Using a Multimodal Logic to Express Conflicting Interests in Security Protocols

Antti Huima (antti.huima@hut.fi)
Tuomas Aura (tuomas.aura@hut.fi)

Helsinki University of Technology, Finland*

July 25, 1997

Abstract

A lot of cryptographic protocols are designed to work in situations where the participants of the protocols have different, or even conflicting, interests. These conflicts must be taken into account when the protocols are developed in order to avoid pitfalls. More generally, it must be ensured that the rights of the different parties can be protected. Usually, the rights and protecting them are included in the underlying goals of a protocol. However, no formal method of examining them do exist. In this paper we propose a formal method that can be used to analyse situations involving conflicting interests and rights to be protected. The method is based on a standard multimodal logic **S4n**. With the logic, we formalize and validate reasoning done in protocol design. As an example we analyse a secret election protocol appeared in the literature.

1 Introduction and Motivation

A lot of cryptographic protocols hare designed to work in situations where the participants of the protocols have different, or even conflicting, interests. Various specialized cryptographic protocols put extra emphasis on ensuring fairness to all involved parties in such situations. For example, designers of electronic payment schemes have to consider the interests of banks, customers, merchants, national banks and economies and law-enforcement agencies. Similarly, specialized digital signature systems must guarantee different properties to signers, verifiers and law-enforcement. Nevertheless, there is no general framework for representing and analysing the conflicting interests.

*We want to thank Dr. Ilkka Niemelä for the valuable help that he gave us through sharing his expertise on the area of modal logics; and his time. He is however not responsible for the contents of this paper in any way.

When systematic proof techniques and automatic verification tools for cryptographic protocols are reaching the level where the actually will be used by protocol developers, it is especially important that the verified properties correspond to the application requirements. That is why we believe that more insight into the derivation of the design goals and the relationships between the formal requirements is needed.

In this paper we present a formal method that can be used to express conflicting interests and to analyse situations that involve them. With the logic, we formalize and validate reasoning done in protocol design. The method that we use is based on a standard multi-modal logic known **S4n**, applied to the current problem domain. We use both propositional and first-order versions of the logic.

This paper is structured as follows.

- We shortly introduce modal logics in Section 2.
- In Section 3, we describe the problem of conflicting interests more carefully.
- We describe our proposed method for arguing about conflicting interests in Section 4.
- Two examples are given in Section 5.
- Finally, conclusions are drawn and directions for future work are discussed in Section 6.

2 What Are Modal Logics?

Modal logics are logics that examine qualified truth. For example, ideas such as "necessarily true", "possibly true" and "known to be true" appear in the applications of modal logics. [Fit93] is a well-written introduction to basic modal logics. More classic works include [Che80, HC84]. A more thorough treatise of the theoritical aspects is found for instance in [HM92].

A propositional modal logic with only one modality is obtained from standard propositional calculus by adding to the language a unary operator $\Box$ (pronounced "box"); it is *the modality operator*. Different axioms can be given. Often these axioms reflect the interpretation of the modality. For example, if $\Box\phi$ is read as "necessarily ϕ" then the following axiom:

$$\Box\phi \supset \phi$$

makes sense. It says that if something is necessarily true, then it is true. On the other hand, interpreting $\Box\phi$ as "it is believed that ϕ", the axiom above is questionable. For it is possible to believe something that is not true.

Different sets of axioms create different modal logics. Some of the logics are given standard names. For example, the logic that obeys the three modal axioms

$$\Box(P \supset Q) \supset (\Box P \supset \Box Q) \tag{1}$$

$$\Box P \supset \Box\Box P \tag{2}$$

$$\Box P \supset P \tag{3}$$

is given the standard name **S4**.

Both Hilbert-style and tableaux-style proof methods exist for modal logics.

Multimodal logics are modal logics that have more than one modality. Relevant to us is the multimodal logic **S4n** where every modality independently obeys (1)–(3).

3 The Problem

In this section we describe the problem of conflicting interests.

Cryptographic protocols are often used in situations where the protocol is meant to ensure protection of the rights of the participating entities. For instance,

a) In digital cash schemes the authorities must be able to detect and demonstrate fraud when it happens.

b) Also, the cash users should be able to detect bank stealing their legally owned cash and demonstrate that stealing has happened.

c) In voting protocols, the voters must be able to detect and demonstrate if some parties try to swindle. For example, if the voters' legal votes are dropped out or changed.

d) If commercial information services are provided over the network, the providers need to be able to demonstrate plausibly that certain parties have used the services, in order to resolve conflicts that might arise.

These issues have been addressed, at least partially, in existing protocols—but on a per case basis. Although formal proofs have been presented about the workings of protocols, the discussion of high-level properties pertaining to the protection of rights have been informal and not well studied.

This is a problem, because the ultimate goals of the protocol are unclearly stated. Technical results of what a protocol achieves can be given and proven using existing methods. However, there is no apparatus for interpreting these results in a standard way. We want to present one here.

