Knowledge transfer and information leakage in protocols

Abdullah Abdul Khadir, Madhavan Mukund, and S.P. Suresh

Chennai Mathematical Institute and UMI RELAX, Chennai, India.
{abdullah, madhavan, spsuresh}@cmi.ac.in

Abstract. A protocol defines a structured conversation aimed at exchanging information between two or more parties. Complete confidentiality is virtually impossible so long as useful information needs to be transmitted. A more useful approach is to quantify the amount of information that is leaked. Traditionally, information flow in protocols has been analyzed using notions of entropy. We move to a discrete approach where information is measured in terms of propositional facts. We consider protocols involving agents holding numbered cards who exchange information to discover each others' private hands. We define a transition system that searches the space of all possible announcement sequences made by such a set of agents and tries to identify a subset of announcements that constitutes an informative yet safe protocol.

1 Introduction

A protocol defines a structured conversation aimed at exchanging information between two or more parties. In a computational setting, there is a natural tension between transmitting relevant information to a trusted partner and leaking confidential data to an intruder.

This has led to the study of security in protocols from the perspective of information flow. Complete confidentiality is virtually impossible so long as useful information needs to be transmitted. For instance, rejecting an invalid password reveals indirectly what the password is *not*. Hence, a more useful approach is to quantify the amount of information that is made available to an eavesdropper and use this as a basis for evaluating the security of protocols.

Several proposals have been made over the past decade to model quantitative information flow [1–4, 6–8]. The general consensus has been to use ideas from information theory, primarily the notion of entropy, as a basis for measuring information leakage. Starting with the classical notion of entropy proposed by Shannon, some of this work has moved towards analyzing alternative notions of entropy. These choices are often motivated by ad hoc synthetic scenarios that bear no clear relationship to protocols in actual use.

We move away from this continuous measurement of information content to a discrete approach in terms of knowledge. To start with, we regard information as consisting of propositional facts, representing knowledge that has to be shared amongst agents. Initially, the eavesdropper does not know any of these facts.

As the protocol evolves and the honest agents participating in the conversation learn facts about each other, the eavesdropper also comes to know certain facts about the system. The goal is to have *informative* protocols that share knowledge effectively, but are still *safe* in terms of leaking this knowledge to an intruder.

Concretely, we focus on problems involving sets of agents holding cards on which distinct numbers are written. Each hand is initially known only to the agent who holds it. The agents' aim is to learn about each others' hands through public announcements, while revealing as little as possible to an eavesdropper.

An example is the Russian Cards problem with distribution $\langle k_1|k_2|k_3\rangle$, denoting that A and B get k_1, k_2 cards respectively while the third player C gets k_3 cards. The objective of A and B is to communicate with each other so that they both eventually learn each other's cards, while C remains ignorant of **every** card. An in-depth analysis of this problem in terms of the logic of public announcements can be found in [9].

One generalization of the Russian Cards setting is the Secure Aggregation of Distributed Information (SADI) problem, where there are k agents and an eavesdropper $\mathcal{E}$. The distribution of cards is then given by $\langle n_1|\ldots|n_k\rangle$, with n_i denoting the number of cards that honest agent i holds. The eavesdropper $\mathcal{E}$ does not receive any cards. The objective is to come up with protocols such that all the honest agents learn each others' cards while $\mathcal{E}$ remains ignorant of the location of at least some, if not all, cards. The SADI problem is analyzed in [5].

We present an approach to the SADI problem based on searching through the state space of a transition system. Each state of the transition system describes the knowledge of the individual agents, in terms of atomic propositions of the form *A knows that B has card i* and *A knows that B does not have card i*. Each announcement updates these knowledge propositions. To effect this update, we set up rules linking the propositions and use a SAT solver to compute the set of possible states after each announcement.

The updates we compute are *first order*—that is, they calculate the knowledge that has been revealed through the current sequence of announcements. However, we also need to capture *second order* knowledge—for instance, the given sequence should be compatible with more than one starting distribution of cards to prevent the eavesdropper from indirectly inferring the cards held by the honest agents from the choice of the announcement sequence. Using the formulation from [5], we show that such second order knowledge can also be captured using our transition system framework.

The paper is organized as follows. We set the framework for the SADI problem in Section 2. In the next section, we describe how we set up a transition system to analyze this problem. Section 4 describes how we can formulate and answer questions about information flow using our transition system. In Section 5 we describe some experimental results. We conclude with a discussion of future directions.

2 Preliminaries

Recalling the definitions from [5], the setting we consider involves a finite set of agents, Ag, with information distributed amongst them. Apart from the (*honest*) agents in Ag, there is also the *eavesdropper*. For convenience, if Ag consists of k honest agents, we assume that they are named $\{0, 1, \ldots, k-1\}$.

For our purposes, the information that the agents hold consists of a set of cards numbered $0, 1, \ldots, n-1$. These cards are distributed amongst the honest agents. In what follows, if X is a set and m is a natural number, then $\binom{X}{m}$ denotes the subsets of X of cardinality m. The cardinality of X is denoted by $\#X$.

