Knowledge-based modelling of voting protocols

A. Baskar[*]
Chennai Mathematical Institute
Chennai, India

R. Ramanujam
Institute of Mathematical Sciences
Chennai, India

S.P. Suresh
Chennai Mathematical Institute
Chennai, India

Abstract

We contend that reasoning about knowledge is both natural and pragmatic for verification of electronic voting protocols. We present a model in which desirable properties of elections are naturally expressed using standard knowledge operators, and show that the associated logic is decidable (under reasonable assumptions of bounded agents and nonces).

1 Summary

Consider the following scenario: an election is announced and is to take place on a given date. Voters are given a choice of candidates, and this is a standard political election, with each voter allowed to vote for exactly one candidate. On election day, during a specified period, voters can exercise their franchise in designated voting stations, or (here is the difference), *vote online*. What are the risks and vulnerabilities associated with such an electronic election? Electronic voting protocols address such issues and pose solutions. See [Rja02] for a discussion of cryptographic schemes intended as solutions to these problems.

Note that such elections are far from hypothetical. The 2005 general elections in Estonia offered such an on-line voting option. Moreover, apart from political elections for office, many corporate decisions (regarding projects, election to positions) require voting where bringing voters together to one place to conduct the election is expensive, and online elections are either being used or being considered in such contexts. In any case, such processes are to be anticipated, and hence articulating properties that **must** be ensured of such mechanisms as well as designing means of verifying such properties of proposed mechanisms becomes critical. Formal models of electronic elections become relevant and important in this regard, and their study can reveal vulnerabilities in current (non-electronic) elections as well.

What have elections got to do with theories of knowledge and rationality? Before we answer this pertinent question, we request the reader's indulgence and list *some* desirable properties in election mechanisms ([Rja02]):

Secrecy Every voter's choice should be private, and others should not be able to figure out how she voted.

Receipt-freeness No voter has any means of proving to another that he has voted in a particular manner.

Fairness Voters do not have any knowledge of the distribution of votes until the tallies are finally announced.

Individual verifiability Each voter should be able to check whether her vote has been counted properly.

The list is merely indicative and by no means exhaustive. While all these properties relate to the knowledge of agents at different stages of the election when it is under way, receipt-freeness is especially interesting. First introduced in [BT94], this is crucial, since lack of receipt-freeness (that is, the presence of a receipt) allows vote buying and coercion which has the potential to drastically affect the election process.

This property is an assertion about mutual knowledge (and in a subtle way, common knowledge as well). When a receipt exists, it could be constructed in any manner that convinces the sceptical second party. Demonstrating that no such way exists is highly demanding, and is seen as a significant challenge to formal models. It is here that knowledge theory helps: based on what is common knowledge, and how every agent's knowledge is updated after every event in the election, we can place limits on what *can* be known (in principle), and hence, what kind of proofs can be constructed by voters. Specifically, this allows us to show what kind of receipts a voter may construct. Moreover, by making the *perfect encryption assumption*, we can carry

[*] Supported by the Council of Scientific and Industrial Research (CSIR), India

out the reasoning in an abstract plane, and reduce the security of the election to that of the underlying cryptographic schemes.

Our contention is that articulating receipt-freeness in terms of knowledge of agents is not only natural, but also that it shows the way for how we might verify such a property in a system. Interestingly, since the reasoning mainly involves placing limits to constructible knowledge, we can avoid many of the philosophical pitfalls associated with epistemic reasoning about security (see [RS05] for a discussion of such issues). Moreover, while we do not elaborate on this here, there are implications for rational agents as well: for instance, when the mechanism allows the possibility of certain kinds of receipts, there are incentives for voters to use them and rational voters would choose their votes accordingly. In a population of voters, when a subset can be assumed to vote thus, the choices of each member of that subset would be influenced. We observe this merely to remark that knowledge theoretic accounts of elections may offer implications for design of voting systems as well.

The main contribution of this paper is to set up a formal model for electronic voting protocols in such a way that associated security properties are easily seen to be assertions on agents' knowledge. The modelling of knowledge itself is standard, and based on agent indexed equivalence relations on agents' information states (which are given by sets of terms that an agent can construct using a proof system). Thus, while the model is standard (as found in [PR85] and [FHMV95], for example), it has elements similar in spirit to [AN05]. For a discussion of the underlying security protocol model, refer to [DY83], [CDL+99], and [RS06]. For work on knowledge-based modelling of security protocols, refer to [HP03], [RS05], and [HO05].

A feature of this model may be of interest to security theorists: while the protocols themselves are expressed in the Dolev Yao model, the knowledge operators work on a more basic *computational* model in the sense of Abadi and Rogaway [AR00]. That is, the algebra of terms used in messages uses encryption, pairing etc as operators, whereas knowledge capabilities of agents are defined only in terms of bit strings seen by agents rather than term structure. Thus when an agent receives a term $\{t\}_k$ encrypted with k and has no knowledge of k or $inv(k)$, she cannot distinguish it from any other term $\{t'\}_{k'}$; indeed, she cannot even be assumed to know that this is an encrypted term, it might be channel noise.

We also present a formal logic of knowledge in which these properties are expressed and show that verification of these properties for a protocol is decidable under reasonable assumptions (of bounded number of agents and nonces). However, we intend this to be only a proof of genericity, that such a decision procedure can be obtained in a generic form, rather than advocate a specific set of logical operators. We hope that further work on principles of reasoning with security primitives will lead to the "right" logic of knowledge for security protocols.

Related work

While electronic voting protocols are well known in the literature, there have been only a few attempts at formal models and verification. Recently Kremer and Ryan [KR05] modelled the FOO protocol [FOO92] using applied pi calculus and expressed receipt-freeness as an observational equivalence. In this sense, this work is similar to ours, though their emphasis is on studying how concurrency and communication mechanisms affect modelling. Another similar work is [DKR06], where a stronger notion of receipt-freeness (known as coercion-resistance) is defined and its relationship with receipt-freeness and privacy discussed using pi calculus. [CMFP+06] shows that simultaneously achieving universal verifiability and receipt freeness is impossible in general.

