where the bidder lied and z is the sealed bid that the adversary thought would be.

In the subsequent bidding verification step, assuming the sealer and the auctioneer are honest, the bidder can verify his bids in both cases and thus no error message would be received. In addition, the published information by the sealer (response-vectors) and the auctioneer (signatures) can only be used to verify the equation (2). After fixed the flaw mentioned earlier, the adversary cannot tell the two processes, P and Q, apart by applying function verify. In particular, in the v-th vector, the sealer publishes $\mathsf{hash}(r'_{v0}, \ldots, r'_{vw}, \ldots, r'_{vu}, \ldots, r'_{v9})$, the auctioneer publishes $\mathsf{sign}(ve_v, \mathtt{sk}_A)$ when

the bidder did not lie, and publishes $\mathsf{sign}(ve'_v, \mathtt{sk}_A)$ when the bidder lied. After obtaining the published information, the only equations that can be additional applied by the adversary is

$$\begin{aligned}
&\mathsf{sign}((\mathsf{penc}(\mathsf{blind}(a_0, r_0), pk, \mathsf{blind}(rs_0, m_0)), \ldots,\\
&\qquad \mathsf{penc}(\mathsf{blind}(a_9, r_9), pk, \mathsf{blind}(rs_9, m_9))), sk)\\
&= \mathsf{combine}(\mathsf{blind}(a_0, r_0), \ldots, \mathsf{blind}(a_9, r_9), sk).\\
&\mathsf{verify}((\mathsf{penc}(\mathsf{blind}(a_0, r_0), pk, \mathsf{blind}(rs_0, m_0)), \ldots,\\
&\qquad \mathsf{penc}(\mathsf{blind}(a_9, r_9), pk, \mathsf{blind}(rs_9, m_9))),\\
&\qquad \mathsf{combine}(\mathsf{blind}(a_0, r_0), \ldots, \mathsf{blind}(a_9, r_9), sk),\\
&\qquad (a_0, \ldots, a_9), \mathsf{hash}(rs_0, \ldots, rs_9)) = \mathsf{true}.
\end{aligned}$$

However, none of them can be used to distinguish whether the bidder lied. In particular,

$$\begin{aligned}
& \nu\, \mathtt{rs}_{v0}.\cdots.\nu\, \mathtt{rs}_{v9}.\{\mathsf{combine}(\mathsf{blind}(\mathtt{rs}_{v0}, \mathsf{M}_{no}), \ldots,\\
& \mathsf{blind}(\mathtt{rs}_{\mathtt{vw}}, \mathsf{B}(\mathtt{r}_v, pk_B, \mathtt{a}_{vw}))), \ldots, \mathsf{blind}(\mathtt{rs}_{\mathtt{vu}}, \mathsf{M}_{no}), \ldots, \mathsf{blind}(\mathtt{rs}_{\mathtt{v9}}, \mathsf{M}_{no}), sk)/t\}\\
\approx_s \quad & \nu\, \mathtt{rs}_{v0}.\cdots.\nu\, \mathtt{rs}_{v9}.\{\mathsf{combine}(\mathsf{blind}(\mathtt{rs}_{v0}, \mathsf{M}_{no}), \ldots, \mathsf{blind}(\mathtt{rs}_{\mathtt{vw}}, \mathsf{M}_{no}), \ldots,\\
& \qquad\qquad \mathsf{blind}(\mathtt{rs}_{\mathtt{vu}}, \mathsf{B}(\mathtt{r}_v, pk_B, \mathtt{a}_{vw})), \ldots, \mathsf{blind}(\mathtt{rs}_{\mathtt{v9}}, \mathsf{M}_{no}), sk)/t\}
\end{aligned}$$

That is, the adversary cannot distinguish the two cases using the first equation. Since applying the verify function on both frames lead to true, the adversary cannot distinguish the two cases using the second equation. Hence, the adversary cannot tell whether the bidder lied. Similarly, the adversary cannot distinguish the cases when the bidder lied in multiple vectors. Therefore, the protocol satisfies receipt-freeness in the bidding phase. Similar to the AS02 protocol, the bids are opened from higher price to lower price, and the opening stops when the highest bid is found. Assuming the auctioneer and sealer are honest, the non-winning bids are not revealed in the opening phase. Hence, the fixed protocol satisfies receipt-freeness for non-winning bidders.