It is assumed that there is a mechanism to distribute the cards initially, at the end of which each agent knows his own hand and the number of cards that everyone has been dealt, but nothing more. The problem is for the honest agents to learn each other's hands via public announcements without leaking information to the eavesdropper.

Definition 1. *The distribution type is a vector* $\bar{s} = (s_p)_{p \in Ag}$ *of natural numbers, where* s_p *denotes the number of cards dealt initially to agent* p*. We denote by* $|\bar{s}|$ *the total number of cards,* $\Sigma_{p \in Ag} s_p$*.*

A deal of type $\bar{s}$ *is a partition* $H = (H_p)_{p \in Ag}$ *of* $\{0, \ldots, |\bar{s}| - 1\}$ *such that* $\#H_p = s_p$ *for each agent* p*. We say* H_p *is the hand of* p*. Further, we denote the set of all deals over* $\bar{s}$ *by* $Deals(\bar{s})$*.*

Given two deals H *and* H' *of type* $\bar{s}$*, and an agent* p*, we say that* H *and* H' *are* indistinguishable *for* p *(in symbols:* $H \sim_p H'$*) if* $H_p = H'_p$*.*

The agents try to learn each other's hand by publicly (and truthfully) announcing information about their own cards. Consider an agent A holding the cards $\{1, 2, 3\}$. One announcement he might make is "My hand is either $\{1, 2, 3\}$ or $\{1, 4, 6\}$ or $\{2, 3, 5\}$." Any other agent who holds 4 and 5, on hearing this announcement, will immediately know that A's hand is $\{1, 2, 3\}$. But the eavesdropper still has uncertainty as to A's hand, since he doesn't have any cards of his own. Another possible announcement is "My cards are all among $\{1, 2, 3, 4, 6, 7\}$." Yet another announcement is "The sum of the numbers on my cards is 6." All these announcements can be encoded as a disjunction of hands. For instance, the last announcement above is the disjunction "My hand is either $\{1, 2, 3\}$ or $\{0, 2, 4\}$."

Definition 2 (Actions). *Fix a distribution type* $\bar{s}$*, and let* $\mathsf{Cards} = \{0, \ldots, |\bar{s}| - 1\}$*. An announcement by agent* p *is a disjunction of possible hands. Since he has* s_p *cards, the announcement can be thought of as a subset of* $\binom{\mathsf{Cards}}{s_p}$*. Thus we can define* Act_p*, the set of* p*-announcements to be* $\mathcal{P}(\binom{\mathsf{Cards}}{s_p})$*, where* $\mathcal{P}(X)$ *denotes the powerset of* X*. The set of actions is defined to be* $Act = \bigcup_{p \in Ag} Act_p$*.*

We assume a situation in which agents take turns to make announcements, starting from 0 and proceeding in cyclic order, till they achieve a certain goal. This is formalized by the following definition.

Definition 3 (Runs). *Fix a distribution type $\bar{s}$ as before, with m agents. An* execution *is a (finite or infinite) sequence of actions $\alpha_0\alpha_1\cdots$ such that $\alpha_i \in Act_p$ whenever $i \bmod m = p$. A finite execution is also called a* run*. Given a run $\rho = \alpha_0\alpha_1\ldots\alpha_n$ and two indices $i \leq j$, $\alpha[i\ldots j]$ denotes the segment $\alpha_i\alpha_{i+1}\ldots\alpha_j$. We denote the length of a run ρ by $|\rho|$. The set of runs is denoted by Runs.*

A protocol describes a strategy for each agent to make announcements given the current history. We constrain each announcement to be truthful. We also insist that a protocol depend only on the local information available to the agent—any two situations that are the same from the agent's point of view must elicit the same response. In other words, if the agent holds the same hand in two different deals and sees the same sequence of announcements, he must respond identically in both situations.

Definition 4 (Protocol). *Fix a distribution type $\bar{s}$, with m agents. A* ***protocol*** *(for $\bar{s}$) is a function π assigning to every deal $H \in Deals(\bar{s})$, and every run $\rho \in Runs$ with $|\rho| \bmod m = p$, a non-empty set of p-actions $\pi(H,\rho) \subseteq Act_p$ such that:*

- *$H_p \in \alpha$ for all $\alpha \in \pi(H,\rho)$ (the announcement is* truthful*), and*
- *if $H \sim_p H'$, then $\pi(H,\rho) = \pi(H',\rho)$ (the announcement is* view-based*).*

A run of a protocol π is a pair (H,ρ) where $H \in Deals(\bar{s})$ and $\rho = \alpha_0\alpha_1\ldots\alpha_m$ is a run such that $\alpha_{i+1} \in \pi(H,\rho[0\ldots i])$ for every $i < m$. The set of runs of π is denoted by $Runs(\pi)$.