4 Presenting Our Method

We begin to develop a formal apparatus to tackle the problem described in the previous section. The central notion will be that of *demonstrability to a third party.*[1] We are interested in claims and clauses such that “If X is true then Alice can demonstrate that [to a third party]”, or “Alice can demonstrate [to a third party] that X implies Y”.

This is because by being able to demonstrate the real situation to anyone else she can claim justice for her.

We use $\mathsf{P}_A(\phi)$ to denote that “A can demonstrate ϕ [to a third party]” (from now on we shall drop “[to a third party]” out). In fact, we shall make P_p a modality for every entity p in the system.

4.1 Defining the Logic

Definition 1 *Let $\mathcal{P}$ be a set of participants. We are interested in the following multimodal logic:*

- *There is one modality, P_p, for every $p \in \mathcal{P}$.*
- *The logic is defined by the following axioms:*
  1. *all propositional tautologies;*
  2. *all formulas of the form $\mathsf{P}_p(X \supset Y) \supset (\mathsf{P}_p(X) \supset \mathsf{P}_p(Y))$;*
  3. *of the form $\mathsf{P}_p(X) \supset X$; and*
  4. *of the form $\mathsf{P}_p(X) \supset \mathsf{P}_p(\mathsf{P}_p(X))$ for all $p \in \mathcal{P}$.*

This definition, though lacking rigor, defines a standard multimodal logic called **S4n**. Because the binding we make to the set of participants $\mathcal{P}$, we denote any specific instanse of the logic by **S4$\mathcal{P}$**.

We assume that the set $\mathcal{P}$ is finite. For this reason we may use the notations $(\forall p \in \mathcal{P})[X]$ and $(\exists p \in \mathcal{P})[X]$ in the apparent sense (i.e. quantify over participants or modalities). We will also use n-ary predicates, computable functions and equality operator where necessary.

We need to make proofs in our logic. The example proofs in this paper will be Hilbert-style. We will use short hand markings in our proofs and use well-established deduction rules in addition to the two basic deduction rules, *Modus Ponens*[2] and *Necessitation*[3].[4] **S4n** is shown to be decidable in [HM92]. Tableaux-based proof methods are quite complicated to systematize, but exist however.

[1]We append the postfix “to a third party” in order to clearly distinguish the concept from that of “real” logics of provability, such as **GL**. See e.g. [Boo93].

[2]From ϕ and $\phi \supset \gamma$ infer γ.

[3]From ϕ infer $\mathsf{P}_p(\phi)$.

[4]Moreover, we will not refer to a semantic interpretation of the logic at all. This is a deliberate choice whose purpose is to avoid the extra complication that presentation of semantics would add to this paper.

4.2 Why These Axioms?

We must, of course, give some justification to the axioms we have just presented, remembering the interpretation that we just gave for the modalities—demonstrability. We will first consider the three modal axiom schemas and then discuss a problem that inavoidably appears.

$\mathsf{P}_p(X \supset Y) \supset (\mathsf{P}_p(X) \supset \mathsf{P}_p(Y))$. This is the basic axiom of all frame based modal logics, to which **S4** belongs. When we interpret the modality in this axiom in the way suggested above, we read: "If p can demonstrate that Y is necessarily true when X is, then if p can additionaly demonstrate X, he can demonstrate Y also." If I can *prove* that Y follows from X and that X is true, then I have just shown that Y must be true also, despite of the interpretation of the subclauses X and Y. For this reason we see that this axiom captures an essential aspect of demonstrability.

$\mathsf{P}_p(X) \supset X$. We claim that X cannot be false if someone is able to prove X, which is what this axiom captures. This implication is inherent in our choice of the interpretation of the modality. Namely, you cannot *prove* a thing that does not hold! For this reason we see that this notion is also a part of the concept of demonstrability.

If the assumptions we make about a system are not consistent this axiom has especially great potential to be the cause of contradictory derivations. We assume that our notion of provability has the property captured by this axiom. Thus we must ensure ourselves that the assumptions we make do not enable anyone to prove anything that should be false. In our examples this consisteness requirement is quite easily seen to hold.

$\mathsf{P}_p(X) \supset \mathsf{P}_p(\mathsf{P}_p(X))$. The last axiom claims, that everyone who is able to prove X, despite of the interpretation of X, is also able prove that he/she is able to prove that. This makes sense because *by giving the proof* of X, the ability to prove X is undeniably demonstrated.

The problem of logical omniscience. This wording is used to refer to a difficulty that appears inevitably in logics like the one we use due to the Necessitation rule, though originally it refers to logics of knowledge [FHMV95]. Namely, if X is always true (it is, for example, a tautology), then we get using Necessitation $(\forall p \in \mathcal{P})[\mathsf{P}_p(X)]$. In essence, all participants must be able to prove all universally true clauses to a third party, despite of the possible complexity of the proofs. This however, turns out not to be a problem in our practical applications.