Strong-bidding-price-secrecy. This protocol also satisfies the strong-bidding-price-secrecy for non-winning bidders (Definition 3) (similar to the proof of the satisfaction of receipt-freeness). The intuition is as follows: two bidding vectors cannot be distinguished from their deniable encryptions without knowing the corresponding secret key,

due to the use of fresh nonces. In particular,

$$\nu \mathtt{r}.\{\mathtt{denc}(\mathtt{r},\mathtt{pk},\mathtt{ve})/t'\} \approx_s \nu \mathtt{r}.\{\mathtt{denc}(\mathtt{r},\mathtt{pk},\mathtt{ve}')/t'\}$$

In the subsequent steps, the adversary cannot distinguish two bidding vectors neither, following similar reasoning in the previous proof. Furthermore, in the opening phase, the non-winning bids are not revealed. Hence, the strong-bidding-price-secrecy is satisfied. More importantly, receipt-freeness is stronger than strong-bidding-price-secrecy, i.e., a protocol satisfying receipt-freeness also satisfies strong-bidding-price-secrecy.

7. Related work

In this section, we summarise works in the literature on formalising privacy properties, including anonymity. In order to verify a claimed privacy property of a protocol, precise definitions of the property are required. A privacy property can be defined in different manners. For instance, we can distinguish binary privacy from quantitative privacy.

- Binary privacy: A protocol either satisfies a privacy property or not.
- Quantitative privacy: It defines to which extent a protocol satisfies a claimed privacy property. For example, sender anonymity can be quantified by the number of participants from which the adversary cannot identify the sender [42].

Quantitative enforced privacy properties have been defined for e-voting in a formal framework proposed by Jonker, Pang and Mauw [25]. In this framework, the enforced privacy property, coercion-resistance, is quantified using the size of possible candidates such that no matter which candidate the coerced voter votes for, the adversary cannot distinguish it from others. Many other ways [42, 43, 44] to quantify privacy can be found in the literature as well.

Definitions of a privacy property also vary depending on the techniques used to prove the satisfaction of the definition. We distinguish directly proving a privacy property (e.g., using game-based provable security) by showing that the adversary cannot

solve the underlying hard problem (e.g., integer factoring, discrete logarithm, 3-SAT, etc.) in order to break the property, from proving a privacy property in a symbolic model.

- Game-based provable security: A privacy property is defined as a game of the adversary and a hypothetical challenger. The privacy property is satisfied if no polynomially bounded adversary has a non-negligible advantage against the challenger in the game. Enforced privacy properties in e-voting have been defined in this way: receipt-freeness for a specific voting protocol (Prêt à Voter) [45] and a generic coercion-resistance for the e-voting domain [46].

- Symbolic model: Typically, the Dolev-Yao assumption is adopted: Cryptographic primitives are assumed to be perfect, e.g., the adversary cannot undo an encryption; and messages are considered to be abstract, e.g., data are expressed as symbols instead of bit-strings.

In the second category, formalisations of privacy properties vary depending on the used formal models. For instance,

- using epistemic model [47, 48]: Protocols are modelled as knowledge of users and the adversary. Epistemic logic is used to reason about knowledge. Privacy properties are formalised as epistemic formulas. Enforced privacy properties in e-voting have been formalised based on epistemic logic in a framework proposed by Küsters and Truderung [49].

- using process algebra: The behaviour of a system can be intuitively modelled as a process. Privacy properties are typically modelled as relations of processes.

Compared to epistemic logic, process algebra is better at modelling the behaviour of protocols. In particular, process algebras are designed for concurrent systems, thus are very suitable to model e-services in which users are often highly distributed. In addition, process algebras are often equipped with proof techniques for process equivalences and some of them are supported by automatic verification tools. Many process algebras are used to model cryptographic protocols and formalise privacy properties,

for example, CSP (communicating sequential processes) [50, 51, 52], μCRL [53, 54], spi calculus [55] and the applied pi calculus [22, 24, 15]. Enforced privacy properties were first formalised using the applied pi calculus for a specific e-voting protocol [24]. Later, a framework for e-voting was proposed using the applied pi calculus – the DKR framework [15]. In addition, enforced privacy properties for weighted voting were proposed using the applied pi calculus as well – the DLL framework proposed by by Dreier, Lafourcade and Lakhnech [56]. The DKR framework has been extended and applied in many formal definitions of enforced privacy properties [24, 57, 15, 56, 36, 58].