We are interested in protocols that are *informative* (all honest agents learn the whole deal) and *safe* (the eavesdropper is uncertain about the deal even after listening to all the announcements). We formalize these notions below.

Definition 5. *A run (H,ρ) of a protocol π is informative for an agent p if there is no execution (H',ρ) of π with $H \sim_p H'$ and $H \neq H'$. (i.e., there is no other starting deal that is consistent with p's hand and the subsequent announcements.)*

A protocol π is

- ***weakly informative (WI):*** *if every run of π is informative for some agent.*
- ***informative (I):*** *if every run of π is informative for every agent.*

The eavesdropper does not have any information about the deal to begin with. Hence, the actual deal could be any deal of the correct distribution type. As the honest agents communicate amongst themselves, the eavesdropper uses the information in the announcements to eliminate various deals from contention. At the end of a sequence of announcements, he will be left with a set of deals that are consistent with this sequence. This set must have at least one element, namely the actual deal. This set is formally defined below.

Definition 6. *Given a protocol π and a run ρ, define the (eavesdropper's) ignorance set $I_\pi(\rho)$ to be $\{H \mid (H,\rho) \in Runs(\pi)\}$.*

$I_\pi(\rho)$ is can be used to determine what the eavesdropper knows at the end of ρ.

Consider a situation where agent 0 holds the card 5 in every deal in $I_\pi(\rho)$. This means that the eavesdropper has ruled out all deals where agent 0 does not hold card 5 as being inconsistent with ρ. Hence, the run ρ has leaked information about the location of card 5 to the eavesdropper.

Consider another situation involving card 5, where agent 1 holds card 5 in some of the deals in $I_\pi(\rho)$, and agent 2 holds card 5 in the rest of the deals in $I_\pi(\rho)$. Here, even though the eavesdropper does not know exactly who holds the card 5, he is certain that no agent other than $\{1, 2\}$ holds card 5.

This leads us to the following two notions of safety of a card (at the end of a run).

Definition 7 (Safety of cards). *A run (H, ρ) of a protocol π is safe for the card c if for every agent p, there is a deal $G \in I_\pi(\rho)$ such that $c \notin G_p$.*

A run (H, ρ) of a protocol π is strongly safe for the card c if for every agent p, there are two deals $F, G \in I_\pi(\rho)$ such that $c \in F_p$ and $c \notin G_p$.

Thus, safety of a run means that the eavesdropper does not know for certain that an agent p has card c, but the eavesdropper may have concluded that p definitely does not have c. On the other hand, strong safety requires that the eavesdropper cannot conclude whether p holds c or p does not hold c.

We can lift the notion of safety from runs to protocols as follows.

Definition 8 (Safety of Protocols). *A protocol π is*

- ***deal safe:*** *if every run of π is safe for some card c. Equivalently, deal safety means that the eavesdropper does not learn the deal at the end of any run of π.*
- *p**-safe (for an agent** p**):*** *if every run of π is safe for all cards in H_p.*
- ***safe:*** *if every execution of π is safe for every card c.*
- ***strongly safe:*** *if every execution of π is strongly safe for every card c.*

In the rest of the paper, we will examine an approach to synthesize informative and safe protocols based on these definitions. Equivalently, our approach can be used to validate if a given protocol is informative and safe.

We assume that there are at least three honest agents, so that negative information of the form *p does not hold card c* does not automatically imply positive information of the form *q holds card c.*

3 Implementation

In this section, we describe a tool written in Python to search for informative and safe runs of a particular distribution type. Before presenting the details of the tool, we present an abstract transition system model for protocols.

3.1 Defining the transition system

Fix a distribution type $\bar{s}$ with k agents $\{0, 1, \ldots, k{-}1\}$. For convenience, we assume that the eavesdropper is agent k, with the understanding that agent k possesses no cards. Let $\{0, 1, \ldots, n{-}1\}$ be the set of cards dealt. We describe a transition system that tracks the uncertainty of every agent about the actual deal. Essentially, each agent implicitly stores a set of *valuations* that represent all deals that are compatible with the information that he has seen so far.

Definition 9 (Valuations). *The set of* knowledge propositions *for agent $p \leq k$, denoted $\mathcal{K}(p)$, is the set*

$$\{K_{pq}(c), K_{pNq}(c) \mid q < k, q \neq p, c < n\}.$$

The proposition $K_{pq}(c)$ describes the fact that agent p knows that agent q has card c, while the proposition $K_{pNq}(c)$ says that p knows that q does not have card c.

