Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 168 (2007) 5–28
www.elsevier.com/locate/entcs

Data Anonymity in the FOO Voting Scheme

S. Mauw J. Verschuren E.P. de Vink[1]

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O. Box 5600 MB Eindhoven
The Netherlands

Abstract

We study one of the many aspects of privacy, which is referred to as *data anonymity*, in a formal context. Data anonymity expresses whether some piece of observed data, such as a vote, can be attributed to a user, in this case a voter. We validate the formal treatment of data anonymity by analyzing a well-known electronic voting protocol.

Keywords: e-voting, anonymity and privacy, data anonymity, attribution set, formal methods for security protocols

1 Introduction

The privacy of users of electronic services is certainly not a matter of course. Electronic services like loyalty schemes and payment systems (e-auctions and electronic tolling for instance) may have severe consequences in the field of privacy. Currently foreseen developments like RFID [4] lead to a growing concern in this respect.

In the framework of "privacy" several aspects can be discerned. In the Common Criteria [31] the functional class FPR distinguishes between four aspects of privacy: anonymity, pseudonymity, unlinkability and unobservability. Anonymity, the topic of our research, ensures that a someone may use a resource or service without disclosing its identity. Pseudonymity ensures that someone may use a resource or service without disclosing its identity, but can still be hold accountable for that use. Unlinkability ensures that a someone may make multiple uses of resources or services without others being able to link these uses together. Unlinkability differs from pseudonymity in the sense that, although in pseudonymity the user is also not known, relations between different actions can be provided. Unobservability ensures that someone may use a resource or service without others, especially third

[1] Corresponding author. Email: evink@win.tue.nl

doi:10.1016/j.entcs.2006.11.001

parties, being able to observe that the resource or service is being used. Informal definitions, as the above, are essential for the understanding of the different notions of privacy, but will only allow to investigate a system informally. In addition to [31], informal definitions of anonymity are given in [34,16,42].

Concerning formal definitions of anonymity several ways of expressing anonymity have been proposed in the literature. Halpern and O'Neill [33] and also Syverson, Stubblebine and Gray [51,27] define information-hiding properties in terms of knowledge known to the agents of the system. They formalize information-hiding requirements and in particular anonymity. Reasoning about these properties is possible using epistemic logic. Hughes and Shmatikov [30] describe the behaviour of a system as a set of functions. Information-hiding properties like anonymity are formalized by means of the knowledge an attacker has about these functions. In this respect Hughes and Shmatikov talk about the function view of the attacker. The authors show that the function view framework is able to specify information hiding properties for any protocol formalism and arbitrary attacker models. Information theoretic approaches to characterize anonymity and unlinkability aspects are described, e.g., in articles written by Serjantov and Danezis [47], Diaz et al. [22] and by Steinbrecher and Köpsell [49].

Another way of expressing forms of anonymity is done by means of process algebras. Schneider and Sidiropoulos [46] use CSP when proposing a definition of anonymity. They define anonymity with respect to a set A of events. Anonymity with respect to A means that if any event in A occurs, it could equally well have been any other event in A. In [37], following Pfitzmann et al. [42], the present authors give a formal definition of anonymity by introducing the concept of an anonymity group AG. An anonymity group precisely indicates a group of users who are identical from the viewpoint of an intruder. An intruder cannot distinguish a certain user from other users of the anonymity group. Subsequently, in [37], it was analyzed to what extent and under what conditions the onion routing network [26,50,52] realized "anonymity" of users making use of the network. Probabilistic analysis of protocols for network anonymity, along the lines of work by Shmatikov [48] for Crowds [44], for the path set-up in Onion Routing [50] and Tarzan [24] has been reported in [3].

In this article it is shown that there exists another aspect of anonymity not formally addressed in articles [46] and [37]. There the focus is on the definition of what can be called *control anonymity*. This refers to hiding the originator of an event. This anonymity aspect plays an important role in anonymizing communication networks.

When considering other systems (e-voting systems for instance) one wants to prove that some piece of information (a specific vote) cannot be related to its originator (i.e. the voter). In other words, the originator of data must be kept secret. This property is called *data anonymity* here. In order to reason about data anonymity we need a model where it is not only possible to reason about the occurrence of events but where it is also possible to reason about the information within an event. In other words, a more sophisticated model is needed than the models introduced in [46] and [37]. In this paper we will concentrate on data anonymity

and define a formal framework to reason about this concept.

As already mentioned, data anonymity plays an essential role in voting systems. Electronic voting systems aim to provide a convenient, efficient and secure facility for recording and tallying votes. E-voting can be used in order to realize elections at a small local scale such as share holder meetings to full-scale national elections. Lipmaa [36] discerns two instances of e-voting: kiosk voting (voting in some fixed location using special hardware) or Internet voting using generally available devices, e.g. a PC or a mobile phone. In the literature several sets of requirements are found applicable to e-voting schemes [6,25,20]. These requirements are not identical. In [19] Cranor and Cytron made a survey of the literature and formulated four security related core properties.

accuracy (1) It should not be possible to alter votes. (2) It should be impossible to cause a situation where a valid vote is not taken into account. (3) On the other hand, an invalid vote should not be counted in the final tally.

invulnerability (1) Only eligible voters are allowed to vote. (2) These voters are only allowed to vote once.

verifiability Everyone should be able to verify independently that counting of the votes has been done correctly.

privacy (1) It is not possible for anyone to link a vote to a voter. (2) Moreover, a voter cannot prove that she voted in a specific manner.

In order to fulfill the above mentioned requirements the designers of e-voting systems have devised security protocols between different entities in the system. The goal of these protocols is to guarantee that the system fulfills the requirements even in the presence of an intruder. It is clear that our notion of data anonymity refers to the first part of the privacy requirement.

Broadly spoken, two groups of e-voting schemes can be discerned: (i) schemes that use homomorphic encryption (see, for example, [6,17,18,29]); (ii) schemes requiring an anonymous channel which is used to hide the identity of the voter when casting his vote (including [14,25,40,11]).

The voting scheme of Fujioka, Okamoto and Ohta [25] – which we concentrate on and which we refer to hereafter as the FOO voting scheme – makes use of blind signatures. In voting schemes using blind signatures, the voter obtains a token from the Administrator, which is a message blindly signed by the Administrator. After this registration phase, the voter can unblind the token and send his encrypted vote (signed by the administrator) to the Counter via an anonymous channel. Finally, the voter sends her decryption key – again via an anonymous channel – to the counter. After decryption, the counter adds the vote to the tally. The complete e-voting protocol will both informally and formally be described in Section 3 below.

As said before, the FOO scheme makes use of an anonymous channel. In their paper Fujioka, Okamoto and Ohta do not formulate explicit requirements concerning the anonymous channel however. Several anonymous channels have been proposed and analyzed. We mention [12,44,41,32,26,50,52] and [21,43].

Informal analysis of the FOO voting scheme [25,45] indicates that it fulfills the

data anonymity requirement. Herschberg, Cranor and Cytron perform informal analyses of the voting systems EVOX and Sensus [28,19], that are based on FOO. Herschberg [28] concludes that in the EVOX system the data anonymity aspect is only compromised if the anonymous channel is broken, that is if the channel reveals information about the origin of votes to the counter. Cranor and Cytron [19] argue that the system Sensus fulfills the first part of the privacy requirement and does not realize the second. In other words, Cranor argues that Sensus fulfills data anonymity whereas it does not prevent a voter to show that he voted in a certain way. In this paper we will *formally* analyze the FOO voting scheme. More specifically, we will verify to what extent FOO is in line with the given definition of data anonymity.

In their paper [35], Kremer and Ryan present a formal analysis of three security aspects of the FOO voting protocol. Fairness and eligibility are proven using the ProVerif tool [8,9], while for privacy they give a manual proof in the applied pi calculus [1,2]. Their privacy proof consists of verifying that two systems are observationally equivalent. In the first system, voter V_1 has vote v_1 and voter V_2 has vote v_2, while in the second system voter V_1 has vote v_2 and voter V_2 has vote v_2. If these systems cannot be distinguished by an observer, the voters and their votes are unlinkable. This is reminiscent to the control anonymity of the permutation based approach of [46].

In [39] Nielsen, Andersen and Nielson present an analysis of the FOO protocol using the LySA calculus. The LySA calculus is a dialect of the π-calculus centered around the concept of a global network [10]. In [39] the FOO protocol is formalized with LySA. It is argued that the protocol satisfies the requirements of accuracy and verifiability, as well as democracy and fairness. For anonymity, the unlinkability of voter and vote is considered, but not proven formally.

The main difference between the above approaches and ours is that we define the *attribution set* of a voter (complementary to the anonymity set), which contains all votes that can possibly be attributed to a voter. This allows us to *measure* the anonymity of a voter, whereas the approach of Kremer and Ryan provides a yes/no answer. In the case of the FOO protocol the attribution set is a useful notion, since Fujioka, Okamoto and Ohta are not completely clear about the status of the synchronization points they mention. Are these synchronizations required for the protocol to work? Are there any possible alternative synchronization points that will do? We will be able to analyze such questions after determining the attribution sets of the FOO protocol without synchronization points. Our approach also supports to design alternative approaches to increase the attribution set without providing full anonymity still. Such *practical anonymity* is attractive for larger scale application and may be achievable without the mentioned synchronization points.