In this work, we adopt the Dolev-Yao assumption as in the symblolic model. Particularly, we model the AS02 protocol and the HRM14 protocol using a process algebra, the applied pi calculus. The privacy properties are formalised in the binary manner, instead of quantitive. We are the first to lift the formalisation of enforced privacy from the voting domain to the e-auction domain, and are the first to propose formalisation of bidding-price-secrecy and receipt-freeness in e-auctions. In the same category, Dreier et al. have formalised other properties in e-auctions, such as fairness, verifiability, non-repudiation and coercion-resistance [59, 60].

8. Conclusion

The main contribution of this paper is that we have proposed a formalisation of two privacy-type properties in sealed-bid e-auctions: strong bidding-price-secrecy and receipt-freeness for non-winning bidders, following definitions of vote privacy and receipt-freeness in voting [15]. We have modelled the AS02 protocol and the HRM14 protocol in the applied pi calculus, verified strong bidding-price-secrecy of the protocols automatically using ProVerif and receipt-freeness of the protocols manually. For the HRM14 protocol, we have found a flaw with receipt-freeness and proposed a fix.

In [6], Chen et al. proposed another auction protocol which can ensure the winner's privacy. Micali and RabinIn [8] recently proposed a protocol for a different type of auctions – Virckery auctions, which ensures both privacy and receipt-freeness for all bidders. We are interested in formally verifying privacy properties of these protocols

in the future.

Acknowledgements. 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. Naipeng Dong was financially supported by the National Research Fund of Luxembourg (project PHD-09-027) when working in University of Luxembourg, where the work was conducted.

References

[1] M. Harkavy, J. D. Tygar, H. Kikuchi, Electronic auctions with private bids, in: Proc. 3rd USENIX Workshop on Electronic Commerce, 1998, pp. 61–74.

[2] C. Cachin, Efficient private bidding and auctions with an oblivious third party, in: Proc. 6th ACM Conference on Computer and Communications Security, ACM Press, 1999, pp. 120–127.

[3] M. Naor, B. Pinkas, R. Sumner, Privacy preserving auctions and mechanism design, in: Proc. 1st ACM Conference on Electronic Commerce, ACM Press, 1999, pp. 129–139.

[4] M. Abe, K. Suzuki, Receipt-free sealed-bid auction, in: Proc. 5th Conference on Information Security, Vol. 2433 of LNCS, Springer, 2002, pp. 191–199.

[5] H. Lipmaa, N. Asokan, V. Niemi, Secure vickrey auctions without threshold trust, in: Proc. 6th Conference on Financial Cryptography, Vol. 2357 of LNCS, Springer, 2003, pp. 87–101.

[6] X. Chen, B. Lee, K. Kim, Receipt-free electronic auction schemes using homomorphic encryption, in: Proc. 6th Conference on Information Security and Cryptology, Vol. 2971 of LNCS, Springer, 2003, pp. 259–273.

[7] B. Ksiezopolski, Z. Kotulski, Cryptographic protocol for electronic auctions with extended requirements, Annales UMCS, Informatica 2 (1) (2004) 391–400.

[8] S. Micali, M. O. Rabin, Cryptography miracles, secure auctions, matching problem verification, Communication ACM 57 (2) (2014) 85–93.

[9] J. Dreier, H. Jonker, P. Lafourcade, Secure auctions without cryptography, in: Proc. 7th International Conference on Fun with Algorithms, Vol. 8496 of LNCS, Springer, 2014, pp. 158–170.

[10] W. Abubaker, Z. Qin, H. Xiong, Z. Qin, , M. Ramadan, A taxonomy of secure electronic English auction protocols, International Journal of Computers and Applications 37 (1) (2015) 28–36.