Definition 10. *A* valuation *for agent p (with respect to an initial deal H) is a function $v : \mathcal{K}(p) \to \{\top, \bot\}$ satisfying the following conditions, where we write $v \models \ell$ to mean $v(\ell) = \top$ and $v \not\models \ell$ to mean $v(\ell) = \bot$.*

- ***consistency with the deal:*** *For all $0 \leq c < n$, if $c \in H_p$ then for all $0 \leq q < k$, $p \neq q$, $v \models K_{pNq}(c)$.*
- ***consistency of knowledge propositions:*** *For all $0 \leq q < k$ and $0 \leq c < n$, either $v \not\models K_{pq}(c)$ or $v \not\models K_{pNq}(c)$.*
- ***ownership of cards:*** *For all $0 \leq q < k$ and $0 \leq c < n$, $v \models K_{pq}(c)$ iff for all $r \notin \{p, q\}$, $v \models K_{pNr}(c)$*
- ***consistency with the distribution type:*** *For each $q \neq p$, there are at most s_q propositions of the form $K_{pq}(c)$ such that $v \models K_{pq}(c)$.*
- ***complete knowledge:*** *For each $q \neq p$, there are exactly s_q propositions of the form $K_{pq}(c)$ such that $v \models K_{pq}(c)$ iff there are exactly $(n - s_q)$ propositions of the form $K_{pNq}(c)$ such that $v \models K_{pNq}(c)$.*

We denote the set of all valuations for agent p by $Vals_p$, and let $Vals = \bigcup_{p \leq m} Vals_p$.

A valuation for p is supposed to capture p's information state. If v maps $K_{pq}(c)$ to $\top$, it means that p believes, in this information state, that q has card c. If v maps $K_{pNq}(c)$ to $\top$, it means that p believes that q does not have card c. If v maps both $K_{pq}(c)$ and $K_{pNq}(c)$ to $\bot$, this means that p is uncertain about whether or not q holds card c.

Assuming three agents $\{0, 1, 2\}$, an example valuation for agent 0 is given in Figure 1. This corresponds to the initial deal $(\{0, 1\}, \{2, 3, 4\}, \{5, 6, 7, 8\})$.

Each agent has a valuation describing the initial state according to his perspective, but as he hears more and more announcements (which are all disjunctions), it might not be possible to represent the information he has by means of one valuation. Each disjunct in the announcement he hears might lead him

	0	1	2	3	4	5	6	7	8
K_{01}	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
K_{02}	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
K_{0N1}	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
K_{0N2}	$\top$	$\top$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

Fig. 1. Valuation for agent 0 at initial deal $\langle 0, 1|2, 3, 4|5, 6, 7, 8\rangle$

to consider a new valuation as a possible world. Thus each agent might need to store a set of valuations as a run progresses. This leads to the following definition of *states*.

Definition 11 (States). *A state is an $k+1$-tuple $(V_0, \ldots, V_k)$ where each $V_p \subseteq Vals_p$.*

Definition 12 (X-extension). *Let v be a q-valuation and X be an s_p-sized subset of $\{0, \ldots, n-1\}$. We say that a q-valuation v' is a X-extension of v if $v \leq v'$ and $v' \models K_{qp}(c)$ for all $c \in X$.*

Note that for a given v and X, it is possible that there is no X-extension of v.

Definition 13 (State updates). *Given a state $\mathbf{s} = (V_0, \ldots, V_k)$ and a p-announcement α, we define update$(\mathbf{s}, \alpha)$ to be $\mathbf{s}' = (V'_0, \ldots, V'_k)$ such that:*

- $V'_p = V_p$
- *for $q \neq p$, $V'_q = \bigcup_{v \in V_q, X \text{ is a disjunct of } \alpha} \{v' \mid v' \text{ is an } X\text{-extension of } v\}$.*

The core component of our tool computes the updated state at the end of a sequence of announcements, starting from an initial deal. The final state encapsulates a nontrivial amount of information, from which one can test whether the honest agents have complete knowledge of the deal, and also whether the eavesdropper knows the original deal. Furthermore, we can also generate the uncertainty set of the eavesdropper, namely, the set of all deals that are compatible with the given announcement sequence. We can use this information in a variety of ways, as detailed in Section 4.

The important point to note is that the tool computes the updated state *implicitly*. Rather than explicitly maintain a set of valuations after each announcement, the tool just collects all the announcements, and invokes the SAT solver to determine the certain knowledge of the agents.

3.2 A high-level description of the tool

In this section, we describe in more detail some key components of our tool. In the interests of space, we present a high-level overview[1]. The tool is written in

[1] Please check `http://www.cmi.ac.in/~spsuresh/projects/russian-cards-z3/` for the full code.

Python, and implements the system described in the previous section. It interacts with the SAT solver Z3 to compute the updated state after each announcement. As detailed in the definition of valuations, each valuation has to satisfy a lot of constraints. These are coded as system invariants. A solver instance is created and formulas corresponding to the constraints are added to the solver, as shown in the following snippet.

```
solvR = Solver()
solvR.push()
vD, oW, hK = self.validDeal(), self.ownership(), self.hand2K()
kC, oK, dK = self.kConsistency(), self.ownershipK(), self.dealK()
solvR.add(vD)
solvR.add(oW)
solvR.add(hK)
solvR.add(kC)
solvR.add(oK)
solvR.add(dK)
solvR.push()
return solvR
```

For instance, `validDeal` corresponds to the constraint that if $c \in H_p$ then $v \models KpNq(c)$, while `ownershipK` corresponds to the constraint that $v \models K_{pq}(c)$ iff $v \models K_{pNr}(c)$ for all $r \neq q$. Similarly for the other constraints.