Our formalization of the FOO voting scheme exploits an ACP style process algebra [5,23,7]. In the modeling of [35], the anonymous channel is not made explicit. In the applied pi calculus, every channel is anonymous unless input and output can be explicitly linked. By explicitly modeling the anonymous channel here, we take the opposite stance. Therefore, we can formally show that the level of data anonymity of the FOO voting scheme depends on two items: (i) the behaviour of

the anonymous channel; (ii) the way of synchronization after the registration phase. Our formal analysis reveals that from the viewpoint of data anonymity these items are interrelated. If no synchronization takes place after the registration phase, data anonymity can only be obtained if extra requirements are put on the anonymous channel. In the case voters may only start voting after everyone has registered, the anonymous channel may be less sophisticated. Finally, as explained in Section 5, our analysis suggests a weakness in case the publication medium is compromised.

This paper is organized as follows. In Section 2 we provide a formal definition of data anonymity. In Section 3 we describe and specify the FOO voting scheme. Section 4 provides a characterization of anonymity of voters in the FOO voting scheme in terms of attribution sets, whereas Section 5 presents the resulting vulnerability analysis. In Section 6 we discuss conclusions and future research.

2 Data Anonymity Framework

Let *Data*, *Keys*, *Nonces* and *Users*, ranged over by d, k, n and u, be the primitive classes of (sensitive) data, keys, nonces and users. The class *Terms* of terms, ranged over by φ, is given by the BNF

$$\varphi ::= d \mid k \mid n \mid u \mid (\varphi, \psi) \mid \{\varphi\}_k \mid [\varphi]_u$$

So, a term is either a primitive element, a pair of terms, the encryption of a term φ with the key k, or the term φ attributed to the user u. The attribution construction $[\varphi]_u$ is used to associate a term with a user. In particular, we will be interested in the sensitive data that can be linked to a specific user. We use $d \subseteq_u \varphi$ to denote that the datum d occurs in subterm $[\psi]_u$ of the term φ.

We will use the construct $[\varphi]_u$ in a process description to indicate with respect to which data we are interested in anonymity. Protocols are modeled such that none of the agents (including the intruder) can inspect or modify the attribution. It should be considered as a construct at the meta-level; its sole purpose is to facilitate verification.

The class *Event* of events, ranged over by e, consists of triples $\langle sender, receiver, \varphi \rangle$ representing the communication of the term φ from the user *sender* to the user *receiver*. Thus, $Event = Users \times Users \times Terms$. In case $e = \langle sender, receiver, \varphi \rangle$, we use $msg(e)$ to denote the term φ. In a given setting we fix a class $Obs \subseteq Event$ of observables.

The class *Traces* of traces, given by $Traces = Event^*$ and ranged over by t, consists of all finite traces of events. The semantics of a system S is given by the set of traces $traces(S) \subseteq Traces$. We refer to $traces(S)$ as the trace set of the system S. The function obs: $Traces \to Traces$ is defined by

$$\begin{aligned} obs(\varepsilon) &= \varepsilon \\ obs(e \cdot t) &= e \cdot obs(t) && \text{if } e \in Obs \\ obs(e \cdot t) &= obs(t) && \text{if } e \notin Obs. \end{aligned}$$

Clearly, for any $t \in Traces$, it holds that $obs(t) \in Obs^*$. We use the notation $\varphi \in t$

for a term φ and trace t if, for some sender *sender* and receiver *receiver* the event $\langle sender, receiver, \varphi\rangle$ is an event of t. We use the notation $d \in_u t$ for a datum d and trace t if $d \subseteq_u \varphi$ for some term $\varphi \in t$. We write, for a trace t, $\bot \in_u t$ if for no datum d it holds that $d \in_u t$. We use a as typical element of $Data_\bot = Data \cup \{\bot\}$.

The class *Know* of knowledge sets, ranged over by K, is the collection of subsets of terms, i.e. $Know = \mathcal{P}(Terms)$. For a particular situation we fix a knowledge set $I \in Know$ called the initial intruder knowledge. The auxiliary function *know*: $Know \times Traces \to Know$, to be given in a minute, returns for a knowledge set K and trace t the knowledge that is built up starting from the knowledge K along the trace t. First, we need a notion of closure for knowledge sets. We say that the term φ can be derived from the knowledge set K, notation $K \vdash \varphi$, if φ can be derived from K by repetitive use of the derivation rules in Table 1.

$\varphi,\ \psi \vdash_{pair} (\varphi, \psi)$	$\varphi,\ k \vdash_{enc} \{\varphi\}_k$
$(\varphi, \psi) \vdash_{left} \varphi$	$\{\varphi\}_k,\ k \vdash_{dec} \varphi$
$(\varphi, \psi) \vdash_{right} \psi$	$[\varphi]_u \vdash_{user} \varphi$

Table 1: Knowledge derivation rules

The closure $closure(K)$ of a knowledge set K is then given by

$$closure(K) = \{\ \varphi \mid K \vdash \varphi\ \}.$$

Based on this notion of closure, the function *know* can be given by

$$know(K, \varepsilon) = K$$
$$know(K, e \cdot t) = know(J, t)$$

where $J = closure(K \cup \{msg(e)\})$.

Note that the accumulation of knowledge in the definition above, ignores sender and receiver roles. So, all communication of a trace can be observed. To compensate for this, we use the function *obs* to prevent that particular traffic is collected.

In order to deal with encrypted terms we introduce the notion of tagging, an auxiliary technical mechanism to mark terms that cannot be decrypted by the observer or intruder. In essence, with keys k, ℓ, m not in the particular knowledge set, it helps to distinguish, for terms φ, ψ and ρ, the two traces $\{\varphi\}_k \cdot \{\psi\}_\ell$ from $\{\rho\}_m \cdot \{\rho\}_m$ while identifying the two traces $\{\varphi\}_k \cdot \{\psi\}_\ell$ and $\{\psi\}_\ell \cdot \{\varphi\}_k$. First we introduce the class *Tags* of tags with typical element τ. The class *TagTerms* is constructed similar to the class *Terms* but with *Tags* added as a new primitive ingredient. Thus, *TagTerms*, also ranged over by φ, is given by the BNF

$$\varphi ::= d \mid k \mid n \mid u \mid \tau \mid (\varphi, \psi) \mid \{\varphi\}_k \mid [\varphi]_u.$$

The class *TagTraces* is the collection of finite strings of tagged events, i.e. finite strings of triples $\langle sender, receiver, \varphi\rangle$ where $\varphi \in TagTerms$. A tagging is an injective mapping $\theta\colon Terms \to Tags$. Given a knowledge set K and a tagging θ, the interpretation function $int^{\theta}_{K}\colon Terms \to TagTerms$ is given as follows:

$$\begin{aligned}
int^{\theta}_{K}(p) &= p && \text{for } p = d, k, n, u\\
int^{\theta}_{K}((\varphi,\psi)) &= (int^{\theta}_{K}(\varphi), int^{\theta}_{K}(\psi))\\
int^{\theta}_{K}(\{\varphi\}_k) &= \{int^{\theta}_{K}(\varphi)\}_k && \text{if } k \in K\\
int^{\theta}_{K}(\{\varphi\}_k) &= \theta(\{\varphi\}_k) && \text{if } k \notin K\\
int^{\theta}_{K}([\varphi]_u) &= [int^{\theta}_{K}(\varphi)]_u.
\end{aligned}$$

So, if a term φ is encrypted with a key k that does not belong to the knowledge set under consideration, the composed term $\{\varphi\}_k$ is interpreted as a tag, viz. the tag $\theta(\{\varphi\}_k) \in Tags$ yielded by the tagging θ. Now, two traces $t, t' \in Traces$ are considered equivalent with respect to a knowledge set K, notation $t \sim_K t'$, if they yield the same knowledge and are equal upto renaming of tags, i.e. $know(K,t) = know(K,t')$ and there exists a bijection $\beta\colon Tags \to Tags$ such that $int^{\theta}_{L}(t) = int^{\beta\circ\theta}_{L}(t')$, for any tagging θ and $L = know(K,t)$.

For example, for traces $t_1 = \{d_1\}_k \cdot d_2 \cdot \{d_3\}_\ell$, $t_2 = \{d_3\}_\ell \cdot d_2 \cdot \{d_1\}_k$, a knowledge set K such that $k, \ell \notin K$, and an arbitrary tagging θ, we have $int^{\theta}_{K}(t_1) = \theta(\{d_1\}_k) \cdot d_2 \cdot \theta(\{d_3\}_\ell) = \tau_1 \cdot d_2 \cdot \tau_3$ where $\tau_1 = \theta(\{d_1\}_k)$ and $\tau_3 = \theta(\{d_3\}_\ell)$. If $\beta\colon Tags \to Tags$ is a bijection of tags that switches τ_1 and τ_3, we obtain $int^{\beta\circ\theta}_{K}(t_2) = \beta(t_3) \cdot d_2 \beta(t_1) = \tau_1 \cdot d_2 \cdot \tau_3$. Hence, $t_1 \sim_K t_2$. On the other hand, no bijection $\gamma\colon Tags \to Tags$ will, assuming $\tau_1 \neq \tau_3$, verify $\gamma(\tau_1) = \tau_1$ and $\gamma(\tau_3) = \tau_1$. So, for t_1 above and $t_3 = \{d_1\}_k \cdot d_2 \cdot \{d_1\}_k$, we have $t_1 \not\sim_K t_3$.