[11] J. Trevathan, Privacy and security in online auctions, Ph.D. dissertation, James Cook University (2007).

[12] J. Trevathan, Security, anonymity and trust in electronic auctions, ACM Crossroads 11 (3) (2005) 2.

[13] J. Howlader, A. Ghosh, T. D. Pal, Secure receipt-free sealed-bid electronic auction, in: Proc. Contemporary Computing – IC3, Vol. 40 of Communications in Computer and Information Science, Springer, 2009, pp. 228–239.

[14] K. Sakurai, S. Miyazaki, An anonymous electronic bidding protocol based on a new convertible group signature scheme, in: Proc. 5th Australasian Conference on Information Security and Privacy, Vol. 1841 of LNCS, Springer, 2000, pp. 385–399.

[15] S. Delaune, S. Kremer, M. D. Ryan, Verifying privacy-type properties of electronic voting protocols, Journal of Computer Security 17 (4) (2009) 435–487.

[16] T. Okamoto, An electronic voting scheme, in: Proc. IFIP World Conference on IT Tools, 1996, pp. 21–30.

[17] T. Okamoto, Receipt-free electronic voting schemes for large scale elections, in: Security Protocols Workshop, 1997, pp. 25–35.

[18] J. Howlader, S. K. Roy, A. K. Mal, Practical receipt-free sealed-bid auction in the coercive environment, in: Proc. 17th Conference on Information Security and Cryptology - ICISC, Vol. 8565 of LNCS, Springer, 2014, pp. 418–434.

[19] G. Lowe, Breaking and fixing the Needham-Schroeder public-key protocol using FDR, in: Proc. 2nd Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Vol. 1055 of LNCS, Springer, 1996, pp. 147–166.

[20] R. Chadha, S. Kremer, A. Scedrov, Formal analysis of multi-party contract signing, in: Proc. 17th IEEE Computer Security Foundations Workshop, IEEE CS, 2004, pp. 266–279.

[21] J. Dreier, J. Dumas, P. Lafourcade, Brandt's fully private auction protocol revisited, Journal of Computer Security 23 (5) (2015) 587–610.

[22] M. Abadi, C. Fournet, Mobile values, new names, and secure communication, in: Proc. 28th Symposium on Principles of Programming Languages, ACM Press, 2001, pp. 104–115.

[23] B. Blanchet, An efficient cryptographic protocol verifier based on prolog rules, in: Proc. 14th IEEE Computer Security Foundations Workshop, IEEE CS, 2001, pp. 82–96.

[24] S. Kremer, M. D. Ryan, Analysis of an electronic voting protocol in the applied pi calculus, in: Proc. 14th European Symposium on Programming, Vol. 3444 of LNCS, Springer, 2005, pp. 186–200.

[25] H. L. Jonker, J. Pang, S. Mauw, A formal framework for quantifying voter-controlled privacy, Journal of Algorithms in Cognition, Informatics and Logic 64 (2-3) (2009) 89–105.

[26] N. Dong, H. L. Jonker, J. Pang, Analysis of a receipt-free auction protocol in the applied pi calculus, in: Proc. 7th Workshop on Formal Aspects in Security and Trust, Vol. 6561 of LNCS, Springer, 2011, pp. 223–238.

[27] D. Dolev, A. C.-C. Yao, On the security of public key protocols, IEEE Transactions on Information Theory 29 (2) (1983) 198–207.

[28] M. D. Ryan, B. Smyth, Applied pi calculus, in: Formal Models and Techniques for Analyzing Security Protocols, IOS Press, 2011.

[29] J. Liu, A proof of coincidence of labeled bisimilarity and observational equivalence in applied pi calculus, available at `http://lcs.ios.ac.cn/~jliu/papers/LiuJia0608.pdf` (2011).

[30] B. Blanchet, From secrecy to authenticity in security protocols, in: Proc. 9th International Symposium on Static Analysis, Vol. 2477 of LNCS, Springer, 2002, pp. 342–359.

[31] B. Blanchet, Automatic proof of strong secrecy for security protocols, in: Proc. 25th IEEE Symposium on Security and Privacy, IEEE CS, 2004, pp. 86–100.