Given any announcement as a tuple consisting of the speaker as well as the DNF formula, the `updateAnn` procedure produces a new state with the appropriate update to the knowledge of each agent. Essentially, it translates the announcement to an appropriate formula denoting how each agent other than the speaker would perceive it and this is appended to the knowledge of each listener.

To process a sequence of announcements, the tool does not calculate the set of valuations at each intermediate state. Rather, repeated calls to `updateAnn` are made, which builds a conjunction of the initial knowledge, the constraints, and all the announcements. Now we can call the SAT solver to check what all propositions are consequences of this formula, and thus determine all the propositions that each agent is certain about.

The tool is meant to go through a set of runs of bounded length, each run consisting of announcements of a specific structure (typically a bound on the number of disjunctions in the announcement), and compute various statistics at the end of each run. For instance, we might want to know how many propositions of the form $K_{mp}(c)$ is definitely known to m (remember $\{0, \ldots, m-1\}$ is the set of honest agents and m is the eavesdropper) at the end of a run. This measures the amount of positive knowledge leaked. We might also want to check how many propositions of the form $K_{mNp}(c)$ is definitely known to m (i.e., for every valuation $v \in V_m$ in the final state, $v \models K_{mNp}(c)$). This measures the amount of negative knowledge leaked. We can use this tool to either discover protocols or check whether a purported protocol is informative and safe, as elaborated in Section 4.

3.3 An example

In this example, we illustrate the functioning of our tool with the $\langle 2|3|4 \rangle$ SADI problem detailed in [5]. Recall that the initial deal is $\langle 0, 1|2, 3, 4|5, 6, 7, 8 \rangle$, and an informative and safe announcement sequence is the announcement $\{01, 08, 18\}$ by A, followed by the announcement $\{0234, 1237, 5678\}$ by C. (B passes its turn – by announcing **true**, for instance).

First, we need to create and initialize the problem instance

```
In [1]: from cpState import *
In [2]: deal = {’a’:[0,1], ’b’:[2,3,4], ’c’:[5,6,7,8], ’e’:[]};
In [3]: infAgts, eaves = [’a’, ’b’, ’c’], ’e’;
In [4]: agts = infAgts + [eaves]
In [5]: s0 = cpState([2, 3, 4, 0], agts, deal, infAgts, eaves)
```

At the end of the above commands, we obtain the initial state s_0 initialized with the deal $\langle 0, 1|2, 3, 4|5, 6, 7, 8 \rangle$. Having obtained the initial state, we now need to update it with the announcements ann_1 of A followed by ann_2 of C.

```
In [6]: ann1 = (’a’, [[0, 1], [0, 8], [1, 8]])
In [7]: ann2 = (’c’, [[0, 2, 3, 4], [1, 2, 3, 7], [5, 6, 7, 8]])
In [8]: s1 = s0.updateAnn(ann1)
In [9]: s2 = s1.updateAnn(ann2)
```

Now that we have the resulting state s_2 alongwith the the intermediate state s_1, we can actually analyze the states and query them to obtain further information about the states of any agent in each of the states.

We can obtain the set of all positive knowledge propositions for any agent

```
In [10]: %time s2.getPosK(’a’)
CPU times: user 3.42 s, sys: 32 ms, total: 3.45 s
Wall time: 3.45 s
Out[10]: [’Kab__2’, ’Kab__3’, ’Kab__4’, ’Kac__5’, ’Kac__6’,
                    ’Kac__7’, ’Kac__8’]

In [11]: %time s2.getPosK(’b’)
CPU times: user 2.95 s, sys: 0 ns, total: 2.95 s
Wall time: 2.95 s
Out[11]: [’Kba__0’, ’Kba__1’, ’Kbc__5’, ’Kbc__6’,
                    ’Kbc__7’, ’Kbc__8’]

In [12]: %time s2.getPosK(’c’)
CPU times: user 2.46 s, sys: 4 ms, total: 2.47 s
Wall time: 2.47 s
Out[12]: [’Kca__0’, ’Kca__1’, ’Kcb__2’, ’Kcb__3’, ’Kcb__4’]

In [13]: %time s2.getPosK(’e’)
CPU times: user 4.43 s, sys: 16 ms, total: 4.44 s
```

```
Wall time: 4.45 s
Out[13]: []
```