The idea behind tagging is that different bit strings in different runs of the system can represent the same encrypted information. The difference can be due to non-essential phenomena, in particular a different choice of nonces in the other run. The identification of bitstrings does not make sense within the same system run where different bit strings represent really different data.

Finally, we are in a position to give the definition of our notion of an attribution set.

Definition 2.1 Let S be a system with set of traces $traces(S)$, set of observables Obs and initial intruder knowledge I. The attribution set $AS_t(u)$ of a user u with respect to the trace $t \in traces(S)$ is given by

$$AS_t(u) = \{\, a \mid \exists t' \in traces(S)\colon obs(t) \sim_I obs(t') \wedge a \in_u t' \,\}.$$

Thus, given a system run t, a datum d can be attributed to user u if for some trace t', that is the same as the trace t from the intruder's perspective, i.e. $t \sim_I t'$, d is in fact associated with u, i.e. d occurs in a subterm that is associated with u in some message in t'.

For instance, if u is a voter and t is a run of a voting protocol, then u's vote is anonymous in this run if from the viewpoint of the intruder every collected vote could have been attributed to u. This is the case if $AS_t(u)$ contains all votes collected in t.

The special element $\perp$ can be assigned to a user u if in a trace t', observational equivalent to the trace t, no datum d is associated with the user u.

In case $traces(S) = \{ t_1, t_2, t_3 \}$ where $t_1 = [\{d_1\}_k]_u \cdot [d_2]_v \cdot \{[d_3]_w\}_\ell$, $t_2 = \{[d_3]_u\}_\ell \cdot [d_2]_v \cdot \{[d_1]_u\}_k$, $t_3 = \{[d_1]_u\}_k \cdot [d_2]_v \cdot \{[d_1]_w\}_k$ and $d_2 \neq k, \ell \notin I$, we have, for user u, $AS_{t_1}(u) = \{ d_1, d_3 \}$ as $d_1 \subseteq_u [d_1]_u \in t_1$, $_1 \sim_I t_2$ and $d_3 \subseteq_u \{[d_3]_u\}_\ell \in t_2$. For user v it holds that $AS_{t_1}(v) = \{d_2\}$ as only d_2 is associated with v. Also, for user w, we have $AS_{t_1}(w) = \{d_3\}$ is a singleton. In t_2 no data is associated with w, whereas the trace t_3 is not observably equivalent to the reference trace t_1.

3 The FOO Voting Scheme

In this section we will present the voting scheme proposed by Fujioka, Okamoto and Ohta in 1992. First, we give an informal explanation, that is used as a basis for an formal description that follows.

3.1 Informal Description

In this section we will explain the electronic voting protocol proposed by Fujioka, Okamoto and Ohta [25], also known as the FOO voting scheme. This scheme is claimed to satisfy a number of security requirements, one of which is privacy of the voter. We will use the informal representation of the protocol in the Message Sequence Chart in Figure 1 to explain the protocol. The symbols occurring in the explanation are summarized in Table 2.

The protocol describes the communication between an *administrator*, a number of *voters* and a *counter*. The role of the administrator is to check if the voter is eligible to vote and to sign the (blinded) ballot of the voter. The role of the counter is to collect all (anonymous) ballots and to publish them.

We focus on the protocol interactions of an individual voter. Voter v starts by selecting his random secret key $k(v)$ and a random nonce $n(v)$ and he fills in his ballot $b(v)$. By encrypting his ballot with his key, he constructs a committed ballot *cb*. At a later stage the voter will make his ballot public by (anonymously) providing *cb* and $k(v)$. Next, the voter blinds his ballot with his secret nonce, which yields *bcb*. For this purpose, he uses the blinding operation denoted by $*$. In order to ensure that this is his blinded ballot, he signs the result, which gives *sv*.

The voter sends his identity, the blinded committed ballot and the signed blinded committed ballot to the administrator in order to allow the administrator to check that the voter is (still) eligible to vote and to verify v's signature. The administrator acknowledges the received ballot by signing it and returning the signed blinded committed vote *sa* to the voter. The signing algorithm used by the administrator is a so-called *blinding signature technique* [13]. The purpose is that the voter can obtain a signed committed ballot from the signed blinded committed ballot by

v		voter identity
$b(v)$		voter v's ballot
$k(v)$		voter v's key
$n(v)$		voter v's nonce
cb	$\{b\}_{k(v)}$	committed ballot
bcb	$cb * n(v)$	blinded committed ballot
sv	$sig_v(bcb)$	blinded committed ballot signed by v
sa	$sig_a(bcb)$	blinded committed ballot signed by a
$check(v)$		can v (still) vote?
$verify_x(p, sp)$		is sp indeed p signed by x?
$scb = sa/n(v)$	$sig_a(cb)$	committed ballot signed by a
L_1		public list of encrypted votes
L_2		public list of opened votes

Table 2: Symbols used

applying the unblinding operator (denoted by /). Formally, this requires that the signing and blinding operations commute.

Therefore, after verifying the signature of the administrator, the voter can deduce *scb*, which is his committed ballot signed by the administrator. After this, the role of the administrator ends and the voter communicates to the counter. Messages sent from the voter to the counter go via an anonymous channel, so that the identity of the sender of the messages cannot be retrieved.

The voter sends his committed ballot *cb* and its signed version *scb* to the counter, who verifies that it is indeed signed by the administrator. He stores the received information from all voters in the list L_1 and after all voters have voted (or after some deadline has passed), he publishes this list. Every voter can now verify that his committed ballot is in the list and sends the key to open the committed ballot to the counter (again using the anonymous channel). The counter opens the ballots and finally publishes the second list L_2, containing all open ballots.

Note that we only explained the main line of the protocol. We did not specify exactly what happens if one of the checks fail due to an attempt to disrupt the voting by one of the participants. The interested reader is referred to [25] for the details.

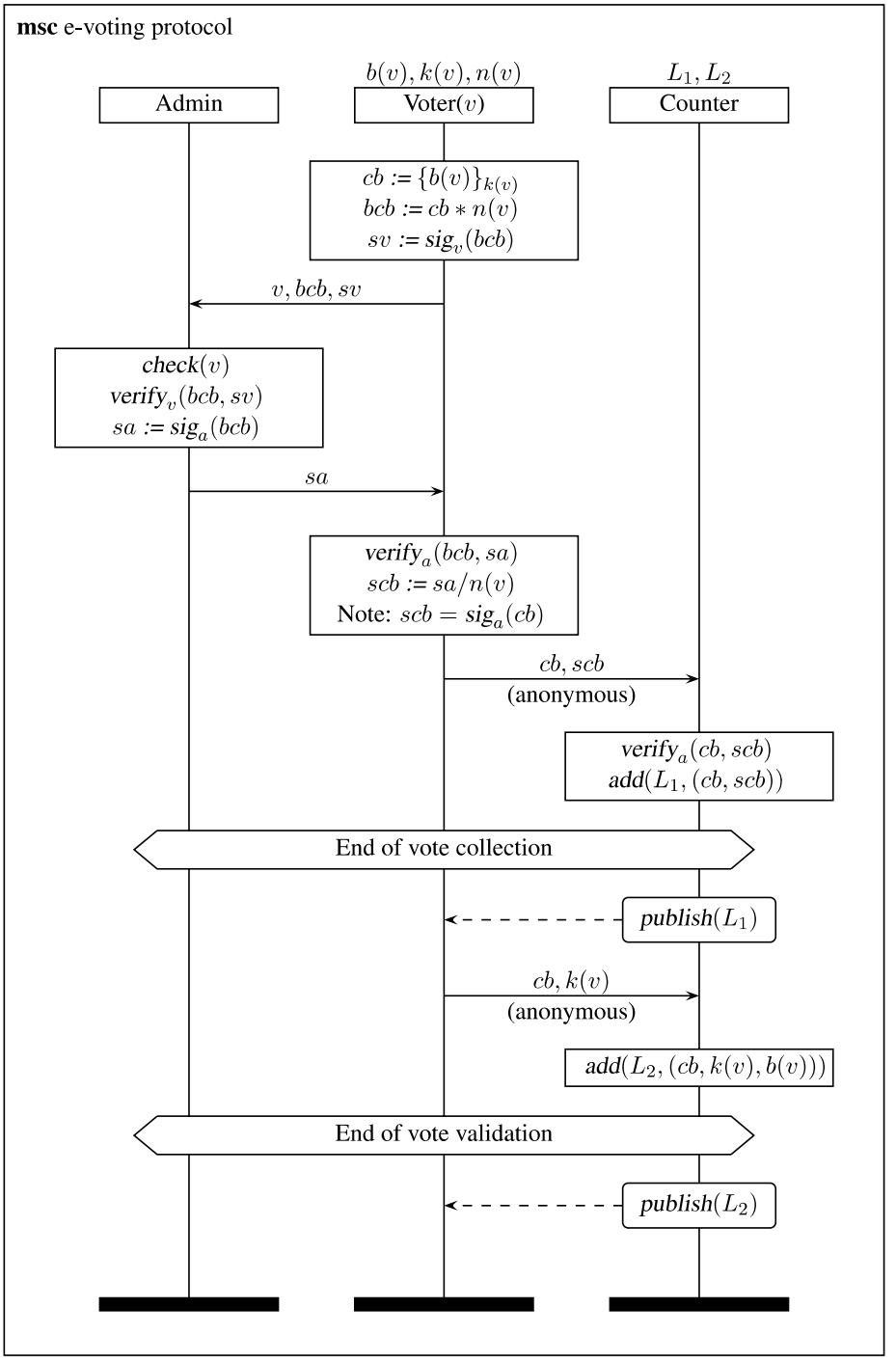

Figure 1: The FOO e-voting protocol.