4.3 General Analysis Process

In modal logics there are two kinds of true formulas: those that are *locally* and those that are *globally* true. In our application the locally true formulas will represent facts that hold but that cannot be necessarily proven by anyone. The globally true formulas are facts that can be necessarily proven by everyone.

Our proposed analysis process goes as follows:

1. Examine the situation/protocol at hand.

2. Choose appropriate predicates and propositions to model the situation.

3. Model the logical interrelations between the predicates using clauses of $\mathbf{S4}\mathcal{P}$. These clauses constitute the set $\mathcal{W}$ which we assume to be locally but not globally true. In other words, the clauses represent facts but they cannot be necessarily proven.

4. Find the situations where some participants' rights must be protected. Model them using clauses of $\mathbf{S4}\mathcal{P}$, attaching the clauses to the pertaining participants.

5. Map these clauses representing "critical situations" into another set of clauses, *requirements* $\mathcal{R}$. The idea is that if the requirement clauses are true, then the participants can protect their rights as they were modelled in the previous step.

6. Try to prove that if $\mathcal{W}$ is locally true, then is $\mathcal{R}$ also. A standard notation for this is

$$\emptyset \vdash_{\mathbf{S4}\mathcal{P}} \mathcal{W} \supset \mathcal{R}$$

   We will drop $\emptyset$ out, however, as it denotes the set of global premises which we will have always empty.

4.4 Critical Situations

Having referred to *critical situations* above we now define them.

Definition 2 (Critical situation) *A certain situation is critical for A if, when in the situation, A needs to be able to know that the situation holds and to prove that. In particular, critical situations include situations where*

- *A can demand benefits from other parties, based on a law or contract; and those where*

- *A's rights are not being honored.*

For example, assume that you have bought a new car and find after two weeks that there is no gear box in the vehicle. This is a critical situation for you, as the seller has cheated. It does not help you much to only *know* that the gear box is missing, you must also be able to *show* that it is missing and that you paid for it in order to get justice.

On the other hand, assume that you paid only for the car, but suddenly you find that there are is also a brand new stereo rack mounted in the car which you didn't paid for. This is not a critical situation for you *per se* because your rights are not being deprived by you receiving a free rack. You do not need to be able to prove that you have a free rack.[5]

The main point of critical situations is that when in one, an entity must be able to prove that the situation really holds. This the subject of the next subsections.

[5] Of course, this thinking has nothing to do with ethical issues.

4.5 Critical Pairs, Critical Sets and Mapping Them to Requirements

Definition 3 (Critical Pair) *Let ϕ denote a situation that is critical for p. Then the pair $\langle \phi, p \rangle$ is a critical pair.*

Definition 4 (Mapping) *Let Φ denote all clauses of an* $\mathbf{S4}\mathcal{P}$ *logic. We define the mapping* $\mathbf{G} : \Phi \times \mathcal{P} \mapsto \Phi$ *as*

$$\mathbf{G}(\langle \phi, p \rangle) =_{def} \phi \supset \mathsf{P}_p(\phi)$$

For $\mathcal{S} \subseteq \Phi$ we use $\mathbf{G}(\mathcal{S})$ to denote

$$\{\mathbf{G}(\phi) \mid \phi \in \mathcal{S}\}$$

$\mathbf{G}$ maps critical pairs into clauses that, when true, guarantee that when ever in a critical situation, the participants concerned can demonstrate it. This is exactly what we discussed in the previous section.

We can now define our method more formally.

4.6 Making a Long Story Short

Let $\mathcal{C}$, a set of critical pairs, and $\mathcal{W}$, containing rules describing the system at hand, be given.

Definition 5 *(Solution) Let $\mathcal{S} \subseteq \Phi$. $\mathcal{S}$ is a solution of $\langle \mathcal{C}, \mathcal{W} \rangle$ iff*

$$\vdash_{\mathbf{S4}\mathcal{P}} (\mathcal{W} \cup \mathcal{S}) \supset \mathbf{G}(\mathcal{C})$$

Thus, if a set of clauses $\mathcal{S}$ is a solution for $\langle \mathcal{C}, \mathcal{W} \rangle$, then the following holds: if $\mathcal{S}$ is true in addition to $\mathcal{W}$, then all participants' rights can be protected in the sense defined previously (they can demonstrate, giving a proof, that their rights are being dishonored if that is the case). We want to separate $\mathcal{W}$ from $\mathcal{S}$ for clarity; $\mathcal{S}$ can contain different assumptions about the behaviour of the participants etc. that cannot be taken directly to be properties of a protocol being examined. However, no theoretical difference between the two types of premises exists.

4.7 Assuming Certain Ideality

It appears that such deductions as "if A has payed the bill, she must have a receipt" occur often in applications. However, this kind of thinking suffers of some inconsistency when stated formally. We would like to translate the clause above to the form

$$\mathit{billPayed} \supset \mathit{haveReceipt}$$

This has the contraposition

$$\neg haveReceipt \supset \neg billPayed$$

But this appears to lead to inconsistent results, if A has paid the bill but for some reason has lost the bill. Namely, she *has* indeed paid the bill, although the logic claims otherwise. For this reason we need sometimes to assume that such situations do not take place. Another solution would be to model these exceptions also, but it would lead to unnecessary complications. The same assumptions must be made in any traditional paper-based system: lost certificates may lead to contradictory results.