Our work is closest in spirit to that of [JdV06] and [JP06]: in the former, [JdV06], a generic and uniform formalism is given to define the notion of receipt and applied to check receipt-freeness for some protocols. In the latter, receipt-freeness is expressed in a manner very similar to ours, but using indistinguishability relations associated with anonymity. In a sense, our work can be seen as a general framework for a formal analysis of such a class of properties. Moreover, the emphasis on a decision procedure distinguishes our treatment from theirs.

2 The formal model

Before we present the formal model, we give an informal description of how electronic voting protocols work. As one may expect, such a system consists of three kinds of agents: voters, administrators and talliers. Administrators know the voters' identity, but cannot see the votes, and their job is to check voters' eligibility to vote. Talliers see the votes, but not the voters' identities, and their job is to count votes for each candidate and announce the result. Voting protocols (typically) use one of two cryptographic mechanisms: homomorphic encryption and blind signature.

Homomorphic encryption refers to a secret sharing scheme by which a secret is split into several parts, and cannot be reconstructed without getting access to (almost) all the parts. In protocols using this scheme, voters split their votes into several shares which they send to administrators. Hence, unless many of he administrators collude, it is difficult to reconstruct the vote.

The blind signature scheme ([Cha83], [Cha85]) involves the ability of an agent A to generate, from a term $\{t\}_k$, a signed term $\{t_A\}_k$, even without having access to k or $inv(k)$. In protocols using this scheme, an administrator receives

an encrypted vote from a voter, and without being able to decrypt the vote, verifies the voter's eligibility, "blindly" signs the message and returns it. Now the voter, who generated k, can strip it off from $\{t_A\}_k$ obtaining t_A, which is the vote t duly attested by A. This is then sent anonymously to the tallier, who verifies the attestation, and gets the vote t (while not knowing the origin).

Another important and relevant detail is that of *commitment*: once a voter commits to making a choice, she should not be able to change her decision. For instance, in the blind signature scheme above, a voter should not be able to get an attestation for vote 0 from an administrator but send 1 to the tallier.

We now proceed to formalize these notions. The formal model we present here is essentially the same as the one proposed for security protocols in [RS05] and refined further in [RS06]. The extensions involve specific primitives intended to model the features discussed above.

Terms and derivations

Fix a finite set of **agents** Ag, which includes the set of **voters** V, the set of **authorities** A, a **counter** C, a **server** S, and the **intruder** I. As one can guess, authorities are those who verify whether an agent attempting to cast a ballot is indeed a registered voter and entitled to vote, and a counter sums up votes for each candidate. The server is an abstraction that allows us to hide details of how keys are generated and stored. (When you need a key, ask the server.) The intruder is again an abstraction that quantifies over the malicious forces at work to compromise security. The intruder is assumed to have unbounded memory, has access to all that travels on the public channel, can forge and block messages. It suffices to consider only one intruder (see [CMS00]).

Fix a countable set of **fresh secrets** N. (This includes *random, unguessable nonces* as well as *temporary session keys*.) Let Ch be the **choices** of the voting scheme. To keep our analysis simple, we consider only Yes-or-No voting schemes. Here $Ch = \{0, 1\}$. $\mathcal{B} \stackrel{\text{def}}{=} N \cup Ag \cup Ch$ is the set of **basic terms**. We define the set of (potential) keys, $\mathcal{K}$, to be $N \cup \{pub(A), priv(A), sk(A, S) \mid A \in Ag\}$. Here $pub(A)$, $priv(A)$, and $sk(A, B)$ denote the public key of A, private key of A, and (long-term) shared key between A and B.

The set of **information terms** is defined to be

$$\mathcal{T} \quad ::= \quad m \mid k \mid (t_1, t_2) \mid \{t_1\}_k \mid [t_1, t_2]$$

where m ranges over $\mathcal{B}$, k ranges over $\mathcal{K}$, A ranges over Ag, and t_1 and t_2 range over $\mathcal{T}$. We define $inv(k)$ for every $k \in \mathcal{K}$ as follows: $inv(pub(A)) = priv(A)$, $inv(priv(A)) = pub(A)$, and $inv(k) = k$ for every other $k \in \mathcal{K}$.

Further, (t_1, t_2) denotes the pair consisting of t_1 and t_2, and $\{t_1\}_k$ denotes the term t_1 encrypted using k. The term $[t_1, t_2]$ denotes a different kind of pairing that we call **blind pairing**.

When agents communicate, they should have the ability to generate new messages from their current knowledge (which includes their initial knowledge and the messages they have previously received). We now define the system of rules for deriving new messages from old.

A **sequent** is of the form $T \vdash t$ where $T \subseteq \mathcal{T}$ and $t \in \mathcal{T}$. A **derivation** or a **proof** π of $T \vdash t$ is a tree whose nodes are labelled by sequents and connected by one of the *analz*-rules in Figure 1 and the *synth*-rules in Figure 2; whose root is labelled $T \vdash t$; and whose leaves are labelled by instances of the Ax rule. We will use the notation $T \vdash t$ to denote both the sequent, and the fact that it is derivable. For a set of terms T, $\overline{T} \stackrel{\text{def}}{=} \{t \mid T \vdash t\}$ is the *closure* of T.

$$\frac{}{T \cup \{t\} \vdash t}\ Ax$$

$$\frac{T \vdash (t_1, t_2)}{T \vdash t_i}\ split_i (i = 1, 2)$$

$$\frac{T \vdash \{t_1\}_k \qquad T \vdash inv(k)}{T \vdash t_1}\ decrypt$$

$$\frac{T \vdash [t_1, t_2] \qquad T \vdash t_i}{T \vdash t_{3-i}}\ blindsplit_i (i = 1, 2)$$