3.2 Formal Specification

Next, we provide a formal specification of the FOO voting scheme in process algebra (see, e.g., [5,23,7]). This process algebra allows us to give a compact and formal specification of the set of all traces of the FOO voting scheme. However, it should be noted that, our treatment of anonymity is not tied to any particular specification formalism, as long as it supports reasoning about traces.

For the reader who is not familiar with the chosen specification language, we summarize the meaning of the constructs used. A process is specified by means of a (possibly recursive) equation. A process may be parametrized by a number of data values. Processes can be combined using operators. We use $\cdot$ to denote sequential composition, $+$ to denote non-deterministic choice, $\Sigma_{x\in X}$ to denote the generalization of $+$ over an index set X, $\|$ to denote (interleaved) parallel execution, and $\|_{x\in X}$ to denote the generalization of $\|$ over a index set X. The parallel composition operator also provides a means to synchronize two processes by synchronizing communicating events. The definition of the communication function takes the form $a \mid b = c$, which expresses that if events a and b occur in parallel, they will result in event c. In order to encapsulate partial communications (i.e. to force synchronization of events), the encapsulation operator ∂_H is used, where H is the set of communicating events that have to synchronize.

Apart from these operators at the process level, we will also need some operators and additional data types at the data level. Most of these have already been defined above. In addition we define types *List1*, *List2*, and *Buffer* to contain lists of pairs, lists of triples, and multisets of terms, respectively. We use the operator $\oplus$ to denote adding elements to a list as well as to a buffer. The deletion of an element is denoted by $\ominus$. We denote projection on the n-th element of a tuple by π_n, and we extend this notation to lists of tuples in the obvious way. By V and *Ballots* we denote the set of all (potential) voters, and all ballots, respectively. The voters play the role of the users as introduced in Section 2.

Since the informal description of the protocol does not specify the precise characteristics of the anonymous channel (surprisingly, no standard definition is available in the literature) and the publication medium, we will have to take some design decisions in this respect. Thus, besides the three processes mentioned in the informal specification, we will define two more processes, representing the anonymous channel and the publication medium.

Therefore, we consider processes

$$\mathit{Admin}(S),\ \mathit{Counter}(L_1, L_2),\ \mathit{Channel}(B),\ \mathit{Publisher}(L_1, L_2),$$

and a family of processes $\mathit{Voter}(v)$, for $L_1 \in \mathit{List1}$, $L_2 \in \mathit{List2}$, $S \subseteq V$, $B \in \mathit{Buffer}$, and $v \in V$. In addition, we use the subprocesses $\mathit{Counter}'$, $\mathit{Publisher}'$, and $\mathit{Publisher}''$. Using shorthand notation c, a, ch, p for these processes, we denote the set of sources and destinations for messages by $\mathit{Dest} = \{\, c, a, ch, p \,\} \cup V$.

Possible events are taken from the alphabet

$$\mathcal{E} = \{\, s_{x\to y}(\varphi), r_{x\to y}(\varphi) \mid x, y \in \mathit{Dest}, \varphi \in \mathit{Terms} \,\}.$$

Event $s_{v\to a}(\varphi)$, for instance, means that voter v sends a message φ, apparently to the administrator. Likewise, $r_{ch\to c}(\varphi)$ means that the counter receives a message φ, apparently coming from the channel.

In the formal description, we will use shorthand notation cb for $\{b\}_k$, bcb for $cb * n$, sv for $sig_v(bcb)$, sa for $sig_a(bcb)$, and scb for $sa/n(v)$.

We strive to obtain a minimal specification by leaving out all exceptional behaviour. Having understood the informal description of the protocol in the previous section, the formal algebraic specification is relatively easy to read. However, we will explain some of the intricacies of the design decisions that we made.

$$\begin{aligned}
&\mathit{Voter}(v) \;=\\
&\quad \textstyle\sum_{b\in \mathit{Ballots}, k\in \mathit{Keys}, n\in \mathit{Nonces}}\\
&\qquad s_{v\to a}(v, bcb, sv)\cdot r_{a\to v}(sa)\cdot s_{v\to ch}(cb, scb)\cdot\\
&\qquad\quad \textstyle\sum_{L_1\in \mathit{List1}, (cb,scb)\in L_1} r_{p\to v}(L_1)\cdot s_{v\to ch}(\{[b]_v\}_k, k)\cdot\\
&\qquad\qquad \textstyle\sum_{(L_1,L_2)\in \mathit{List1}\times \mathit{List2}} r_{p\to v}(L_1, L_2).
\end{aligned}$$

The voter starts by (non-deterministically) selecting his ballot, key, and nonce. Without making this explicit by the use of an internal event for this selection process, we assume that this choice is completely under control of the voter. Moreover, we require that different voters select different keys and different nonces. After having sent the first message, the voter receives his blinded committed ballot, signed by the administrator (sa). As an exception, this read event is not preceded by a summation. The reason is that, although the voter cannot construct this message himself, it is possible for him to verify whether the received datum satisfies the requirements. Given a deterministic signing algorithm, there will be only one such term. The ability to select only such validly signed term is expressed by the syntactic form of the expansion of sa, which is $sig_a(bcb)$, where bcb is known to the voter.

When the voter opens his ballot, by sending his key via the channel to the counter, we will have to model that the vote which is to be opened is attributed to this particular user. This attribution does not influence the behaviour of the system, nor is it observable by the intruder. The only reason to include this attribution is to enable the formal analysis of the system by determining the ballots that could be possibly attributed to this voter. Please notice that we did not require the voters to explicitly synchronize with the different phases of the election. The only event that can play this role is the reception of the lists from the publisher.

$$\mathit{Admin}(S) = \textstyle\sum_{v\in V, \varphi\in \mathit{Terms}} r_{v\to a}(v, \varphi, sig_v(\varphi))\cdot s_{a\to v}(sig_a(\varphi))\cdot \mathit{Admin}(S \setminus \{\, v \,\}).$$

The administration process is parameterized by the set of voters that still have to

vote. After having sent a signed ballot to a voter, the user is deleted from this set.

$$\begin{aligned}
&\mathit{Counter}(L_1, L_2) \;=\\
&\quad \textstyle\sum_{\varphi \in \mathit{Terms}} r_{ch\to c}(\varphi, \mathit{sig}_a(\varphi)) \cdot \mathit{Counter}(L_1 \oplus (\varphi, \mathit{sig}_a(\varphi)), L_2)\\
&\quad + s_{c\to p}(L_1) \cdot \mathit{Counter}'(L_1, L_2)\\
&\mathit{Counter}'(L_1, L_2) \;=\\
&\quad \textstyle\sum_{\varphi \in \pi_1(L_1), k \in \mathit{Keys}, \mathit{dec}_k(\varphi) \in \mathit{Ballots}} r_{ch\to c}(\varphi, k) \cdot \mathit{Counter}'(L_1, L_2 \oplus (\varphi, k, \mathit{dec}_k(\varphi)))\\
&\quad + s_{c\to p}(L_2).
\end{aligned}$$

The counter proceeds in two stages, which are modeled by two different process variables. In the first stage the counter receives a term signed by the administrator. The transition from the first to the second stage is modeled in a non-deterministic way. This allows us to validate the system in case the counter makes this transition "too early". In the second stage the counter accepts keys which allow him to unpack the committed ballots. We remark that the condition $\varphi \in \pi_1(L_1)$ should be interpreted as not to take the user attribution in φ into account. The end of the second stage is also modeled non-deterministically.

$$\begin{aligned}
\mathit{Channel}(B) \;=\; & \textstyle\sum_{v \in V, \varphi \in \mathit{Terms}} r_{v\to ch}(\varphi) \cdot \mathit{Channel}(B \oplus \{\varphi\})\\
& + \textstyle\sum_{\varphi \in B} s_{ch\to c}(\varphi) \cdot \mathit{Channel}(B \ominus \{\varphi\}).
\end{aligned}$$

The anonymous channel receives messages from a voter, which are added to the buffer. Alternatively, the channel can select any message from the buffer and pass it on to the counter.

$$\begin{aligned}
\mathit{Publisher}(L_1, L_2) &= \textstyle\sum_{L_1' \in \mathit{List}} r_{c\to p}(L_1') \cdot \mathit{Publisher}'(L_1', L_2)\\
\mathit{Publisher}'(L_1, L_2) &= \|_{v \in V}\, s_{p\to v}(L_1) \parallel r_{c\to p}(L_2') \cdot \mathit{Publisher}''(L_1, L_2')\\
\mathit{Publisher}''(L_1, L_2) &= \|_{v \in V}\, s_{p\to v}(L_1, L_2).
\end{aligned}$$

The publisher process accepts a list from the counter and sends it to all voters. In the course of publishing the first list, the publisher can also receive the second list, after which this list gets distributed too.