[32] M. Abadi, B. Blanchet, Computer-assisted verification of a protocol for certified Email, Science of Computer Programming 58 (1-2) (2005) 3–27.

[33] M. Abadi, B. Blanchet, C. Fournet, Just fast keying in the pi calculus, ACM Transactions on Information and System Security 10 (3) (2007) 1–59.

[34] B. Blanchet, A. Chaudhuri, Automated formal analysis of a protocol for secure file sharing on untrusted storage, in: Proc. IEEE Symposium on Security and Privacy, IEEE CS, 2008, pp. 417–431.

[35] L. Luo, X. Cai, J. Pang, Y. Deng, Analyzing an electronic cash protocol using applied pi-calculus, in: Proc. 5th Conference on Applied Cryptography and Network Security, Vol. 4521 of LNCS, Springer, 2007, pp. 87–103.

[36] N. Dong, H. L. Jonker, J. Pang, Formal analysis of privacy in an eHealth protocol, in: Proc. 17th European Symposium on Research in Computer Security, Vol. 7459 of LNCS, Springer, 2012, pp. 325–342.

[37] J. Dreier, R. Giustolisi, A. Kassem, P. Lafourcade, G. Lenzini, A framework for analysing verifiability in traditional and electronic exams, in: Proc. 11th Information Security Practice and Experience, Vol. 9065 of LNCS, springer, 2015, pp. 514–529.

[38] A. Horn, On sentences which are true of direct unions of algebras, Journal of Symbolic Logic 16 (1) (1951) 14–21.

[39] B. Blanchet, M. Abadi, C. Fournet, Automated verification of selected equivalences for security protocols, Journal of Logic and Algebraic Programming 75 (1) (2008) 3–51.

[40] B. Blanchet, Proverif: automatic cryptographic protocol verifier user manual for untyped inputs, `http://prosecco.gforge.inria.fr/personal/bblanche/proverif` (October 2012).

[41] V. Cheval, B. Blanchet, Proving more observational equivalences with ProVerif, in: Proc. 2nd Conference on Principles of Security and Trust, Vol. 7796 of LNCS, Springer, 2013, pp. 226–246.

[42] D. Chaum, The dining cryptographers problem: Unconditional sender and recipient untraceability, J. Cryptology 1 (1) (1988) 65–75.

[43] M. K. Reiter, A. D. Rubin, Crowds: anonymity for web transactions, ACM Transactions on Information and System Security 1 (1) (1998) 66–92.

[44] O. Berthold, A. Pfitzmann, R. Standtke, The disadvantages of free mix routes and how to overcome them, in: Proc. Workshop on Design Issues in Anonymity and Unobservability, 2000, pp. 30–45.

[45] D. Khader, P. Y. A. Ryan, Receipt freeness of Prêt à voter provably secure, IACR Cryptology ePrint Archive 2011 (2011) 594.

[46] R. Küsters, T. Truderung, A. Vogt, A game-based definition of coercion-resistance and its applications, in: Proc. 23rd IEEE Computer Security Foundations Symposium, IEEE CS, 2010, pp. 122–136.

[47] P. F. Syverson, S. G. Stubblebine, Group principals and the formalization of anonymity, in: Proc. 5thWorld Congress on Formal Methods, Vol. 1708 of LNCS, Springer, 1999, pp. 814–833.

[48] J. Y. Halpern, K. R. O'Neill, Anonymity and information hiding in multiagent systems, Journal of Computer Security 13 (3) (2005) 483–512.

[49] R. Küsters, T. Truderung, An epistemic approach to coercion-resistance for electronic voting protocols, in: Proc. 30th IEEE Symposium on Security and Privacy, IEEE CS, 2009, pp. 251–266.

[50] S. Schneider, Security properties and CSP, in: Proc. 17th IEEE Symposium on Security and Privacy, IEEE CS, 1996, pp. 174–187.

[51] S. Schneider, A. Sidiropoulos, CSP and anonymity, in: Proc. 4th European Symposium on Research in Computer Security, Vol. 1146 of LNCS, Springer, 1996, pp. 198–218.

[52] S. Older, S. Chin, Formal methods for assuring security of protocols, Computer Journal 45 (1) (2002) 46–54.