Figure 1: Analysis rules

$$\frac{T \vdash \{[t, \{m\}_{inv(k)}]\}_k}{T \vdash [\{t\}_k, m]}\ blindsign$$

$$\frac{T \vdash t_1 \qquad T \vdash t_2}{T \vdash (t_1, t_2)}\ pair$$

$$\frac{T \vdash t_1 \qquad T \vdash k}{T \vdash \{t_1\}_k}\ encrypt$$

$$\frac{T \vdash t_1 \qquad T \vdash t_2}{T \vdash [t_1, t_2]}\ blindpair$$

Figure 2: Synthesis rules

The *blindsign* rule is a new kind of rule in this system. To appreciate it, we need to first consider the implementability of the operators we have introduced. The standard implementation treats terms as numbers, a pair (t, t') as concatenation of the numbers representing t and t' (with the numbers viewed as bit strings), $\{t\}_k$ as the result of applying any of the standard public/shared key cryptographic algorithms using the numbers representing t and k (in many implementations this is just raising t to the k^{th} power modulo some prime), and $[t, t']$ as the product of the numbers representing t and t'. It can be seen that our proof rules are valid under this interpretation. The *blindsign* rule represents the commonly used blind signature scheme, where $inv(k)$ is chosen to be such that $r^{k \cdot inv(k)} = r \mod p$ for some prime p, and hence $(q \cdot r^{inv(k)})^k = q^k \cdot r \mod p$. But there is

a caveat: the *blindsign*-rule, is restricted to the cases where $m \in \mathcal{B}$. Note that the following more general version of the *blindsign* rule, where t and t' are arbitrary terms, is also valid under the standard interpretation:

$$\frac{T \vdash \{[t, t']\}_k}{T \vdash [\{t\}_k, \{t'\}_k]}\ blindsign$$

But we stick to the simpler version since this is the most common use of the *blindsign* rule in practice, and since the more general version introduces proof-theoretic difficulties which will detract attention from the main thrust of the paper.

In general, the state of an agent (to be formally defined below) will be a finite set of terms T, and $\overline{T}$ will be the set of terms that this agent can generate in that state, and hence compose and use in messages. Thus, $\overline{T}$ represents the explicit knowledge of data possessed by the agent at that state, and the following lemma (whose detailed proof is presented in the Appendix) is hence important.

Theorem 1 *Given a finite set of terms T and a term t, checking whether $t \in \overline{T}$ is decidable in time polynomial in size of T.*

In general, the message-generation capabilities of different kinds of agents will be different. Typically, an authority has more power than a normal voter. For instance, we can define an authority to be one who uses all the proof rules in the system presented in Figure 1 and Figure 2, whereas a voter is one who cannot apply the *blindsign* rule in deriving new messages from old.

Protocols and their runs

We model communication between agents by **actions**. An action is either a **send action** of the form $+(A, B, t)$ or a **receive action** of the form $-(A, B, t)$, or an **anonymous send** of the form $!(A, B, t)$, or an **anonymous receive** of the form $?(*, B, t)$, where t is an arbitrary term, and A and B are agent names.

We emphasize that while the sender name in a send action, and a receiver name in a receive action denote the actual agents that send and receive the messages, respectively, in a send action we can only name the *intended receiver*, and in a receive action we can only name the *purported sender*. Further, in an anonymous receive, B does not even have any indication of the purported sender. As we will see later, every send action is an instantaneous receive by the intruder, and similarly, every receive action is an instantaneous send by the intruder. *Broadcast communications* are also widely used in electronic voting protocols in practice. For simplicity, we do not model it explicitly, though we use it in some of the examples. The model can be easily extended to handle this.

A **protocol** is just a finite set of **parametrized roles** $\{\eta_1, \ldots, \eta_n\}$. A parametrized role $\eta[m_1, \cdots, m_k]$ is a finite sequence of actions in which the *basic terms* $m_1, \ldots, m_k$ are singled out as parameters. The idea is that an agent participating in the protocol can execute many *sessions* of a role in the course of a single run, by instantiating the parameters in many different ways.

Typically protocols are presented as a sequence of communications of the form $A \to B : t$, which denotes the sending of the message t by A and its receipt by B. To formally model the fact that the intruder can block messages, and also fake messages, one typically extracts the actions (send or receive) of each agents from such a sequence of communications, and considers interleavings of various sessions of the roles. We directly present protocols as sets of such action sequences.

While modelling election protocols, we need to consider three phases of any agent's role. The first phase consists of the message exchanges needed to obtain the necessary keys from the server for further communication. We think of this as the pre-election phase.

Then there is the election phase itself, prescribed by the protocol. Specific to voting protocols is the third phase, the post-election phase. We need to consider this for a faithful modelling and verification of voting protocols. The point is that in any election the voters can reveal certain information and gain certain benefits much after the election process. Think of somebody being able to prove that she voted for a particular candidate and claim her reward. We need to consider such capabilities as part of the protocol, though not of the election itself, so that the verification guarantees (of crucial properties like receipt-freeness) that we provide are meaningful.

So, technically each role η consists of three parts η_1, η_2, and η_3. But we treat it as a sequence of actions, and in examples we present only η_2, for simplicity of notation. η_1 is mostly standard. We will let the reader infer η_3 from the context. This doesn't affect the technical details of the model or the results.

Runs of a protocol

We define the semantics of a protocol in this subsection. It is given by the set of its runs. Informally, a run is got by interleaving various sessions of the protocol, where a session of the protocol is just a role being played out by some agent with a particular instantiation of the parameters. Furthermore, this interleaving should be *admissible* in that the messages communicated at any stage by an agent should be constructible by him or her using the current knowledge. We formalize all these details (succinctly) below. More details can be seen in [RS06], for instance.

A **substitution** σ is a map from $\mathcal{B}$ to $\mathcal{T}$ such that $\sigma(Ag) \subseteq$

Ag and $\sigma(I) = I$ and $\sigma(N) \subseteq N$. The notion is extended to arbitrary terms, actions, etc in the obvious manner.