4.8 Proving disprovability

Another commonly occurring hard situation is that where A must be able to prove that B cannot prove ϕ although A cannot give a direct proof for $\neg\phi$. This happens when ϕ is beneficial for B and unbeneficial for A, thus $\neg\phi$ is beneficial for A. A wants to show that B can prove ϕ whenever it is true. Then proving that B cannot do that, A has proven by contraposition that ϕ is false. In the terms of logic:

$$\begin{aligned}
&\mathsf{P}_A(\phi \supset \mathsf{P}_B(\phi)) && && (4)\\
&\neg\mathsf{P}_B(\phi) \supset \mathsf{P}_A(\neg\mathsf{P}_B(\phi)) && && (5)\\
&\neg\phi && && (6)\\
&\phi \supset \mathsf{P}_B(\phi) && \text{from (4) by axiom 2} && (7)\\
&\neg\mathsf{P}_B(\phi) \supset \neg\phi && \text{from (7) by contraposition} && (8)\\
&\mathsf{P}_B(\phi) \supset \phi && \text{straightly from axiom 2} && (9)\\
&\neg\phi \supset \neg\mathsf{P}_B(\phi) && \text{from (9) by contraposition} && (10)\\
&\neg\mathsf{P}_B(\phi) && \text{from (6) and (10) by MP} && (11)\\
&\mathsf{P}_A(\neg\mathsf{P}_B(\phi)) && \text{from (5) and (11) by MP} && (12)\\
&\mathsf{P}_A(\neg\mathsf{P}_B(\phi) \supset \neg\phi) && \text{from (4) by c.p. inside } \mathsf{P}_B && (13)\\
&\mathsf{P}_A(\neg\phi) && \text{from (12) and (13) by axiom 1} && (14)
\end{aligned}$$

The crucial assumption is (5), and it must be discussed. Namely, it formalizes the argument that if B is able to prove ϕ and it is beneficial for him to do so, then he will exhibit a proof. Or, when B is asked to give a proof but he does not, it is concluded that B cannot give a proof. This is quite commonsense in everyday life. For example, a contract may contain wording such as: "A receipt must be presented in two weeks of the request. In the case of failure to do that, the receipt is assumed not to exist." For this reason we will use this rule when necessary, though giving a comment when it is used.

5 Examples

In this section we give two examples. First we give a very easy example to demonstrate the method. The second example is about analysing a voting protocol.

5.1 A Small Example

In this example we consider a situation that takes place between a phone user U and a phone company C. The user subscribes to telecommunication services provided by the company. We have three predicates:

- *called*: the user has made a call
- *charge*: the company charges the user for an unpaid call
- *paid*: the user has paid his call

Note that we are just talking about "a call" and not making any distinction between different calls. That is, we assume that every call is independently charged.

Naturally $\mathcal{P} = \{C, U\}$.

We assume that there exists the following relationship between the predicates: the user does not pay a call before he has dialled it. We assume also that both the user and the company are able to prove when the user is being charged. This is reasonable, because otherwise the bill has not much legal force. (Other relationships will later be added in the solution set $\mathcal{S}$.) We thus get

$$\mathcal{W} = \left\{ \begin{array}{l} paid \supset called, \\ charge \supset (\forall p \in \mathcal{P})[\mathsf{P}_p(charge)] \end{array} \right\}$$

We proceed to define the critical pairs. From the user's point of view, the company should charge him only if he has made a call that is not yet paid. From the company's point of view, it should be able to charge the user always when this is the situation. From these two observations we get the set of critical pairs

$$\mathcal{C} = \left\{ \begin{array}{l} \langle \neg(charge \supset (called \wedge \neg paid)), U \rangle \\ \langle \neg((called \wedge \neg paid) \supset charge), C \rangle \end{array} \right\}$$

From these we get the set of requirements $\mathcal{R} = \mathbf{G}(\mathcal{C})$:

$$\mathcal{R} = \left\{ \begin{array}{l} \neg(charge \supset (called \wedge \neg paid)) \supset \mathsf{P}_U(\neg(charge \supset (called \wedge \neg paid))), \\ \neg((called \wedge \neg paid) \supset charge) \supset \mathsf{P}_C(\neg((called \wedge \neg paid) \supset charge)) \end{array} \right\}$$

We should then try to prove $\vdash_{\mathbf{S4}\mathcal{P}} \mathcal{W} \supset \mathcal{R}$. This turns out to be impossible, however. The reason is that the parties cannot prove anything about the predicates *called* and *paid*.[6]