Finally, we specify the complete system *FOO* as the parallel composition of all its agents. Hence,

$$\mathit{FOO} = \partial_H(\|_{v \in V}\, \mathit{Voter}(v) \parallel \mathit{Admin}(V) \parallel \mathit{Counter}(\emptyset, \emptyset) \parallel \mathit{Channel}(\emptyset) \parallel \mathit{Publisher}(\emptyset)).$$

We initialize the administrator with the set of all voters, and the other processes with empty lists and buffers. The communication function matches read and sent events, i.e.

$$s_{x\to y}(\varphi) \mid r_{x\to y}(\varphi) = c_{x\to y}(\varphi)$$

for $x, y \in \mathit{Dest}$ and $\varphi \in \mathit{Terms}$. The events $c_{x\to y}(\varphi)$ are the concrete versions of the abstract events $\langle x, y, \varphi\rangle$ introduced in Section 2. The set H of encapsulated or forbidden events is given by

$$H = \{\, s_{x\to y}(\varphi), r_{x\to y}(\varphi) \mid x, y \in \mathit{Dest}, \varphi \in \mathit{Terms} \,\}.$$

The set of traces $\mathit{traces}(FOO)$ of the specified system now easily follows by applying the familiar operational semantics described in e.g. [5]. The definition of the anonymous channel requires that its input events cannot be observed by the intruder. Therefore, we define the set of observable events $\mathit{Obs} = \{\, c_{x\to y}(\varphi) \mid (x, y) \neq (v, ch), \varphi \in \mathit{Terms} \,\}$. We will not provide a precise definition of the initial intruder knowledge I. The reason is that we will consider the situation where all voters are trusted, as well as the situation in which some voters may be untrusted. Untrusted voters will be modeled by assuming that their chosen ballot, key and nonce are in the initial knowledge of the intruder.

4 The FOO Attribution Set

For the characterization of the attribution set of a voter we need an auxiliary definition. The predicate $\mathit{acmatch}(i, t)$ holds true iff in trace t upto position i the number of blinded votes signed by the administrator exactly matches the number of covered votes received by the counter. The specific property of an index i such that $\mathit{acmatch}(i, t)$, in a trace t of the FOO system, is that voters that have been registered no later than position i in t must have sent one of the covered votes collected by the counter upto this point.

Definition 4.1 The predicate *acmatch* is given by

$$\begin{aligned}\mathit{acmatch}(i,t) \iff\ & i \leq \mathit{len}(t) \wedge \\ & \#\{\, j \mid j < i \wedge \exists v, sa\colon t[j] = c_{a\to v}(sa) \,\} = \\ & \#\{\, j \mid j \leq i \wedge \exists cb, scb\colon t[j] = c_{ch\to c}(cb, scb) \,\}.\end{aligned}$$

The set of indices $\mathit{chunk}(i, t)$ for an index i and trace t such that $i \leq \mathit{len}(t)$ is given by

$$\begin{aligned}\{\, j \mid\ & (j < i \to \nexists h\colon j \leq h < i \wedge \mathit{acmatch}(h,t)) \wedge \\ & (i < j \to \nexists h\colon i \leq h < j \wedge \mathit{acmatch}(h,t)) \,\}.\end{aligned}$$

It holds that, if $j \in \mathit{chunk}(i, t)$ and $t[j]$ is a registration sent by the administrator or a covered vote received by the counter, than $t[j]$ is matched by a corresponding covered vote or registration $t[h]$, respectively, with $h \in \mathit{chunk}(i, t)$.

Given the above definition we are in a position to precisely describe the attribution set of a voter v with respect to a trace t of FOO. The characterization theorem 4.2 states that the attribution set of a voter v with respect to a trace t of FOO contains a ballot b if a covered vote, that apparently carried b, has been received by the counter in the same chunk as v's registration. Moreover, voter v could possibly not have voted at all (either because her covered vote or her opening

did not reach the counter in time) if a registration of a voter in the chunk of v remains unmatched, or all registrations of voters in the chunk of v are matched but not all are opened in the end.

Theorem 4.2 *For all $v \in V$ and $t \in traces(FOO)$ it holds that*

$$
\begin{aligned}
&AS_t(v) = \\
&\quad \{\, b \mid \exists i, j, \ell, sa, scb, k, v' : \\
&\qquad t[i] = c_{a\to v}(sa) \wedge \\
&\qquad i < j \wedge j \in chunk(i,t) \wedge \\
&\qquad t[j] = c_{ch\to c}(\{b\}_k, scb) \wedge \\
&\qquad t[\ell] = c_{ch\to c}(\{[b]_{v'}\}_k, k) \,\} \cup \\
&\quad \{\, \bot \mid \exists i, sa : \\
&\qquad t[i] = c_{a\to v}(sa) \wedge \\
&\qquad \nexists j : i < j \wedge acmatch(j,t) \,\} \cup \\
&\quad \{\, \bot \mid \exists i, j, sa, cb, scb, k : \\
&\qquad t[i] = c_{a\to v}(sa) \wedge \\
&\qquad i < j \wedge j \in chunk(i,t) \wedge \\
&\qquad t[j] = c_{ch\to c}(cb, scb) \wedge \\
&\qquad \nexists \ell : t[\ell] = c_{ch\to c}(cb, k)) \,\}
\end{aligned}
$$

Proof. *Directly from Lemma 4.5 and Lemma 4.8 below.* □

So, Theorem 4.2 states that the attribution set $AS_t(v)$ of a voter v with respect to a trace t of the FOO system, contains a vote b iff

- the registration of v of her blinded committed vote bcb is confirmed by the administrator, i.e. v receives $sa = sig_a(bcb)$, at some position i in t,
- at some position j in t, the counter received a covered vote $\{b\}_k$ together with the administrator's signature scb for $\{b\}_k$
- that could have matched the registration conformation at position i, i.e. the position j is later than i in t and is in the same chunk of t as i is,
- whereas the opening of the covered vote $\{b\}_k$, i.e. the receipt of the counter of the pair $(\{b\}_k, k)$, at some position ℓ in t, yields the vote b.

The special element $\bot$ is included in $AS_t(v)$ iff the registration of voter v was confirmed by the administrator at position i in t, but

- registrations and deposits of committed ballots where never in balance since then, i.e. for no position j later than i in t the predicate $acmatch(j,t)$ holds, so that voter v may not have sent her vote to the counter at all or that her vote was lost underway, or,
- there is a committed ballot that could match v's registration, i.e. for some position j later than i and in the same chunk as i in t, which is never opened, i.e. the corresponding key k never reached the counter.

Note that an opening $(\{b\}_k, k)$ as described above, may originate from a voter v'

different from v. Thus, with the meta-level annotation to attribute data to users in processes (not interpreted at the process level) the action $c_{ch\to c}(\{[b]_{v'}\}_k, k)$ occurs at position ℓ in the trace t, rather then $c_{ch\to c}(\{[b]_v\}_k, k)$ or $c_{ch\to c}(\{b\}_k, k)$.

The two inclusions underlying a proof of the theorem have been split over two lemmas. First we collect a few properties of the traces of *FOO* that follow straightforward from the definitions.

Lemma 4.3

(a) $b \in_v t$ *iff* $\exists i\colon t[i] = c_{ch\to c}(\{[b]_v\}_k, k)$.

(b) $t[i] = c_{ch\to c}(\{[b]_v\}_k, k)$ *implies* $\exists g, h, n\colon$
$g < h < i \wedge t[h] = c_{ch\to c}(\{b\}_k, sig_a(\{b\}_k)) \wedge t[g] = c_{a\to v}(sig_a(\{b\}_k * n))$. □

The next lemma states that *acmatch* and hence *chunk* respect the equivalence $\sim_I$. The lemma follows from the observation that the matching is based on the sender-receiver information and the form, not the content, of events.

Lemma 4.4 *Suppose* $t \sim_I t'$ *and* $k \in know(I, t)$. *If* $t[i] = c_{ch\to c}(\{b\}_k, sig_a(\{b\}_k)) \wedge acmatch(i, t)$ *then* $\exists i'\colon t'[i'] = c_{ch\to c}(\{b\}_k, sig_a(\{b\}_k)) \wedge acmatch(i', t')$. □

Next we prove half of the characterization theorem.