An **event** of a protocol Pr is a triple $e = (\eta, \sigma, lp)$ where η is a role of Pr, σ is a substitution suitable for Pr and σ, and $1 \leq lp \leq |\eta|$. For events $e = (\eta, \sigma, lp)$ and $e' = (\eta', \sigma', lp')$ of Pr, we say that $e \prec e'$ (meaning that e is in the *local past* of e') if $\eta = \eta'$, $\sigma = \sigma'$, and $lp < lp'$.

An **information state** (or just *state*) is a tuple $(s_A)_{A \in Ag}$, where $s_A \subseteq \mathcal{T}$ for each $A \in Ag$. The **initial state** of Pr, denoted by $init(Pr)$, is the tuple $(s_A)_{A \in Ag}$ such that for all $A \in Ag \setminus \{S\}$,

$$s_A = Ag \cup Ch \cup \{priv(A), pub(A), pub(S), sk(A, S)\},$$
$$s_S = Ag \cup Ch \cup \{priv(S), pub(A), sk(A, S) \mid A \in Ag\}.$$

The idea is that the server initially holds all the keys, and the agents have to request it for the appropriate keys.

The notions of an action **enabled** at a state, and $update(s, a)$, the **update** of a state s on an action a, are defined as follows:

A send action a is always enabled at any state s.
A receive action a is enabled at s iff $term(a) \in \overline{s_I}$.
$update(s, +(A, B, t)) = update(s, !(A, B, t)) \stackrel{\text{def}}{=} s'$
  where $s'_I = s_I \cup \{t\}$, $s'_C = s_C$ for $C \neq I$.
$update(s, -(A, B, t)) = update(s, ?(*, B, t)) \stackrel{\text{def}}{=} s'$
  where $s'_B = s_B \cup \{t\}$ and $s'_C = s_C$ for $C \neq B$.

$update(s, \eta)$ for a state s and a sequence of actions η is defined in the obvious manner. Given a protocol Pr and a sequence of its events ξ, $infstate(\xi)$ is defined to be $update(init(Pr), act(\xi))$.

Given a protocol Pr, a sequence $e_1 \cdots e_k$ of events of Pr is said to be an **admissible sequence of events** of Pr iff the following conditions hold:

for all $i, j \leq k$ such that $i \neq j$, $e_i \neq e_j$,
for all $i \leq k$ and for all $e \prec e_i$,
  there exists $j < i$ such that $e_j = e$, and
for all $i \leq k$, $act(e_i)$ is enabled at $infstate(e_1 \cdots e_{i-1})$.

Usually, runs are also required to satisfy the property of *unique origination of nonces*. This means that there is a designated set of nonces in the protocol specification which are meant to be *fresh*, and that if we use substitute one value for one of these nonces in one context, the same value cannot be used for any other designated nonce or for the same nonce in a different context. The formal details can be found in [RS06].

Modelling primitives of election protocols

We now discuss examples of primitives used in election protocols in terms of the formal model.

Blind signatures. Suppose Ammu wants to get Balu to sign a message m for her, without revealing m to Balu. This can be done as follows. Ammu sends $[m, \{r\}_{pub(B)}]$ to Balu, where r is a some random number chosen by Ammu. Now Balu signs this message to get $\{[m, \{r\}_{pub(B)}]\}_{priv(B)}$. He will now apply the *blindsign* rule and get $[\{m\}_{priv(B)}, r]$. From this, Balu cannot get m as he does not know r. But on receiving this message, Ammu can get $\{m\}_{priv(B)}$ using the *blindsplit* rule (since she has r).

Bit commitment. Suppose Ammu wants to commit a bit to Balu now and reveal it later. She should not be allowed to change her mind in the meantime. This can be implemented as follows: Ammu first sends $\{b\}_k$ and later reveals $inv(k)$. Balu cannot know b unless he has k. Ammu cannot fool Balu into believing that the bit is some bit b' different from b. This is because of the following fundamental property of the Dolev-Yao abstraction: two terms $\{b\}_k$ and $\{b'\}_{k'}$ if and only if $b = b'$ and $k = k'$. Thus Ammu cannot find any k' such that sending $inv(k')$ to Balu, she can make him extract a b' different from b.

Now consider a variant of the bit commitment scheme, which we call **trapdoor bit commitment**. Ammu wants to commit two bits to Balu. Later, she wants to reveal *only one* of the committed bits to Balu. To implement this, Ammu sends $[\{b\}_k, \{b'\}_{k'}]$ to Balu. If she wishes to reveal b, she sends $(\{b'\}_{k'}, inv(k))$ to Balu. If she wishes to reveal b', she sends $(\{b\}_k, inv(k'))$. It is easy to show that exactly one of the bits is revealed to Balu.

3 A protocol example

We present an abstract version of the FOO protocol [FOO92] here, and show how it is formally specified in our model. This protocol is based on the blind signature scheme outlined in Section 2.

In formally modelling the protocol, we assume a bounded number of agents (say m, with n of them being voters). We have three roles: the voter role, the administrator role, and the counter role. We also assume that when an agent receives a message through an anonymous channel, she does not know the sender of the message. We use $*$ to denote the anonymous sender. Similarly, $\times(A, *, t)$ denotes the agent A broadcasting the message t.

The various roles are given below:

- The voter role:

  1. $+(V, A, \{[\{b\}_r, \{k\}_{pub(A)}]\}_{priv(V)})$
  2. $-(A, V, ([\{\{b\}_r\}_{priv(A)}, k])$
  3. $-(A, V,$ validation over!)
  4. $!(V, C, (\{\{b\}_r\}_{priv(A)}, r)$

  The voter first decides on her vote and applies a bit commitment scheme with a random number r. She

then sends it to the administrator to affix a blind signature using another random number k. On getting a blind signature from the administrator, she retrieves the signed message using her random number k, and then waits for the administrator to broadcast a message that starts the next stage of election. Now the voter sends the signed message (which is a bit commitment of her vote with r) to the counter through an anonymous channel.