We will now show that

$$\mathcal{S} = \left\{ \begin{array}{l} \mathsf{P}_U(called \supset \mathsf{P}_C(called)), \\ \mathsf{P}_C(paid \supset \mathsf{P}_U(paid)), \\ \neg\mathsf{P}_C(called) \supset \mathsf{P}_U(\neg\mathsf{P}_C(called)), \\ \neg\mathsf{P}_U(paid) \supset \mathsf{P}_C(\neg\mathsf{P}_U(paid)) \end{array} \right\}$$

[6] Were we to refer to the semantics we could easily present a counter model here. We do, however, skip this issue as mentioned previously.

is a solution of $\langle \mathcal{C}, \mathcal{W} \rangle$, i.e. that $\vdash_{\mathbf{S4}\mathcal{P}} (\mathcal{W} \cup \mathcal{S}) \supset \mathcal{R}$. Indeed,

$$\begin{array}{rll}
\mathit{paid} \supset \mathit{called} & & (15)\\
\mathit{charge} \supset (\forall p \in \mathcal{P})[\mathsf{P}_p(\mathit{charge})] & & (16)\\
\mathsf{P}_U(\mathit{called} \supset \mathsf{P}_C(\mathit{called})) & & (17)\\
\mathsf{P}_C(\mathit{paid} \supset \mathsf{P}_U(\mathit{paid})) & & (18)\\
\neg\mathsf{P}_C(\mathit{called}) \supset \mathsf{P}_U(\neg\mathsf{P}_C(\mathit{called})) & & (19)\\
\neg\mathsf{P}_U(\mathit{paid}) \supset \mathsf{P}_C(\neg\mathsf{P}_U(\mathit{paid})) & & (20)\\
\mathit{paid} \supset \mathsf{P}_U(\mathit{paid}) & \text{from (18)} & (21)\\
\mathsf{P}_C(\mathit{called}) \supset \mathit{called} & \text{axiom} & (22)\\
\neg\mathit{called} \supset \neg\mathsf{P}_C(\mathit{called}) & \text{from (22)} & (23)\\
\mathsf{P}_U(\neg\mathsf{P}_C(\mathit{called}) \supset \neg\mathit{called}) & \text{from (17)} & (24)\\
\neg\mathit{called} \supset \mathsf{P}_U(\neg\mathsf{P}_C(\mathit{called})) & \text{from (19) and (23)} & (25)\\
\neg\mathit{called} \supset \mathsf{P}_U(\neg\mathit{called}) & \text{from (24) and (25)} & (26)\\
(\neg\mathit{called} \vee \mathit{paid}) \supset (\mathsf{P}_U(\neg\mathit{called}) \vee \mathsf{P}_U(\mathit{paid})) & \text{from (21) and (26)} & (27)\\
(\neg\mathit{called} \vee \mathit{paid}) \supset \mathsf{P}_U(\neg\mathit{called} \vee \mathit{paid}) & \text{from (27)} & (28)\\
\begin{array}{r}(\mathit{charge} \wedge (\neg\mathit{called} \vee \mathit{paid})) \supset\\ \mathsf{P}_U(\mathit{charge}) \wedge \mathsf{P}_U(\neg\mathit{called} \vee \mathit{paid})\end{array} & \text{from (16) and (28)} & (29)\\
\begin{array}{r}(\mathit{charge} \wedge (\neg\mathit{called} \vee \mathit{paid})) \supset\\ \mathsf{P}_U(\mathit{charge} \wedge (\neg\mathit{called} \vee \mathit{paid}))\end{array} & \text{from (29)} & (30)
\end{array}$$

(30) is equal with the first requirement. The second requirement is proved using a similar argument. Note that actually (15) (item of $\mathcal{W}$) was not used at all.

The solution set $\mathcal{S}$ can be interpreted as resulting from a contract of the form: "the company must be able to demonstrate dialled call when requested. Otherwise all requests of charge will be declared void", and similarly for the customer's paid bills. This is exactly the situation discussed in Sec. 4.8.

5.2 Analysing the Two-Agency Secret Election Protocol

We analyse a modified version [Sal96, Sch96] of the Two-Agency Secret Election Protocol, proposed originally by Salomaa et al. [NSS91, Sal91].

5.2.1 Presenting the Protocol

The protocol goes as follows. There are two agencies: one for legitimizating the voters (L) and another for computing the votes (C). We have also a voter (A).

Step 1: A asks L for an identification number.

Step 2: L checks if A is allowed to vote. If this is the case and A has not received it before, L sends A a random identification number i_A.

Step 3: After all voters have requested their identification numbers, L sends the set of the granted numbers, N, to C.

Step 4: A chooses a secret identification number s_A. Then he sends C the triple $\langle i_A, v_A, s_A \rangle$ where v_A is her vote.

Step 5: C finds out whether $i_A \in N$ or not. If it is in the set, C removes i_A from N and adds s_A to the list of voters that voted for v_A.

Step 6: When the voting period is over, C publishes a list of pairs $\langle s_p, v_p \rangle$, linking every s_p to the corresponding vote.