Lemma 4.5 *Let* $RHS_t(v)$ *be short for the concrete characterization of* $AS_t(v)$ *as given by Theorem 4.2. Then it holds that* $AS_t(v) \subseteq RHS_t(v)$, *for every voter* v *and trace* t *of FOO.*

Proof. *Let* $t \in FOO$ *and* $v \in V$. *Assume* $b \in AS_t(v)$. *Pick* $t' \in traces(FOO)$ *such that* $t' \sim_I t$ *and* $b \in_v t'$. *From Lemma 4.3 we obtain that* $t'[i'] = c_{a\to v}(sig_a(\{b\}_k * n))$, $t'[j'] = c_{ch\to c}(\{b\}_k, sig_a(\{b\}_k))$ *and* $t'[\ell'] = c_{ch\to c}(\{[b]_v\}_k, k)$ *for some indices* i', j' *and* ℓ' *and suitable key* k *and nonce* n. *Note* $k \in know(I, t')$, *so the term* $\{[b]_v\}_k$ *can be interpreted. Since* $t \sim_I t'$ *if follows that, for suitable indices* i, j *and* ℓ, $t[i] = c_{a\to v}(\varphi)$, $t[j] = c_{ch\to c}(\psi)$, $t[\ell] = c_{ch\to c}(\rho)$ *for terms* φ, ψ *and* ρ *that correspond to* $sig_a(\{b\}_k * n)$, $(\{b\}_k, sig_a(\{b\}_k))$ *and* $(\{[b]_v\}_k, k)$, *respectively. From the definition of the administrator and voter processes of FOO it follows that* $\varphi = sig_a(\{b'\}_{k'} * n')$ *for some ballot* b', *key* k' *and nonce* n', *as this is the only type of communication from the administrator to voter* v. *Since both* ψ *and* ρ *can be completely interpreted by the intruder modulo the user attribution, we get that* $\psi = (\{b\}_k, sig_a(\{b\}_k))$ *and* $\rho = (\{[b]_{v'}\}_k, k)$ *for some voter* v'. *It remains to show that* $j \in chunk(i, t)$. *As* i' *and* j' *match in* t', *this follows from Lemma 4.4 since* $t \sim_I t'$.

Assume $\perp \in AS_t(v)$. *Pick again* $t' \in FOO$ *such that* $t' \sim_I t$ *and* $\perp \in_v t'$. *We distinguish two cases: (i) voter* v *did only register; (ii) voter* v *sent in her covered vote, but did not open it. In the first case, we have, for some index* i *and suitable* b, k *and* n, $t'[i] = c_{a\to v}(sig_a(\{b\}_k * n))$ *but for no index* j, $j > i$, *it holds that* $t'[j] = c_{ch\to c}(\{b\}_k, sig_a(\{b\}_k))$. *So,* $t'[j]$ *remains unmatched. In the second case, we can pick indices* i *and* j *such that* $t'[i] = c_{a\to v}(sig_a(\{b\}_k * n))$, $t'[j] = c_{ch\to c}(\{b\}_k, sig_a(\{b\}_k))$. *As* $t'[i]$ *and* $t'[j]$ *match, it follows that* $j \in chunk(i, t')$. *By the assumption on unique keys and* $\perp \in_v t'$ *we derive that for no index* ℓ *it holds*

that $t'[\ell] = c_{ch\to c}(\{[b]_v\}_k, k)$. *Exploitation of the equivalence* $t \sim_I t'$, *similar to the reasoning above, yields corresponding indices of* t. *It follows that* $\bot \in RHS_t(v)$ *too.* □

For the proof of the other half of Theorem 4.2 we need a combinatorial result. Suppose a's and b's come in pairs with the a of a pair preceding the corresponding b (such as registrations and covered votes in traces of FOO). In general, one can not change the arrangements of pairs without disturbing the precedence. E.g., in the trace $a_1a_2b_1b_2$ we can swap b_1 and b_2 while maintaining $a_1 \prec b_1$ and $a_2 \prec b_2$, but in $a_1b_1a_2b_2$ we cannot. Roughly speaking, as long as some a is in the same chunk as any b with $a \prec b$, we can find a pairing containing a pair of the occurrences of the particular a and b that respects the precedence.

Lemma 4.6 *Let* w *be a string of length* $2s$ *in which the symbols* a_r, b_r, *for* $1 \le r \le s$, *all occur exactly once. If*

(i) a_r *occurs before* b_r, *for all* r, $1 \le r \le s$, *and*

(ii) w *has no proper prefix with an equal number of* a*'s and* b*'s*

then, for any two p, q, $1 \le p < q \le s$, *there exists a permutation* π *of* $\{1, \ldots, s\}$ *such that* $\pi(q) = p$ *for which the string* w' *obtained from* w *by replacing each* b_r *by* $b_{\pi(r)}$ *satisfies properties* (i) *and* (ii).

Proof. *Induction on* s. *Base case,* $s = 1$: *trivial. Induction step,* $s + 1$: *Pick indices* p *and* q, $1 \le p < q \le s + 1$. *If* a_q *occurs before* b_p *swapping of* b_p *and* b_q *clearly satisfies the claim. So, suppose* b_p *occurs before* a_q *in* w. *By condition (ii), for some* r *it holds that* b_p *occurs between* a_r *and* b_r. *The string* w' *of length* $2s$ *obtained from* w *by first applying the swapping of* b_p *and* b_r *and then removing the pair* a_r, b_r *satisfies the two conditions. By induction hypothesis, there exists a permutation* π *on* $\{1, \ldots, s + 1\} \backslash r$ *and a corresponding string* w'' *satisfying* (i) *and* (ii). *The permutation* π' *that extends* π *with* $\pi'(r) = r$ *and the string* w''' *obtained from* w'' *by inserting the pair* a_r, b_r *back again verifies the claim.* □

Example 4.7 Put $a_1 = w$, $a_2 = x$, $a_3 = y$, $a_4 = z$ and $b_1 = W$, $b_2 = X$, $b_3 = Y$, $b_4 = Z$. The string $wxXyzZYW$ satisfies the requirements (i) and (ii) of Lemma 4.6 above. For indices 1 and 2, we can simply swap W and X obtaining $wzWyzZYX$ for a permutation that maps 1 to 2. For the indices 2 and 4, we cannot, on $wxXyzZYW$, do a similar swapping of X and Z, as z would then no longer preceed Z. Therefore, we use W as auxilliary permuting element by mapping 2 to 4 (as intended) and 4 to 1, 1 to 2 yielding $wxWyzXYZ$.

We next prove the remaining part of the characterization of FOO's attribution set.

Lemma 4.8 *It holds that* $RHS_t(v) \subseteq AS_t(v)$, *for every voter* v *and trace* t *of FOO.*

Proof. *Suppose* $t \in FOO$ *and* $v \in V$. *Pick* $b \in RHS_t(v)$. *We consider the typical case that* v *timely submits its covered vote and opening, leaving the other two degenerate cases to the reader. Choose suitable indices, data elements and voter* v' *such that* $t[i'] = c_{a\to v}(sig_a(\{b'\}_{k'} * n))$, $t[j] = c_{ch\to c}(\{b\}_k, scb)$,

$t[\ell] = c_{ch\to c}(\{[b]_{v'}\}_k, k)$, $t[i] = c_{a\to v'}(sig_a(\{b\}_k * n))$, $t[j'] = c_{ch\to c}(\{b'\}_{k'}, scb')$ and $t[\ell'] = c_{ch\to c}(\{[b']_v\}_{k'}, k')$. Moreover, $i < j$ and $j \in chunk(i,t)$. By the matching lemma 4.6 we can arrange for a trace t' of FOO such that $t' \sim_I t$ and voter v submits a covered vote containing b. It follows that $b \in AS_t(v)$.

Suppose $\perp \in RHS_t(v)$. Suppose data that $t[i] = c_{a\to v}(sig_a(\{b\}_k * n))$ and for no j, $j > i$, it holds that $acmatch(j,t)$, for suitable indices. Then there exists $h \in chunk(i,t)$ such that $t[h] = c_{a\to v'}(sig_a(\{b'\}_{k'} * n'))$, but $t[j] = c_{ch\to c}(\{b'\}_{k'}, scb')$ for no j. Hence, for no ℓ, $t[\ell] = c_{ch\to c}(\{[b']_{v'}\}_{k'}, k')$. By rearranging the activity of v and v' we find a trace $t' \in FOO$ such that $t \sim_I t'$ and, for no ℓ, $t'[\ell] = c_{ch\to c}(\{[b']_v\}_{k'}, k')$. Thus $\perp \in_v t'$ and $\perp \in AS_t(v)$. The other case is similar and omitted. □

5 Vulnerability Analysis

In this section we interpret the results above and discuss (potential) vulnerabilities of the FOO voting scheme. Rather than providing a yes/no answer to the question whether the FOO voting scheme satisfies the privacy requirement claimed by Fujioka, Okamoto and Ohta, we have calculated the set of all ballots that could be attributed to a given user. There are several ways in which we can use this information to assess the vulnerability of the FOO voting scheme to privacy attacks.

Synchronization

The precise description provided by Theorem 4.2 is a starting point to look for ways to influence the privacy of the voter beneficially. In order to avoid the situation that votes are being submitted to the counter while voters are still able to register, Fujioka, Okamoto and Ohta suggest, as modeled in [35], to synchronize the registration, sending and opening of ballots of voters. However, the authors of [25] make not explicit which synchronizations are essential. The same applies to the privacy analysis that is part of [35].