As observed, the above queries take about 3 to 4 seconds even for this simple example. However, if all we are interested is in the informativity property, we can reduce the time taken by using `isInfAgt` or `isInformative` as shown below,

```
In [14]: %time s2.isInfAgt(a)
CPU times: user 472 ms, sys: 0 ns, total: 472 ms
Wall time: 480 ms
Out[14]: True

In [15]: %time s2.isInfAgt(b)
CPU times: user 468 ms, sys: 0 ns, total: 468 ms
Wall time: 471 ms
Out[15]: True

In [16]: %time s2.isInfAgt(c)
CPU times: user 472 ms, sys: 0 ns, total: 472 ms
Wall time: 476 ms
Out[16]: True

In [17]: %time s2.isInformative(infAgts)
CPU times: user 1.48 s, sys: 0 ns, total: 1.48 s
Wall time: 1.48 s
Out[15]: True

In [16]: %time s2.isInformative([eaves])
CPU times: user 508 ms, sys: 0 ns, total: 508 ms
Wall time: 509 ms
Out[16]: False
```

Hence, we've ascertained that the state s_2 is informative to a, b and c but e doesn't learn the owner of any card. This tallies with the analysis in [5]. However, we can also query the states for negative propositions revealed to `e`,

```
In [17]: %time s1.getNegK(’e’)
CPU times: user 8.93 s, sys: 0 ns, total: 8.93 s
Wall time: 8.97 s
Out[17]: [’KeNa__2’, ’KeNa__3’, ’KeNa__4’, ’KeNa__5’,
                     ’KeNa__6’, ’KeNa__7’]

In [18]: %time s2.getNegK(’e’)
CPU times: user 8.86 s, sys: 4 ms, total: 8.87 s
Wall time: 8.89 s
Out[18]: [’KeNa__2’, ’KeNa__3’, ’KeNa__4’, ’KeNa__5’,
 ’KeNa__6’, ’KeNa__7’, ’KeNb__0’, ’KeNb__1’, ’KeNb__8’]
```

From the above, it is clear that even though e doesn't know any of the owners of any cards, he does know 9 propositions of negative information involving cards $[0, 8]$. In fact, for each of the cards, he knows at least one agent that *does not own the card.* Thus, for any card, e initially did not know which of the 3 agents it belonged to, but at the end, his uncertainty is restricted to 2 of the 3 agents.

4 Formulating information leakage problems

We can use the transition system defined in the previous section in two ways: to search for an informative and safe protocol, and to validate if a given protocal is informative and safe.

4.1 Synthesis of protocols

We can characterize informative states based on the knowledge propositions of the agents—in an informative state, each agent should know completely all the cards of the agents.

Likewise, safety can be described in terms of the knowledge propositions of the eavesdropper. Unlike the classical SADI problem, we can *quantify* the level of safety we tolerate by placing a threshold on the knowledge revealed to the eavesdropper.

Our first task is to identify a set of runs that lead to informative and safe states. We call such a run a first-order informative run. It suffices to start the search at a fixed initial state corresponding a canonical distribution of cards. Every other deal is a permutation of this deal. If we can find a protocol for this starting deal, we can construct a symmetric protocol for every other deal by permuting all announcements in the same manner as the initial deal.

Having identified first order informative runs (through, say, depth-first search) we have to check if they satisfy *second order safety* and if they meet the view-based criterion laid down for protocols.

First order safety guarantees that the eavesdropper has at least two possible deals in his ignorance set at the end of each such run. However, this does not guarantee that the *same* run, starting from one of the alternative states, achieves informativeness and safety. This is what we call second order safety. Our first task, therefore, is to identify a set of first order informative runs that is closed with respect to this pairing: for every run ρ starting from the initial deal H, there is another deal H' such that ρ from H' is informative and safe.

Having identified such a set of runs closed with respect to second order safety, we then ensure that there is a subset that is view-based—that is, if $H \sim_p H'$, then the choice made after any sequence of announcements ρ starting from H matches that made after ρ starting from H'.

After these two stages of pruning, any first order informative runs that survive constitute a protocol that solves the given problem.

4.2 Verification of protocols

Conversely, we can use our transition system to verify a given protocol. This follows conventional lines, where we search for a state that is reachable with respect to the protocol that violates the given safety requirement.

5 Experimental results

In this section, we document some experiments with our tool on a 3 agent SADI system. To assist readability, we refer to the 3 honest agents as $\{A, B, C\}$ rather than $\{0, 1, 2\}$. The eavesdropper is denoted $\mathcal{E}$.

5.1 Need for second order safety

To illustrate the inadequacy of direct first order reasoning, we consider an example with the deal type $\langle 2, 3, 4\rangle$. Suppose the initial deal is $d_0 = \langle 0, 1|2, 3, 4|5, 6, 7, 8\rangle$. Let the first announcement (ann_1) by A be $\{01, 12, 23\}$. This would result in B obtaining complete knowledge of the deal (as he has card 2). B could then make the second announcement (ann_2) as $\{234, 056, 178\}$ to inform the others.