In the following we assume that L works honestly and correctly. Of course, if L were cheating voters could, for instance, get many identification numbers. This would break the whole system immediately. (L can be replaced by running an all-or-nothing-disclosure-of-secrets protocol with C [NSS91].)

Our analysis will concentrate on the rights of A, an individual voter.

We also make the following minor modification to the protocol. If somebody tries to vote multiple times with the same identification number the first vote sent is the one registered by C in Step 5. We change this to let C choose the vote to use from the many votes sent by the party instead.

5.2.2 First-order Modal Logic

At this point we will begin to use first-order modal logic, i.e. add predicates and quantifiers to the previously propositionally based logic we have used. (We use rigid designators [Fit93].) Incorporating predicate calculus into modal logics contains some subtleties due to the amount of choices available regarding to the interpretation of the semantics of the quantifiers.

5.2.3 Model

We will use the following sets: $\mathcal{I}$ is the set of possible identification numbers, $\mathcal{S}$ is the set of possible secret identifications, $\mathcal{V}$ is the set of possible voting strategies and $\mathcal{O}$ is the set of voters.

The predicates that we have chosen to model the protocol are presented in Figure 5.2.3. In order to get $\mathcal{W}$ we do the following observations:

- As L is assumed to be honest, two different voters do not get the same identification numbers.
- Moreover, a voter does not get two distinct identification numbers from L.
- All numbers granted to voters are sent to C in step 3. The list contains no extra numbers.

$sent(o, \langle i, v, s\rangle)$	the voter o has sent the triple $\langle i, v, s\rangle \in \mathcal{I} \times \mathcal{V} \times \mathcal{S}$ to C
$published(p)$	C has included the pair $p \in S \times V$ in the list of step 6
$granted(o, i)$	L has granted the identification number i to the voter o
$given(i)$	L has given the identification number i to C in step 3

Figure 1: Predicates used in analysing the Two-Agency Protocol

- The list published in step 6 is public.

When we state these observations formally we get the following clauses $\mathcal{W}$:

$$(\forall o, o' \in \mathcal{O}; i \in I)[(granted(o, i) \wedge granted(o', i)) \supset o = o'] \tag{31}$$
$$(\forall o \in \mathcal{O}; i, i' \in I)[(granted(o, i) \wedge granted(o, i')) \supset i = i'] \tag{32}$$
$$(\forall o \in \mathcal{O}; i \in \mathcal{I})[granted(o, i) \supset given(i)] \tag{33}$$
$$(\forall i \in \mathcal{I})[given(i) \supset (\exists o \in \mathcal{O})[granted(o, i)]] \tag{34}$$
$$(\forall p \in \mathcal{S} \times \mathcal{V})[(\forall a \in \mathcal{P})[publised(p) \supset \mathsf{P}_a(published(p))]] \tag{35}$$
$$(\forall p \in \mathcal{S} \times \mathcal{V})[(\forall a \in \mathcal{P})[\neg publised(p) \supset \mathsf{P}_a(\neg published(p))]] \tag{36}$$

We let $\mathcal{P} = \{A, C, L\}$.
We use also the derived predicates:

$$sentLegal(o, i) \equiv (\exists v \in \mathcal{V}; s \in \mathcal{S})[sent(o, \langle i, v, s\rangle) \wedge granted(o, i)] \tag{37}$$
$$noPair(i) \equiv \neg(\exists o \in \mathcal{O}; s \in \mathcal{S}; v \in \mathcal{V})[sent(o, \langle i, v, s\rangle) \wedge published(\langle s, v\rangle)] \tag{38}$$
$$notCounted(o) \equiv (\exists i \in \mathcal{I})[sentLegal(o, i) \wedge noPair(i)] \tag{39}$$
$$duplicate(\langle s, v\rangle, \langle s', v'\rangle) \equiv (\exists o, o' \in \mathcal{O}; i \in \mathcal{I})[sent(o, \langle i, v, s\rangle) \wedge sent(o', \langle i, v', s'\rangle)] \tag{40}$$
$$dupPub(p, p') \equiv p \neq p' \wedge published(p) \wedge published(p') \wedge duplicate(p, p') \tag{41}$$
$$noSender(\langle s, v\rangle) \equiv \neg(\exists o \in \mathcal{O}; i \in \mathcal{I})[sent(o, \langle i, v, s\rangle) \wedge granted(o, i)] \tag{42}$$
$$fraudulentPair(p) \equiv (published(p) \wedge noSender(p)) \vee \tag{43}$$
$$(\exists p' \in \mathcal{S} \times \mathcal{V})[dupPub(p, p')] \tag{44}$$

Here *sentLegal* is true for a voter and an identification number if the voter has sent at least one triple using the identification number and the identification number is valid, i.e. granted by L. *noPair* is true for an identification number if there exists no $\langle s, v\rangle$-pair that the voter had sent with the identification number and C had published in the protocol step 6. *notCounted* is true for a voter if there exists an identification number that the voter has used when sending at least one vote to C but none of the votes sent with the identification number do appear in the list.

duplicate is true for two $\langle s, v\rangle$-pairs when they have been sent using the same identification number. *dupPub* denotes the situation that two pairs are published that are *duplicate*. *noSender* is true for a similar pair if nobody has sent it with a valid identification number.