One can distinguish three phases in the voting process: all registration takes place before any sending of ballots; openings are sent only after all votes have been sent in. This can be arranged by having explicit time lines. Looking at our specification, we find two places where synchronization may be implemented. The first way to synchronize affects the behaviour of the voters: they will have to wait until a certain deadline has passed before they submit their vote to the anonymous channel. After this deadline they will not try to register anymore. Disadvantage, when elections take place at a large scale, is that voter processes span a relatively large time frame. This may be undesirable from a usability point of view if such is at the responsibility of each individual voter. Therefore, as our formal specification facilitates, it is better to have the synchronization at another place, viz. to have the anonymous channel in charge for this. The anonymous channel specified above already starts producing output while still accepting input. One can adapt the

process such that it first collects all inputs and only then starts to transmit:

$$\begin{aligned} \mathit{Channel}(B) &= \textstyle\sum_{v\in V,\varphi\in \mathit{Terms}} r_{v\to ch}(\varphi)\cdot \mathit{Channel}(B\oplus\{\varphi\}) \;+\; \mathit{Channel}'(B)\\ \mathit{Channel}'(B) &= \textstyle\sum_{\varphi\in B} s_{ch\to c}(\varphi)\cdot \mathit{Channel}'(B\ominus\{\varphi\}). \end{aligned}$$

(Further details to deal with time or voter counting have been suppressed here.) This implements a synchronization point and solves the problem. Note that, the solution boils down to the plain old ballot box which may only be opened for counting after collection of the votes.

Active intruder

Taking the intruder's perspective the goal is to minimize the attribution set for a (or any) voter. In case of a malicious administrator in a non-synchronized system, he can distinguish the ballot of the first voter by delaying all other voters until the first voter's ballot is transmitted by the anonymous channel. If time allows to repeat this process, the intruder can jeopardize the privacy of all voters.

In a synchronized system, though, at the publisher side a similar attack is possible for an active intruder, i.e. a malicious party that also adapts the information sent out to voters: Instead of presenting the voters with the correct list L_1 of committed ballots, the intruder blocks this communication (or takes over control over the publisher process) and sends a specific voter v_1 with a list with only one entry from the original list and all other entries garbled. All voters different from v_1 receive a completely garbled list. Now two cases should be distinguished: the remaining entry indeed was the covered vote of v_1 or it was not. In the former case, v_1 will be the only voter sending in his opening, since the other voters will miss their covered votes from the list. In the latter case, all voter processes block as their covered votes are missing. Thus, for a population of n voters, chances are 1 out of n that the privacy of a single voter, v_1 in our case, is compromised. In the possibilistic stance this amounts to stating that an active intruder has for any voter a trace that violates the voter's anonymity.

Although havoc may result from complaints of voters missing their covered votes from the published list, one may even try to stretch the approach a bit further. Instead of only presenting voter v_1 with a list with only one valid entry, now voter v_1 gets a list that is scrambled except for entry e_1, voter v_2 gets a list that is scrambled except for entry e_2, and so on. The qualitative interpretation then yields that the intruder has a trace in which all votes are revealed. However, as a variation of the well-known drunker sailor problem [38], it follows, by linearity of expectation, that the expectation of the number of voters of which the privacy is breached is 1 only, thus showing no improvement (from the intruder's perspective) on the earlier attack. Due to their modeling of the privacy requirement, the analysis of the FOO protocol by Kremer and Ryan [35] did not reveal this weakness. In future work we aim at providing a precise analysis in which a Dolev-Yao intruder is modeled at the same level as the present representation of the processes involved.

Dummy votes

An alternative manner to enlarge the voter's attribution set is by exploiting the anonymous channel. Before any activity of voters, the anonymous channel accumulates a sufficient amount of signed dummy ballots from the administrator in the same way an eligible voter would do. Note that, the administrator is aware of the number of dummy votes that is generated this way. Next, when a regular voter presents a blinded committed vote to the anonymous channel, the channel not only outputs any of the regular votes it chooses, but also arbitrary many of its own registered dummy votes. The team of administrator and counter will not be able to tell the dummy votes and regular votes apart. As soon as the counter publishes the blinded committed vote on the first list of received votes, the voter can submit its opening without endangering its privacy.

The reason of the increased privacy lies in the size of the chunks, or rather, chunk. The predicate *acmatch* will only hold after the last bit of dummy votes have come out of the channel. if plenty of these are at disposal of the channel, this can be prolongated well after the last voter has sent in both its blinded vote and the opening thereof. Therefore the attribution set of all voters will be the set of all opened votes, together with $\perp$, if applicable.

As is to be expected, the increased anonymity without a two or three phase voting regime comes at a price. The anonymous channel needs to be trusted, not only for prudence when dealing with private information as before, but also regarding the very outcome of the election. Obviously, the channel can cover the number of votes for any option on its part and open this after the votes have been collected. The net outcome will then be the grand total per option minus the number of dummy votes for this option. However, there is a priori no guarantee that the channel will deliver the votes it commits itself to, as long as there is no control mechanism in place for this. It is conjectured that zero-knowledge techniques can help here, a topic for further investigation. It is noted though, that the formal description of the attribution set as given by Theorem 4.2 has catalyzed the above line of reasoning, that is, to our best knowledge, not conceived before.

Unlinkability

One of the drawbacks of a formal verification is the fact that it considers a formal object, rather than an actual implementation. While implementing the protocol many decisions have to be made, e.g. with respect to the actual cryptographic primitives. A concrete cryptographic algorithm may have certain properties, which in a particular setting could be used by an intruder to his advantage. In the FOO voting scheme anonymity of the voter essentially depends on the unlinkability between two events: the sending of a blinded committed ballot to the administrator and the sending of the committed ballot (and its corresponding key) to the counter. Now, if the implementation allows an observer to link these events, anonymity is breached. Such vulnerabilities could e.g. be introduced by naive use of a probabilistic signing algorithm or by including network information of the voter in the payload of messages and thus to travel unmodified through the supposedly anonymous channel.

In case a Public Key Infrastructure is adopted when implementing the FOO voting scheme, then also possibilities might emerge for linking the blinded committed ballot *bcb* to the committed ballot *cb*. More specifically, one could think of an implementation of the FOO voting scheme where the administrator sends the signed blinded committed vote to the voter together with its public key and corresponding certificate. The certificate was created by the Certification Authority (CA). The voter can check the validity and integrity of the received public key by checking the corresponding certificate.

If the CA is not strictly separated from the administrator, then an attack is possible in the following way: The administrator generates a public/secret key pair for each voter and asks the CA to generate a certificate for each public key. The administrator signs the blinded committed ballot of each voter with a different secret key. He sends the signed blinded committed vote together with the corresponding public key and its certificate to the voter.

The voter checks the certificate of the received public key and verifies the signature of the blinded committed vote. Subsequently, he unblinds the signature and sends the unblinded signature to the counter. The counter (together with the administrator) can determine the identity of the voter by finding out what public key matches the received signature.

Another problem might arise if the administrator adopts a group signature scheme. Group signatures, see e.g. [15], denote a signature scheme where each member of a group can generate a signature. The receiver (i.e. the voter in our case) cannot find out what group member generated it. When the administrator uses a group signature scheme he can act as a specific group member. Thus he has got the possibility to generate a signature specific for a certain voter. This leads to a link between the blinded committed ballot and the committed ballot.

Conspiring voters

In principle, a flooding attack might be possible in the presence of malicious and conspiring voters, affecting the availability of the system. However, if the system is synchronized in any of the two ways indicated above, conspiring agents can hardly influence the privacy of other agents. In order to model conspiring voters, we extend the initial intruder knowledge with the ballots, keys and nonces of the malicious voters. Repeating the calculation of the attribution set with this extended intruder knowledge will give a reduction of the attribution set as the ballots from the conspiring voters can be identified and deleted. We will not provide these calculations, since they are a straightforward extension of the case without conspiring voters. Care has to be taken to define the *acmatch* predicate in order to skip communications of the conspiring voters.

6 Conclusions

The main contribution of this paper is the definition of an attribution set. This set consists of all objects that can possibly be attributed to some user, given the

observations of an intruder. It can be considered a measure for the data anonymity of the user. If, for some system, this attribution set can be calculated in an explicit form, it will give insight in the vulnerabilities of the system and it can be used to strengthen the protocol.

The analysis of the FOO voting scheme clearly supports our view. Not only did it show how an intruder could try to break anonymity, it also showed that a synchronization point would solve the problem. We indicated two possibilities for such a synchronization, one of which was not considered by the designers of the FOO voting scheme. Our analysis also revealed some other possible weakness. If the communication from the publisher process to the voters is compromised, the intruder can manipulate this in such a way that at least one vote can be attributed to its voter. Furthermore, a weakness occurs if the administrator can manipulate the distribution of his public signing key.

Common experience with the use of formal methods shows that the act of formalizing clarifies the assumptions underlying the correctness of a system. The original description by Fujioka, Okamoto and Ohta suffers from underspecification (the properties required from the anonymous channel are not made clear) as well as from overspecification (cryptographic algorithms are chosen, rather than their relevant properties). Our specification and analysis clarified some of these issues.

Our work can be extended in several directions. First of all, our specification only considered the main operation of the protocol: whenever an unexpected situation occurs, the involved agent will simply deadlock. In the original description such exceptions are treated in a more meaningful way. However, we think that such exception handling is not essential for reaching privacy, but to satisfy other properties. A second extension, as mentioned above, is to consider a Dolev-Yao intruder modeled as a first-class citizen, instead of an eavesdropping intruder as in the present setting. Our current analysis hints at system flaws in this setting, but further work on modeling synchronization is needed to make this precise. A last promising point for follow-up research is to look for zero-knowledge mechanisms for a secure injection of dummy votes, so that the attribution set of all voters can be maximized without putting unnecessary trust in the anonymous channel.