The only alternative deals compatible with the above announcements are

a) $d_1 = (\{1, 2\}, \{0, 5, 6\}, \{3, 4, 7, 8\})$
b) $d_2 = (\{2, 3\}, \{1, 7, 8\}, \{0, 4, 5, 6\})$
c) $d_3 = (\{2, 3\}, \{0, 5, 6\}, \{1, 4, 7, 8\})$

One may check that none of these deals are completely informative to all three agents A, B and C when using the run ρ consisting of exactly the two announcements above. For the first deal, A's announcment is not informative to B and the run itself is not informative to any of the agents. So, (d_1, ρ) is not even weakly informative. The runs (d_2, ann_1) as well as (d_3, ann_1) are informative to exactly B and C respectively (the agent that has card 1) but, (d_2, ρ) is not informative to A and (d_3, ρ) is informative to neither A nor C.

Currently, our tool can evaluate announcement sequences for first order safety. It is straightforward to extend it to check for second order safety as defined above. However, one would need to check what one could handle with a naïve implementation of second order safety. In the next section, we describe experiments using our tool for first order safety of larger instances with varying parameters.

5.2 Hand Size and Informativity

For a particular deal type (say $\langle 4, 4, 4, 0\rangle$), and a particular initial deal (say $\langle\{0, 1, 2, 3\}|\{4, 5, 6, 7\}|\{8, 9, 10, 11\}\rangle$), we generate random truthful announcement sequences using the function `genRun` defined in `cpState`. Though an announcement is simply a disjunction of hands, it makes sense for an agent to reveal only partial information in every announcement. This can be done by uniformly restricting the size of each set in the announcement. We call this the size of the hand (*hSize*) revealed in the announcement. The function `genRun` accepts the following arguments

a) The size of each set in the announcement (*hSize*).
b) The number of sets (or disjuncts) in an announcement (*annLen*).
c) The number of announcements for each run (*runLen*).

The executions were generated for the deal type $\langle 4,4,4,0\rangle$, varying *hSize* in the range $\{1,2,3\}$. Furthermore, we also varied *annLen* to take values in $\{3,5,7\}$. The results obtained are presented in Tables 1, 2, 3. We also ran some experiments for the deal type $\langle 8,8,8,0\rangle$—the results are tabulated in Table 4. For all runs, we have fixed *runLen* as 6, denoting sequences of 6 announcements, corresponding to exactly 2 rounds across the 3 agents. Our eventual goal is to move beyond the 1 round protocols studied in the literature ([5, 9]).

The *hSize* column denotes the hand size used in the announcements. The other columns are labeled by sets of agents. The entries in the matrix denote the number of runs which were informative for the corresponding set of agents.

If we look at the tables for $\langle 4,4,4,0\rangle$, (that is, Tables 1, 2 and 3) we notice that, as we increase *hSize*, the number of runs which are informative for any agent increases. This reflects our intuition that more information is transferred when each part of the announcment reveals more details about the hand. Another expected outcome is that the number of cards revealed to $\mathcal{E}$ also increases with increasing *hSize*, as observed in Figure 2. Note that the number of cards revealed to $\mathcal{E}$ is not represented in the tables but was computed independently using the same set of runs.

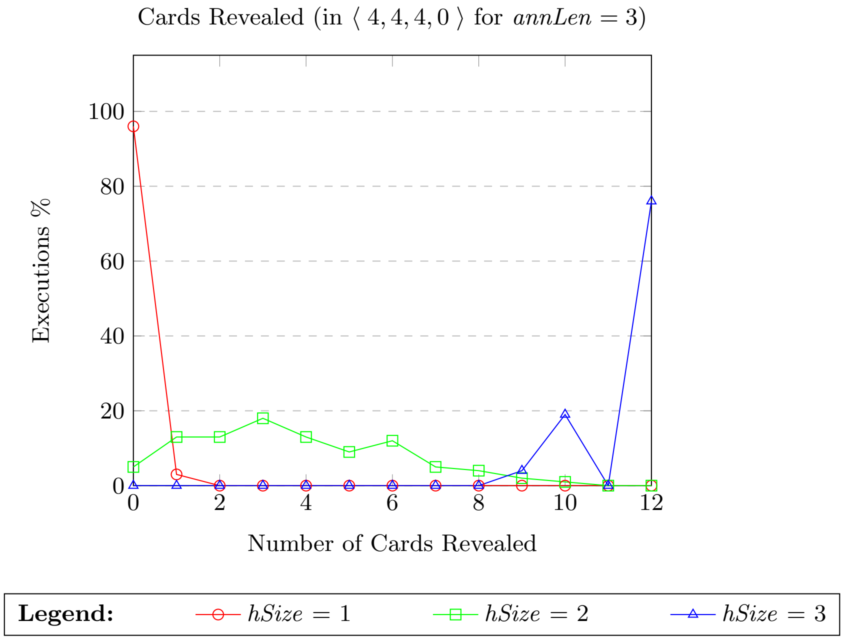

Fig. 2. Infomation leakage to $\mathcal{E}$

$hSize$	$\emptyset$	A	B	C	A,B	A,C	B,C	A,B,C	A,B,C,E
1	500	0	0	0	0	0	0	0	0
2	355	24	27	34	17	19	22	1	1
3	9	2	6	1	36	30	35	0	381

Table 1. Executions with $annLen = 3$ for $\langle 4,4,4,0\rangle$ (500 per entry).