[53] J. Pang, Analysis of a security protocol in μCRL, in: Proc. 4th Conference on Formal Engineering Methods, Vol. 2495 of LNCS, Springer, 2002, pp. 396–400.

[54] T. Chothia, S. Orzan, J. Pang, M. T. Dashti, A framework for automatically checking anonymity with *mu*crl, in: Proc. 2nd Symposium on Trustworthy Global Computing, – TGC'06, 2006, pp. 301–318.

[55] M. Abadi, A. D. Gordon, A calculus for cryptographic protocols: The spi calculus, in: Proc. 4th ACM Conference on Computer and Communications Security, 1997, pp. 36–47.

[56] J. Dreier, P. Lafourcade, Y. Lakhnech, Defining privacy for weighted votes, single and multi-voter coercion, in: Proc. 17th European Symposium on Research in Computer Security, Vol. 7459 of LNCS, Springer, 2012, pp. 451–468.

[57] M. Backes, C. Hriţcu, M. Maffei, Automated verification of remote electronic voting protocols in the applied pi-calculus, in: Proc. 21st IEEE Computer Security Foundations Symposium, IEEE CS, 2008, pp. 195–209.

[58] N. Dong, H. L. Jonker, J. Pang, Enforcing privacy in the presence of others: Notions, formalisations and relations, in: Proc. 18th European Symposium on Research in Computer Security, Vol. 8134 of LNCS, Springer, 2013, pp. 499–516.

[59] J. Dreier, H. Jonker, P. Lafourcade, Defining verifiability in e-auction protocols, in: Proc. 8th ACM Symposium on Information, Computer and Communications Security, ACM, 2013, pp. 547–552.

[60] J. Dreier, P. Lafourcade, Y.Lakhnech, Formal verification of e-auction protocols, in: Proc. 1st Conference on Principles of Security and Trust, Vol. 7796 of LNCS, Springer, 2013, pp. 247–266.

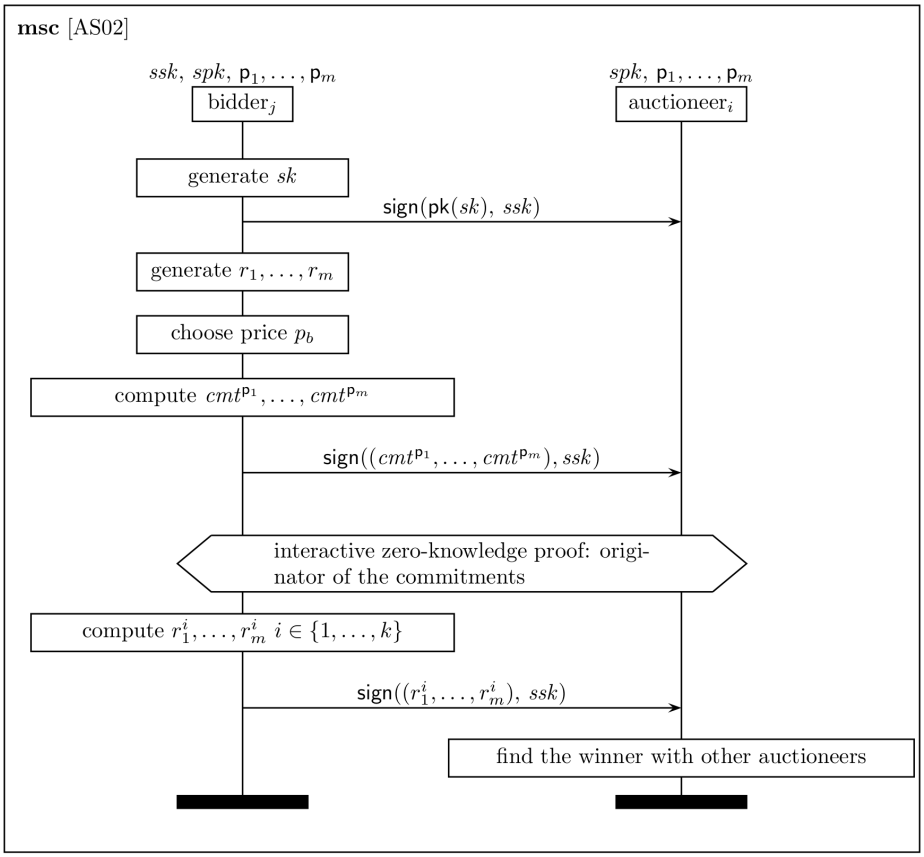

Figure 6: The AS02 protocol.