References

[1] Abadi, M. and C. Fournet, *Mobile values, new names, and secure communication*, in: *Proc. POPL '01* (2001), pp. 104–115.

[2] Abadi, M. and C. Fournet, *Private authentication.*, Theoretical Computer Science **322** (2004), pp. 427–476.

[3] Adithia, M., "Probabilistic Analysis of Network Anonymity using PRISM," Master's thesis, Eindhoven University of Technology (2006).

[4] *Association for Automatic Identification and Mobility, RFID.org*, http://www.aimglobal.org/technologies/rfid/ (2004).

[5] Baeten, J. and W. Weijland, "Process Algebra," Cambridge Tracts in Theoretical Computer Science **18**, Cambridge University Press, 1990.

[6] Benaloh, J. and D. Tuinstra, *Receipt-free secret-ballot elections (extended abstract)*, in: *Proc. STOC '94, Montreal* (1994), pp. 544–553.

[7] Bergstra, J., A. Ponse and S. A. Smolka, "Handbook of Process Algebra," Elsevier, 2001.

[8] Blanchet, B., *An efficient cryptographic protocol verifier based on Prolog rules.*, in: *14th Computer Security Foundations Workshop, Cape Breton, Nova Scotia* (2001), pp. 82–96.

[9] Blanchet, B., M. Abadi and C. Fournet, *Automated verification of selected equivalences for security protocols.*, in: *20th IEEE Symposium on Logic in Computer Science, Chicago* (2005), pp. 331–340.

[10] Bodei, C., M. Buchholtz, P. Degano, F. Nielson and H. Nielson, *Automatic validation of protocol narration*, in: *Proc. CSFW 2003, Pacific Grove* (2003), pp. 126–140.

[11] Boyd, C., *A new multiple key cipher and an improved voting scheme*, in: *Proc. Eurocrypt '89* (1989), pp. 617–625.

[12] Chaum, D., *Untraceable electronic mail, return addresses, and digital pseudonyms*, Communications of the ACM **24** (1981), pp. 84–88.

[13] Chaum, D., *Security without identification: transaction systems to make big brother obsolete*, Communications of the ACM **28** (1985), pp. 1030–1044.

[14] Chaum, D., *Elections with unconditionally secret ballots and disruption equivalent to breaking rsa*, in: *Proc. Eurocrypt '88* (1988), pp. 177–182.

[15] Chaum, D. and E. van Heyst, *Group signatures*, in: D. Davies, editor, *Proc. EUROCRYPT '91* (1991), pp. 257–265.

[16] Clarke, R., *Introduction to dataveillance and information privacy, and definitions of terms*, http://www.anu.edu.au/people/Roger.Clarke/DV/Intro.html (1999).

[17] Cramer, R., M. Franklin, B. Schoenmakers and M. Yung, *Multi-autority secret-ballot elections with linear work.*, in: U. Maurer, editor, *Proc. Eurocrypt '96* (1996), pp. 72–83.

[18] Cramer, R., R. Gennaro and B. Schoenmakers, *A secure and optimally efficient multi-authority election scheme.*, in: W. Fumy, editor, *Proc. Eurocrypt '97* (1997), pp. 103–118.

[19] Cranor, L. and R. Cytron, *Sensus: A security-conscious electronic polling system for the internet*, http://lorrie.cranor.org/pubs/hicss/hicss.html.

[20] Cranor, L. and R. Cytron., *Design and implementation of a practical security-conscious electronic polling system*, Technical Report WUCS–96–02, Dept. of Computer Science, Washington University (1996).

[21] Desmedt, Y. and K. Kurosawa, *How to break a practical mix and design a new one*, in: B. Preneel, editor, *Proc. Eurocrypt 2000* (2000), pp. 557–572.

[22] Díaz, C., S. Seys, J. Claessens and B. Preneel, *Towards measuring anonymity*, in: R. Dingledine and P. Syverson, editors, *Proc. PET 2002* (2003), pp. 54–68.

[23] Fokkink, W., "Introduction to Process Algebra," Texts in Theoretical Computer Science, an EATCS Series, Springer, 2000.

[24] Freedman, M. and R. Morris, *Tarzan: a peer-to-peer anonymizing network layer.*, in: V. Atluri, editor, *ACM Conference on Computer and Communications Security*, 2002, pp. 193–206.

[25] Fujioka, A., T. Okamoto and K. Ohta, *A practical secret voting scheme for large scale elections*, in: J. Seberry and Y. Zheng, editors, *Advances in Cryptology – AUSCRYPT'92* (1992), pp. 244–251.

[26] Goldschlag, D., M. Reed and P. Syverson, *Hiding routing information*, in: R. Anderson, editor, *Proc. 1st International Workshop on Information Hiding* (1996), pp. 137–150.

[27] Gray, J. and P. Syverson, *A logical approach to multilevel security of probabilistic systems*, Distributed Computing **11** (1998), pp. 73–90.

[28] Herschberg, M. A., *Secure electronic voting over the world wide web*, http://theory.lcs.mit.edu/~cis/theses/herschberg/ (1997).

[29] Hirt, M. and K. Sako, *Efficient receipt-free voting based on homomorphic encryption.*, in: B. Preneel, editor, *Proc. Eurocrypt 2000* (2000), pp. 539–556.

[30] Hughes, D. and V. Shmatikov, *Information hiding, anonymity and privacy: A modular approach*, Journal of Computer Security (2002), pp. 3 – 36.

[31] ISO, *Common Criteria–ISO/IEC/15408*, http://csrc.nist.gov/cc/ (1999).

[32] Jakobsson, M., *A practical mix*, in: K. Nyberg, editor, *Proc. Eurocrypt '98* (1998), pp. 448–461.

[33] J.Y. Halpern, K. O., *Anonymity and information hiding in multiagent systems*, in: *Proceedings of the 16th IEEE Computer Security Foundations Workshop*, 2003, pp. 75–88.

[34] Korkea-Aho, M., *Anonymity and privacy in the electronic world* (1999), seminar on Network Security, Helsinki University of Technology.

[35] Kremer, S. and M. Ryan, *Analysis of an electronic voting protocol in the applied pi calculus*, in: S. Sagiv, editor, *Proc. ESOP 2005*, 2005, pp. 186–200.

[36] Lipmaa, H., *Secure electronic voting protocols*, in: H. Bidgoli, editor, *The Handbook of Information Security* (2006).

[37] Mauw, S., J. Verschuren and E. de Vink, *A formalization of anonymity and onion routing*, in: P. Samarati, P. Ryan, D. Gollmann and R. Molva, editors, *ESORICS'04* (2004), pp. 109–124.

[38] Motwani, R. and P. Raghavan, "Randomized algorithms," Cambridge University Press, 1995.

[39] Nielsen, C., E. Andersen and H. Nielson, *Static validation of a voting protocol*, in: P. Degano and L. Viganó, editors, *Proc. ARSPA 2005)* (2005), pp. 115–134.

[40] Okamoto, T., *Receipt-free electronic voting schemes for large scale elections*, in: B. Christianson, B. Crispo, T. Lomas and M. Roe, editors, *Proc. Security Protocols Workshop '97* (1998), pp. 25–35.

[41] Park, C., K. Itoh and K. Kurosawa, *Efficient anonymous channel and all/nothing election scheme*, in: T. Helleseth, editor, *Proc. Eurocrypt '93* (1993), pp. 248–259.

[42] Pfitzmann, A. and M. Köhntopp, *Anonymity, unobservability, and pseudonymity*, in: H. Federrath, editor, *Designing Privacy Enhancing Technologies* (2001), pp. 1–9.

[43] Pfitzmann, B., *Breaking an efficient anonymous channel*, in: L. Guillou and J.-J. Quisquater, editors, *Proc. Eurocrypt '94* (1995), pp. 332–340.

[44] Reiter, M. and A. Rubin, *Crowds: Anonymity for web transactions.*, ACM Transactions on Information and System Security **1** (1998), pp. 66–92.

[45] Rjaskova, Z., *Electronic voting schemes*, http://people.ksp.sk/~zuzka/elevote.pdf (2002).

[46] Schneider, S. and A. Sidiropoulos, *CSP and anonymity*, in: *Proc. ESORICS'96* (1996), pp. 198–218.

[47] Serjantov, A. and G. Danezis, *Towards an information theoretic metric for anonymity*, in: H. Federrath, editor, *Proc. PET 2002* (2003), pp. 41–53.

[48] Shmatikov, V., *Probabilistic analysis of an anonymity system.*, Journal of Computer Security **12** (2004), pp. 355–377.

[49] Steinbrecher, S. and S. Köpsell, *Modelling unlinkability*, in: R. Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)* (2003), pp. 32–47.

[50] Syverson, P., D. Goldschlag and M. Reed, *Anonymous connections and onion routing*, in: *IEEE Symposium on Security and Privacy*, Oakland, California, 1997, pp. 44–54.

[51] Syverson, P. and S. Stubblebine, *Group principals and the formalization of anonymity*, in: J. Wing, J. Woodcock and J. Davies, editors, *Proc. FM'99* (1999), pp. 814–833.

[52] Syverson, P., G. Tsudik, M. Reed and C. Landwehr, *Towards an analysis of onion routing security*, in: H. Federrath, editor, *Designing Privacy Enhancing Technologies* (2001), pp. 96–114.