$hSize$	$\emptyset$	A	B	C	A,B	A,C	B,C	A,B,C	A,B,C,E
1	500	0	0	0	0	0	0	0	0
2	434	23	19	16	0	4	3	1	0
3	12	2	8	8	41	38	42	4	345

Table 2. Executions with $annLen = 5$ for $\langle 4,4,4,0\rangle$ (500 per entry).

Notice that for $\langle 4,4,4,0\rangle$, setting $annLen = 7$ and $hSize = 3$, we obtained about 20 runs that were informative and safe. For the same configuration, we obtained a larger number of runs (about 169 runs or about 33% of the runs) that were informative for 2 of the agents.

One motivation for running these experiments is to identify parameters for which the probability of hitting an informative run is high. Once we identify informative runs, the next step would be to validate these runs with respect to second order reasoning of $\mathcal{E}$ in order to design or search for informative and safe protocols. We can also use these experiments to guide us towards impossibility proofs when a protocol does not exist.

6 Discussion

We have attempted to describe a framework for quantifying information flow in protocols in terms of discrete items of knowledge. We have used card playing protocols because the cards themselves act as natural units of knowledge.

We have identified two kinds of knowledge propositions: *A knows that B has card i* and *A knows that B does not have card i*. In the Russian Cards Problem where there are only two honest agents, these two are dual to each other. However, in the SADI framework, the second type of knowledge is strictly weaker than the first. Positive information about where a card is implies negative information about where the card is not to be found, but not vice versa.

We can thus impose a partial order on different knowledge states of the eavesdropper and use this to rank different protocols according the amount of information that they reveal. A challenge would then be to synthesize an optimal protocol with respect to this information ordering.

A more ambitious extension would be to extend our analysis to settings such as bidding in the game of bridge. In the bridge bidding process, each pair tries to

$hSize$	$\emptyset$	A	B	C	A,B	A,C	B,C	A,B,C	A,B,C,E
1	500	0	0	0	0	0	0	0	0
2	480	7	8	5	0	0	0	0	0
3	12	12	13	10	55	59	55	20	264

Table 3. Executions with $annLen = 7$ for $\langle 4,4,4,0\rangle$ (500 per entry).

$hSize$	$\emptyset$	C	A, B	A, C	B, C	A, B, C, E
2	250	0	0	0	0	0
4	243	0	4	2	1	0
6	38	1	30	19	31	131

Table 4. Executions with $annLen = 3$ for $\langle 8, 8, 8, 0\rangle$ (250 per entry)

understand the strength of the team's hand without revealing too much to the opponent. Bids are restricted to be an ascending sequence of announcements and the number of such announcements is fixed a priori. Hence, the goal is to maximize the information shared between partners while minimizing the information revealed to the opponents.

References

1. Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Probability of error in information-hiding protocols. In *20th IEEE Computer Security Foundations Symposium, CSF 2007, 6-8 July 2007, Venice, Italy*, pages 341–354. IEEE Computer Society, 2007.
2. David Clark, Sebastian Hunt, and Pasquale Malacaria. Quantitative information flow, relations and polymorphic types. *J. Log. Comput.*, 15(2):181–199, 2005.
3. David Clark, Sebastian Hunt, and Pasquale Malacaria. A static analysis for quantifying information flow in a simple imperative language. *Journal of Computer Security*, 15(3):321–371, 2007.
4. Michael R. Clarkson, Andrew C. Myers, and Fred B. Schneider. Belief in information flow. In *18th IEEE Computer Security Foundations Workshop, (CSFW-18 2005), 20-22 June 2005, Aix-en-Provence, France*, pages 31–45. IEEE Computer Society, 2005.
5. David Fernández-Duque and Valentin Goranko. Secure aggregation of distributed information: How a team of agents can safely share secrets in front of a spy. *Discrete Applied Mathematics*, 198:118 – 135, 2016.
6. Boris Köpf and David A. Basin. An information-theoretic model for adaptive side-channel attacks. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 286–296. ACM, 2007.
7. Gavin Lowe. Quantifying information flow. In *15th IEEE Computer Security Foundations Workshop (CSFW-15 2002), 24-26 June 2002, Cape Breton, Nova Scotia, Canada*, pages 18–31.
8. Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Approximate non-interference. In *15th IEEE Computer Security Foundations Workshop (CSFW-15 2002), 24-26 June 2002, Cape Breton, Nova Scotia, Canada*, pages 3–17.
9. Hans P. van Ditmarsch, Wiebe van der Hoek, Ron van der Meyden, and Ji Ruan. Model checking russian cards. *Electr. Notes Theor. Comput. Sci.*, 149(2):105–123, 2006.