Finally, *fraudulentPair* is true for a pair if it has been published but either does not have a legal sender or another published pair exists that has been sent with the same identification number.

Critical pairs for A are then

$$\langle notCounted(A), A\rangle \tag{45}$$
$$\langle(\exists p \in \mathcal{S} \times \mathcal{V})[fraudulentPair(p)], A\rangle \tag{46}$$

In essence, the situations where A's rights are in danger are those where her vote is ignored or modified (45) and those where illegal votes are counted (46).

We get then the requirements

$$\mathcal{R} = \left\{ \begin{array}{l} notCounted(A) \supset \mathsf{P}_A(notCounted(A)) \\ (\exists p \in \mathcal{S} \times \mathcal{V})[fraudulentPair(p)] \supset \mathsf{P}_A((\exists p \in \mathcal{S} \times \mathcal{V})[fraudulentPair(p)]) \end{array} \right\} \tag{47}$$

5.2.4 Checking the Requirements

Trying to prove $\vdash_{\mathbf{S4}\mathcal{P}} \mathcal{W} \supset \mathcal{R}$, we get stuck. Starting from (45) and looking at the definition of *notCounted*, the first problem seems to be that A cannot prove that she has sent anything. Indeed, this is a problem in the protocol. Because A cannot prove that she sent a vote to C, C can claim that he never received it.

Another problem is that A cannot prove $granted(A, i)$ for any i.

Problems are encountered when examining (46) also. A has no way to prove *noSender*-clauses, for example.

We thus see that the protocol as formalized above does not fulfil the requirements we stated as the critical pairs (45)–(46). This reflects real world problems and/or bad formalization.

We will now begin to revise the formalization and the assumptions concerning the system until we find a result that fulfils (45) and (46). We do this trying to get a better voting protocol.

We start from (45) and work top-down. We make first the assumptions (their interpretations will be discussed below)

$$(\forall i \in \mathcal{I})[sentLegal(A, i) \supset \mathsf{P}_A(sentLegal(A, i))] \tag{48}$$
$$(\forall i \in \mathcal{I})[((granted(A, i) \wedge noPair(i)) \supset \mathsf{P}_A(noPair(i))] \tag{49}$$

(48) and (49) together ensure that (45) is handled:

$$\begin{array}{rll}
notCounted(A) & \text{assumption} & (50) \\
(\exists i \in \mathcal{I})[sentLegal(A, i) \wedge noPair(i)] & \text{from (39), (50)} & (51) \\
(\exists i \in \mathcal{I})[sentLegal(A, i) \wedge granted(A, i) \wedge noPair(i)] & \text{from (51), (37)} & (52) \\
(\exists i \in \mathcal{I})[\mathsf{P}_A(sentLegal(A, i)) \wedge \mathsf{P}_A(noPair(i))] & \text{from (52), (49), (48)} & (53) \\
(\exists i \in \mathcal{I})[\mathsf{P}_A(sentLegal(A, i) \wedge noPair(i))] & \text{from (53)} & (54) \\
\mathsf{P}_A((\exists i \in \mathcal{I})[sentLegal(A, i) \wedge noPair(i)]) & \text{from (54)} & (55) \\
\mathsf{P}_A(notCounted(A)) & \text{from (55)} & (56)
\end{array}$$

Similarly we take care of (46) by assuming

$$(\forall p \in \mathcal{S} \times \mathcal{V})[noSender(p) \supset \mathsf{P}_A(noSender(p))] \tag{57}$$

$$(\forall p, p' \in \mathcal{S} \times \mathcal{V})[dupPub(p, p') \supset \mathsf{P}_A(dupPub(p, p'))] \tag{58}$$

Thus (48), (49), (57) and (58) together form a solution set of the system. What has been assumed? (48) states that A can demonstrate when appropriate that she has sent a triple with a legal identification number. (49) formalizes that A can show that none of the $\langle s, v\rangle$ pairs sent by her in triples with an identification number are published in the public list. (57) claims that A can prove that a pair has no legal sender when that is the case, and (58) that she can always demonstrate, when appropriate, that the same identification number was used to make two counted votes.

We now see how the protocol can be modified to satisfy the first two of the four necessary assumptions, i.e. (48) and (49).

The straightforward (48) has actually a problem. Namely, if A does prove that she has sent a triple with i, then she reveals the connection between i and herself and the anonymousness of her vote is lost. We assume that a trusted third party can be used to mediate without disclosing A's identity. This is just a design choice we make; we could try to remove his problem by making another assumptions instead of (48). Similar problems appear with (49).

We must now examine how to implement (48) and (49).