- The administrator role

  1. $-(V_1, A, \{[\{b_1\}_{r_1}, \{k_1\}_{pub(A)}]\}_{priv(V_1)})$
  2. $+(A, V_1, [\{\{b_1\}_{r_1}\}_{priv(A)}, k_1])$
  ...
  $2n-1$. $-(V_n, A, \{[\{b_n\}_{r_n}, \{k_n\}_{pub(A)}]\}_{priv(V_n)})$
  $2n$. $+(A, V_n, [\{\{b_n\}_{r_n}\}_{priv(A)}, k_n])$
  $2n+1$. $\times(A, *,$ validation over!)

  The administrator's role is to receive encrypted votes from registered voters and return blind signatures to them. When this is done for all voters, the administrator announces the end of this stage of election. Note that, for simplicity, we do not present error conditions here.

- The counter role

  1. $-(A, C,$ validation over!)
  2. $?(*, C, (\{\{b'_1\}_{r'_1}\}_{priv(A)}, r'_1))$
  ...
  $n+1$. $?(*, C, (\{\{b'_n\}_{r'_n}\}_{priv(A)}, r'_n))$
  $n+2$. $\times(C, *,$ announce results)

  The counter simply collects anonymously mailed votes, and counts every vote that has been attested by an administrator. Again, we do not present error situations.

Actually, the FOO protocol is more complicated: after registration, voters send the signed votes to the counter, who then publishes (on a bulletin board, say) the list of bit commitments. This is an ordered list. Looking at this list, the voter identifies her commitment and sends to the counter the list number and r through an anonymous channel. This allows the counter to determine the actual votes and tally them properly. The counter can also publish the associated random number with the previously published list, so that anybody can check if the votes have been counted properly. We have avoided these communications to save clutter, but they can be modelled as well.

It is known that the FOO protocol attains secrecy, individual verifiability, fairness and eligibility. However it does not satisfy universal verifiability, which asserts that at the end of the election, we can check that all voters' votes have been counted. (This is because the counter can add votes according to his wish, if some voters refrain from voting.) The FOO protocol is not receipt-free either, and we discuss this in some detail, but later, when we have knowledge operators on hand.

4 Logic

We now consider the propositional logic of knowledge and tense as a logical language for specifying properties of voting protocols. While the modalities are standard, we considerably restrict the atomic propositions to be in a specific form, so that the knowledge assertions in this logic are limited to knowledge that can be constructed by agents. (See [RS05] for the rationale.) The semantics of the proposition A *has* t, given by $t \in \overline{T}$, where T is A's "current state" codifies constructible basic knowledge. The semantics of $K_A\ \alpha$ is based on an observational equivalence on runs from A's point of view, and represents the meaning that an agent attaches to constructible terms.

The set of formulas Φ is given by:

$$\Phi ::= A\ has\ t \mid a \mid vote_A(c) \mid \neg\alpha \mid \alpha \vee \beta \mid \mathbf{G}\ \alpha \mid \mathbf{H}\ \alpha \mid K_A\ \alpha$$

where $A \in Ag$, $t \in \mathcal{T}$, a is any action, $c \in Ch$ and α, β range over Φ. The other connectives, $\wedge$, $\supset$ and the dual modalities $\mathbf{F}$, $\mathbf{P}$, and L_A are defined in the usual manner. The atomic proposition $vote_A(c)$ says that in the current state A has decided on the option $c \in Ch$ as her vote.

The semantics of the logic crucially hinges on an equivalence relation on runs, which is defined as follows. Intuitively, an agent cannot distinguish a term $\{t\}_k$ from any other bitstring in a state where she has no information about k. A similar remark applies to a blind pair $[t_1, t_2]$ when she has neither of the terms inside.

We define the set *Patterns* of patterns as follows (where $\Box$ denotes an unknown pattern):

$$Patterns ::= b \in \mathcal{B} \mid (P, Q) \mid \{P\}_k \mid [P, Q] \mid \Box$$

where P, Q range over *Patterns*.

We can now define the patterns derivable by an agent on seeing a term t in the context of a set of terms S. In a sense, this is the only certain knowledge that the agent can rely on at that state.

$$pat(b, S) = \begin{cases} b & \text{if } b \in \mathcal{B} \cap S \\ \Box & \text{if } b \in \mathcal{B} \setminus S \end{cases}$$

$$pat((t_1, t_2), S) = (pat(t_1, S), pat(t_2, S))$$

$$pat(\{t\}_k, S) = \begin{cases} \{pat(t, S)\}_k & \text{if } inv(k) \notin \overline{S} \\ \Box & \text{otherwise} \end{cases}$$

$$pat([t_1, t_2], S) = \begin{cases} [pat(t_1, S), pat(t_2, S)] & \text{if } t_1 \in \overline{S} \\ [pat(t_1, S), pat(t_2, S)] & \text{if } t_2 \in \overline{S} \\ \Box & \text{otherwise} \end{cases}$$

We extend the definition to $pat(a, S)$ and $pat(e, S)$ for an action a, event e and a set of terms S in the obvious manner. For any sequence of actions $\xi = e_1 \cdots e_n$, we define $pat(\xi, S)$ to be the sequence $pat(e_1, S) \cdots pat(e_n, S)$.

An agent A's *view* of a run A, denoted $\xi\restriction A$, is the sequence got by retaining the first occurrence (in order) of every action a of A. For two runs ξ and ξ' of Pr and an agent A, we define ξ and ξ' to be A-equivalent (in symbols $\xi \sim_A \xi'$) iff $pat(\xi\restriction A, S) = pat(\xi'\restriction A, S')$, where $S = infstate(\xi)$ and $S' = infstate(\xi')$.

The semantics of the logic can now be given on standard lines, where the formulas are evaluated at an instant of a run of the protocol. The inductive definition is as follows (where we use the notations $\xi(i)$ and ξ_i for the i^{th} event of ξ and the prefix of length i of ξ, respectively):

$\xi, i \models A$ *has* t if and only if $t \in \overline{infstate_A(\xi_i)}$.
$\xi, i \models a$ if and only if $act(\xi(i)) = a$.
$\xi, i \models vote_A(c)$ if and only if $V(\xi, A) = c$.
$\xi, i \models \mathbf{G}\ \alpha$ if and only if $\xi, j \models \alpha$ for all $j \geq i$.
$\xi, i \models \mathbf{H}\ \alpha$ if and only if $\xi, j \models \alpha$ for all $j \leq i$.
$\xi, i \models K_A\ \alpha$ if and only if $\xi', i' \models \alpha$
for all ξ', i' such that $(\xi, i) \sim_A (\xi', i')$.