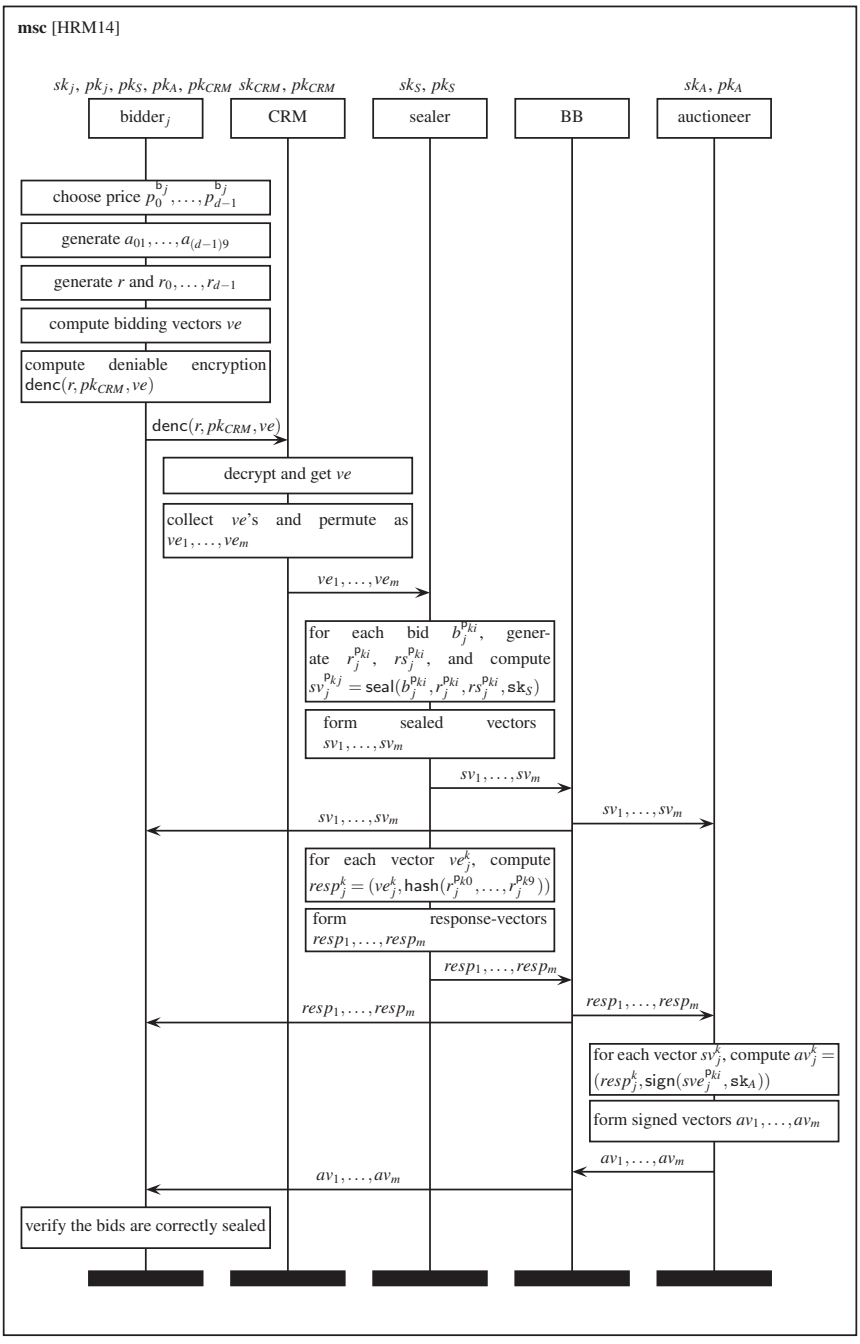

Figure 21: The HRM14 protocol.