(48) is clearly a consequence of

$$(\forall i \in I)[granted(A, i) \supset \mathsf{P}_A(granted(A, i))] \tag{59}$$

$$(\forall \langle i, s, v\rangle \in I \times S \times V)[sent(A, \langle i, s, v\rangle) \supset \mathsf{P}_A(sent(A, \langle i, s, v\rangle))] \tag{60}$$

In order to be able to implement (59) we assume that the identification numbers granted to voters have certificates that include the intended user's name. (60) can be implemented by requiring C to send a signed certificate (receipt) of all the received messages to the senders.

Thus making these two changes to the protocol we can ensure (48). (49) is a bit more complicated as a non-existence must be proved. We accomplish this using an indirect proof, assuming:

$$\mathsf{P}_A((\forall o \in \mathcal{O}; i \in \mathcal{I}; v \in \mathcal{V}; s \in \mathcal{S})[(sent(o, \langle i, v, s\rangle) \wedge given(i)) \supset \mathsf{P}_C((\exists o' \in \mathcal{O})[sent(o', t)])]) \tag{61}$$

$$(\forall i \in \mathcal{I})[\neg \mathsf{P}_C(\neg noPair(i)) \supset \mathsf{P}_A(\neg \mathsf{P}_C(\neg noPair(i)))] \tag{62}$$

Here (61) says that whenever something has been sent using a legal identification number, C is able to prove that; and A is able to prove that this holds. This can be implemented so that i's itself are public keys and every voter is sent a private key corresponding to the public one. The private keys are used to sign the triples. C receives the public keys in the list N, and L certifies this list. (62) is an instance of the situation presented in Sec. 4.8. A is able to demonstrate that C cannot prove $\neg noPair(i)$ when this is the case.

We will give a sketch of a proof that with these assumpions, (49) follows. It is straightforward to prove that whenever $\neg noPair(i) \wedge given(i)$, $\mathsf{P}_C(\neg noPair(i))$. This is based on (61). As this is commonly provable by all participants, we can let in particular A be able to prove that. From that the conclusion follows via a contraposition argument using (62).

5.3 Conclusive Remarks

We sum up what we have done at this point:

1. We found that the protocol did not fulfil our requirements. This is a signal of either real problems or formalization bugs. In this case they represent real problems at least to some extent.

2. We were able to modify the protocol and the assumptions regarding to it in a way that some of the requirement are satisfied, i.e. the voter can detect and demonstrate that her vote has been modified or neglected. What we done was the following: (1) the identification numbers are associated with their meant receivers by certificates; (2) the voters get signed receipts when they send their triples; (3) C is able to prove that it has received a certain triple. This is done by making the identifier numbers actually pairs of public and private keys and using them to certify the votes anonymously; and (4) when C fails to give a proof of a triple received, it can be assumed guilty of a fraud.

We find that the method we used was easy and intuitive to use and it helped us to find problems and examine solutions to them.

6 Conclusions and Future Work

In this paper we presented a new method for analysing security protocols from the perspective of protecting the rights of the participants. The method was based on a standard multimodal logic commonly known as **S4n**. We defined our method and discussed some details of it. We demonstrated the method by analysing a voting protocol appeared in the literature.

The main contribution of our work to the area of trustworthy security engineering is giving a language to formally argue about conflicting interests and protection of rights; and proposing an applicable method that forces one to examine the underlying assumptions of a protocol to a great detail.

It may be possible to use the logic here in conjunction with existing authentication logics such as BAN [BAN90]. This is an area that should be examined. More case studies are needed in order to weight the usefulness of the method. Making proofs by hand is currently very cumbersome. Lemmas and more intuitive notation for certain commonly occurring clauses should be developed. The details of using a first-order multimodal logic still needs more attention. The case studies show this to be a promising direction. However, research on this subject is still in a very preliminary phase. Finally, other methods for examining conflicting interests could be developed.

References

[BAN90] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February 1990.

[Boo93] George Boolos. *The Logic of Provability.* Cambridge University Press, 1993.

[Che80] B.F. Chellas. *Modal Logic: an Introduction.* Cambridge University Press, Cambridge, 1980.

[FHMV95] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*, chapter Logical Omniscience. The MIT Press, 1995.

[Fit93] Melvin Fitting. Basic modal logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 365–448. Oxford Science Publications, 1993.

[HC84] G.E. Hughes and M.J. Cresswell. *A Companion to Modal Logic.* Methuen and Co., London, 1984.

[HM92] J.Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[NSS91] Hannu Nurmi, Arto Salomaa, and Lila Santean. Secret ballot elections in computer networks. *Computers & Security*, 10(6):553–560, October 1991.

[Sal91] Arto Salomaa. Verifying and recasting secret ballots in computer networks. In H. Maurer, editor, *New Results and New Trends in Computer Science*, pages 283–289. Springer-Verlag, 1991. Lecture Notes in Computer Science No. 555.

[Sal96] Arto Salomaa. *Public-Key Cryptography.* Springer-Verlag, 2nd edition, 1996.

[Sch96] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, New York, 2nd edition, 1996.