We say that $Pr \models \alpha$ if for all runs ξ of Pr and all instants $i \leq |\xi|$, $\xi, i \models \alpha$.

Properties

We now consider the properties of election protocols discussed in Section 1, and show how they can be expressed in this logic.

Secrecy. A protocol is said to preserve secrecy (of votes) if the intruder cannot figure out anyone's vote. This is specified by:

$$\bigwedge_{A \in Ag, c \in Ch} (vote_A(c) \supset \neg\ K_I vote_A(c)).$$

Receipt-freeness: This asserts that no voter has any means of proving to another agent that she has voted in a particular manner. It is surprisingly simple to express in terms of our logic. Consider a voter A and a run ξ of a protocol Pr. We want to say that no other agent B "looking at" ξ can determine for certain what A's vote (A might have tried her best to communicate (usually indirect) information about her vote to B, but even so). The formula is as follows:

$$\bigwedge_{c \in Ch} (vote_A(c) \supset \bigwedge_{B \neq A} \neg\ K_B vote_A(c)).$$

It is to be noted that such a definition might not be very effective in practice. For instance, if B can convince A that B voted in a particular way with 99% certainty, then we should deem that B has a receipt, even though our model might not recognise the situation as such. We haven't addressed this subtle issue in this paper, and leave it for future research.

Fairness: A protocol is said to be fair if the voter is prevented from changing her vote as a consequence of partial results. This often means that the voters do not have any knowledge of the distribution of the votes until they are finally announced. Here is one version of this property formalized (where we assume a special atomic proposition *ann*, true exactly of runs in which a special announce action has happened in the past):

$$\neg ann \supset \bigwedge_{A \in Ag} (L_A(\bigwedge_{B \neq A} vote_B(0)) \wedge L_A(\bigwedge_{B \neq A} vote_B(1))).$$

Individual verifiability: Each voter should be able to check whether her vote has been counted properly. Fix a voter V and a counter C:

$$\bigwedge_{c \in Ch} (!(V, C, \{c\}_r) \supset \mathbf{G} K_A(ann \supset \mathbf{P}?(V, C, \{c\}_r))).$$

Receipt in FOO

Consider the FOO protocol as modelled in the previous section, and a situation where there are two voters V_1 and V_2, who engage in a normal session of the protocol with the administrator and the counter. We present only the send messages in the run. It is implicit that the corresponding receive events happen immediately.

$V_1 \cdot 1.$	$+(V_1, A, \{[\{0\}_{r_1}, \{k_1\}_{pub(A)}]\}_{priv(V_1)})$
$A \cdot 1.$	$+(A, V_1, [\{\{0\}_{r_1}\}_{priv(A)}, k_1])$
$V_2 \cdot 1.$	$+(V_2, A, \{[\{1\}_{r_2}, \{k_2\}_{pub(A)}]\}_{priv(V_2)})$
$A \cdot 2.$	$+(A, V_2, [\{\{1\}_{r_2}\}_{priv(A)}, k_2])$
$A \cdot 3.$	$\times(A, *, \text{validation over!})$
$V_2 \cdot 2.$	$!(V_2, C, (\{\{1\}_{r_2}\}_{priv(A)}, r_2)$
$V_1 \cdot 2.$	$!(V_1, C, (\{\{0\}_{r_1}\}_{priv(A)}, r_1)$
$C.$	$\times(C, *, \text{announce results})$

Now V_1 has voted 0, V_2 has voted 1, and the counter has announced the results. At the end of the run $\neg K_A vote_{V_1}(0)$

is true. This is because A can only see the pattern $\Box$ on receving the first message. Therefore there is another run of the protocol A-equivalent to the above in which V_1 could have voted differently. Thus A cannot know V_1's vote. But we claim that (k_1, r_1) is a "receipt" for the voter V_1 with respect to the administrator. By this we mean the following: if V_1 manages to send the pair (k_1, r_1) to the administrator (in violation of the protocol), A can now discern a deeper pattern in the term he received first. In fact A can determine that term completely (though not at the time of receiving the first message).

It is easy to see that, after the receipt of (k_1, r_1), there is no run A-equivalent to the above in which V_1 could have voted differently. This is because of a fundamental property of the basic Dolev-Yao model: $\{t\}_k = \{t'\}_{k'}$ if and only if $t = t'$ and $k = k'$, and $[t, t'] = [u, u']$ if and only if $t = t'$ and $u = u'$. Therefore the vote in the first message can only be 0 in all equivalent messages.

This shows that the FOO protocol does not satisfy receipt-freeness.

There is more: when V_1 sends (k_1, r_1), we can show that $K_{V_1} \neg K_A vote_{V_1}(0)$ holds, but that after it is received, $K_{V_1} K_A vote_{V_1}(0)$ holds. This reveals the intentionality of receipt as well.

This scenario highlights another feature: the formula $\mathbf{F} K_A \alpha \supset K_A \mathbf{F} \alpha$ is not a validity; while the administrator knows the receipt at the end of the run, and hence the knowledge formula is initially true, the future receipt is not known initially. This is unlike typical logics of knowledge and time, and illustrates an aspect typical of knowledge in the context of security protocols.

Decidability

Theorem 2 *Fix a finite $T \subseteq \mathcal{T}_0$. The problem of checking for a given election protocol Pr and a formula α in Φ, whether all T-runs of Pr satisfy α (in symbols $Pr \models^T \alpha$), is decidable.*

Note that the set of runs is infinite even though the set of basic terms is finite, since events may repeat in a run. (Modelling such repetition is forced on us, since an agent cannot distinguish between repeated events and distinct events where she sees the same patterns.)

The proof of the theorem proceeds along lines similar to the one described in [RS05]. We construct an atom graph for a given formula, whose nodes are locally consistent sets of formulas. The edge relation relates to the tense modalities and an equivalence relation on nodes (for each agent) is defined using patterns as above. We then need to define a reduce relation which collapses repeated events (based on subformulas), and construct a larger graph. Satisfiability then reduces to the existence of a good subgraph which satisfies certain closure conditions: crucially, paths in the subgraph are closed with respect to eventuality requirements, and those imposed by the L_A operator dual to knowledge. Verifying the existence of such a subgraph crucially relies on Theorem 1 which asserts efficient decidability of $T \vdash t$.

As a corollary to the theorem above, we see that checking receipt freeness is decidable for electronic voting protocols, when agents and nonces are assumed to be bounded. A direct proof of this result is much simpler, and uses the fact that under the assumptions made. The equivalence relation on runs is of finite index. Each equivalence class for agent B is further partitioned into two, for each agent $A \neq B$: those in which A votes 0 and those in which A votes 1. Verifying receipt freeness amounts to checking that each such partition is non-empty. We take the simplicity of this proof as further demonstration of our contention that knowledge based modelling offers a pragmatic basis for reasoning about receipt-freeness.

5 Conclusion

We have shown that the implementation of electronic voting schemes can be fruitfully studied using aspects of the theory of knowledge. This paper is in the nature of a preliminary study, where the model is set up and the basic decidability questions are addressed. Much more work needs to be done. For instance, the decidability result presented in this paper is only for a bounded number of agents, nonces, etc. The next step is to extend the decidability result to more general settings.

References

[AN05] S. Artemov and E. Nogina. On epistemic logic with justification. In *Proceedings of TARK X*, pages 279–294, Singapore, July 2005.

[AR00] Martin Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). In *Proceedings of the IFIP International Conference on TCS (IFIP TCS2000)*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22, 2000.

[BT94] Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections(extended abstract). In *Proceedings of 26th Symposium on Theory of Computing*, pages 544–553, 1994.

[CDL+99] Iliano Cervesato, Nancy A. Durgin, Patrick D. Lincoln, John C. Mitchell, and Andre Scedrov. A Meta-notation for Protocol Analysis. In P. Syverson, editor, *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, pages 35–51. IEEE Computer Society Press, 1999.

[Cha83] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology - Crypto '82*, pages 199–203. Springer-Verlag, 1983.

[Cha85] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[CMFP+06] Benoit Chevallier-Mames, Pierre-Alain Fouque, David Pointcheval, Julien Stern, and Jacques Traore. On Some Incompatible properties of Voting Schemes. In *Proceedings of the IAVoSS Workshop on Trustworthy Elections*, 2006.

[CMS00] Iliano Cervesato, Catherine A. Meadows, and Paul F. Syverson. Dolev-Yao is no better than Machiavelli. In P. Degano, editor, *Proceedings of WITS'00*, pages 87–92, July 2000.

[DKR06] Stephanie Delaune, Steve Kremer, and Mark Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *19th Computer Security Foundations Workshop*, pages 28–42. IEEE Computer Society, 2006.

[DY83] Danny Dolev and Andrew Yao. On the Security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.

[FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. M.I.T. Press, 1995.

[FOO92] Atsushi Fujioka, Tatsuaki Okamoto, and Kaazuo Ohta. A practical secret voting scheme for large scale elections. In *ASIACRYPT*, pages 244–251, 1992.

[HO05] Joseph Y. Halpern and Kevin R. O'Neil. Anonymity and Information Hiding in Multiagent Systems. *Journal of Computer Security*, 13(3):483–512, 2005.

[HP03] Joseph Y. Halpern and Riccardo Pucella. Modeling adversaries in a logic for security protocol analysis. In *Formal Aspects of Security, First International Conference, FASec 2002*, volume 2629 of *Lecture Notes in Computer Science*, pages 115–132, 2003.

[JdV06] Hugo Jonker and E.P. de Vink. Formalising Receipt-Freeness. In *Information Security Conference*, volume 4176 of *Lecture Notes in Computer Science*, pages 476–488. Springer, 2006.

[JP06] Hugo Jonker and Wolter Pieters. Receipt-freeness as a special case of anonymity in epistemic logic. In *Proceedings of the IAVoSS Workshop on Trustworthy Elections*, 2006.

[KR05] Steve Kremer and Mark Ryan. Analysis of an Electronic Voting Protocol in the Applied Pi Calculus. In *Proceedings of the European Symposium on Programming*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200. Springer, 2005.

[PR85] Rohit Parikh and R. Ramanujam. Distributed Processes and the Logic of Knowledge. In *Logic of Programs*, pages 256–268, 1985.

[Rja02] Zuzana Rjaskova. Electronic voting schemes. Master's thesis, Comenius University, 2002.

[RS05] R. Ramanujam and S. P. Suresh. Decidability of context-explicit security protocols. *Journal of Computer Security*, 13(1):135–165, 2005.

[RS06] R. Ramanujam and S. P. Suresh. A (restricted) quantifier elimination for security protocols. *Theoretical Computer Science*, 367:228–256, 2006.

A The decidability of the message derivation system

The core of the decidability argument for any logic is to prove that for a given term t and a set of terms T, the problem of checking whether $T \vdash t$ is decidable. We do that in this section. There is a preliminary definition first:

Define $st(t)$, the set of subterms of a term t, in the standard manner, with the the following additional clause: $st([t, \{m\}_k]) \stackrel{\text{def}}{=} \{[t, \{m\}_k], [\{t\}_{inv(k)}, m]\}$ $\cup st(\{m\}_k) \cup st(\{t\}_{inv(k)})$, and similarly for $[\{m\}_k, t]$.

For a set of terms T, $st(T)$ is defined to be $\bigcup_{t \in T} st(t)$.

Define $|t|$, the size of a term t, to be the number of symbol occurrences in t. It is easy to see that $|st(t)| \leq 7 \cdot |t|$, and that $|ST(T)| \leq 7 \cdot |T|$, where by $|T|$ we mean the sum of the sizes of the terms in T.

For ease of presentation, we will assume that the terms in $T \cup \{t\}$ are *normal*, i.e., they do not contain a term of the form $\{[t, \{m\}_{inv(k)}]\}_k$ as a subterm. This assumption allows us to modify the *blindsign* rule to the following form:

$$\frac{T \vdash [t, \{m\}_{inv(k)}] \qquad T \vdash k}{T \vdash [\{t\}_k, m]}\ blindsign$$

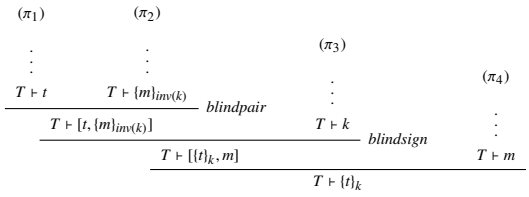
A *normal proof* is a proof π such that there is no shorter proof with the same conclusion as π. Notice that every subproof of a normal proof is a normal proof, and that no proper subproof of a normal proof π has the same conclusion as π. The following observations are useful:

Two successive applications of the *blindsign* rule cannot occur in a normal proof. The proof is as follows: Suppose a normal proof π ends with two successive applications of the *blindsign* rule. Then it can only look as follows:

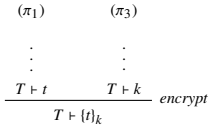
$$\dfrac{\dfrac{\begin{array}{cc}(\pi_1) & (\pi_2)\\ \vdots & \vdots \\ T \vdash [\{n\}_k, m] & T \vdash inv(k)\end{array}}{T \vdash [n, \{m\}_{inv(k)}]}\ blindsign \qquad T \vdash k}{T \vdash [\{n\}_k, m]}\ blindsign$$

But then, π_1 has the same conclusion as π, contradicting the normality of π.

In a normal proof, there cannot be an application of a *blindpair* rule followed by an application of a *blindsign* rule followed by an application of a *blindsplit* rule. The proof is as follows: Suppose a normal proof π ends with a *blindpair* rule followed by a *blindsign* rule followed by a *blindsplit* rule. Then it looks as follows:



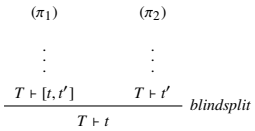
But then π is not normal, since the following is a shorter proof of $T \vdash \{t\}_k$.



We use these facts in the proof of the following proposition.

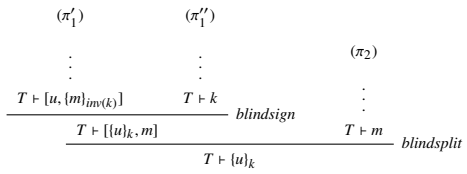
Proposition 3 *Let π be a normal proof of $T \vdash t$, and let r be a term occurring in π. Then $r \in st(T \cup \{t\})$, and if π ends in an application of an analz rule, $r \in st(T)$.*

Proof: We prove this by induction on the structure of proofs. We will use the fact that subproofs of normal proofs are also normal. So the induction hypothesis is always available to us. We present only the most important case:
Suppose π is of the following form and r is a term occurring in π:



We claim that $[t, t'] \in st(T)$, whence $st(T \cup \{[t, t']\}) = st(T \cup \{t'\}) = st(T)$. Now by the induction hypothesis, any r' occurring in π_1 belongs to $st(T \cup \{[t, t']\}) = st(T)$, and any . Since $[t, t']$ belongs to $st(T)$, t also belongs to $st(T)$. Now, any r occurring in π either occurs in π_1 or is the same as t. In either case, $r \in st(T)$.

If π_1 ends in an *analz*-rule, then by induction hypothesis, $[t, t'] \in st(T)$, and we are done. Otherwise, there are two cases to consider. The first case is that π_1 ends in a *blindpair* rule. This case cannot happen, since then π_1 would have a subproof with $T \vdash t$ as conclusion, contrary to the assumption that π is normal. The only remaining case is that π_1 ends in a *blindsign* rule. Then it has to be the case that $t = \{u\}_k$ and $t' = m$ for some $k \in \mathcal{K}$ and $m \in \mathcal{B}$, and π looks as follows:



Now since π is a normal proof, π'_1 will not end in a *blindpair* rule or a *blindsign* rule, as observed earlier. The only other possibility is that π'_1 ends in an *analz*-rule. Thus, by induction hypothesis $[u, \{m\}_{inv(k)}]$ belongs to $st(T)$, and hence so does $[\{u\}_k, m]$ and $\{u\}_k$, by our definition of $st(T)$. Thus $[t, t'] \in st(T)$ in all cases, and we are done. ⊣

Theorem 1 *Given a finite set of terms T and a term t, checking whether $t \in \overline{T}$ is decidable in time polynomial in size of T.*

Proof:

Suppose there is a proof of $T \vdash t$. Then there is a normal proof of $T \vdash t$. Also, all the terms occurring in this proof are subterms of $T \cup \{t\}$. Further, along every branch of a normal proof, the same term cannot occur twice. Thus the height of a normal proof of $T \vdash t$ is bounded by the size of $st(T \cup \{t\})$, and hence by $D = 7 \cdot |T \cup \{t\}|$. Therefore it suffices to check if there exists a proof of $T \vdash t$ of height D. This is easy to check. Start with $T' = T \cup \{t\}$ and repeat D times the following step: Replaces T' by $T'' \cap st(T \cup \{t\})$, where T'' is all the terms got by *one* application of a *synth* or *analz* rule to two terms in T'. Finally check if t belongs to T'. Since the proof system we have presented is monotone, this ensures that if a rule is applicable at some stage, it remains applicable even at a later stage. So the above procedure correctly yields all the terms of interest that are derivable from T by proofs of height at most D. Thus the problem of checking whether $t \in \overline{T}$ is decidable in polynomial time. ⊣