We describe a few intuitive strategies that are clearly informative across deal types and we then analyse that strategy for information leakage to the eavesdropper.

1 Direct Exchanges

1.1 Strategy 1

Given the deal type $\langle k_1, k_2, k_3 \rangle$, such that $k_1 = k_2 = k_3 = k$, do the following,

- A picks a card x that he does not have and announces $H_A \vee D_1 \vee \ldots D_n$ such that for any i, (where $H_A = a_1 \ldots a_{k_1}$),

$$D_i = a_1 \wedge \ldots a_{i-1} \wedge x \wedge a_{i+1} \ldots a_{k_1}$$

- $P \in \{B, C\}$, who has card x responds with the announcement $H_P \vee D_1 \vee D_2$ where the D_i are constructed as follows,
  - Choose 2 cards $a_1, a_2 \in H_A$
  - Pick card $x_1 \in H_Q$ where Q is neither P nor A
  - Let X be either $H_Q \backslash x_1$ or $H_P \backslash x$
  - Finally, $D_1 = a_1 \wedge (H_Q \backslash x_1)$ and $D_2 = a_2 \wedge (X)$
- Note that both announcements are sorted before announcing.

The above strategy is implemented in python and the strategy is analysed in the next section. Here, we illustrate how we go about setting up the system alongwith sample announcements generated.

```
> a,b,c,e = 'a', 'b', 'c', 'e';
> dL,agtL, cards = [4,4,4,0],[a,b,c,e], range(12);
> deal = ut.minDeal(dL, agtL, cards); st = cps.cpState;
> s0 = st(dL,agtL, deal, [a,b,c], e)

> ann1, x = firstAnn(s0, deal,a); ann2 = secondAnn(s0, deal, ann1)
> ann1, ann2, x
 (('a', [[0, 1, 2, 5], [0, 1, 3, 5], [0, 2, 3, 5], [1, 2, 3, 5]]),
  ('b', [[1, 4, 6, 7], [2, 8, 9, 11], [4, 5, 6, 7]]),
  5)
```

In the above run, x was chosen as 5, hence P was B due to x and finally X was taken to be $H_P \backslash x$.

1.2 Strategy 2

Vary strategy 1 by modifying the second announcement by allowing the number of disjuncts to be exactly equal to the number of disjuncts in first announcement.

Basically choose D_i where $i \in [1, k]$ such that

a) $D_i = a_i \wedge X_i$ where $X_i \subseteq (H_P \setminus x) \cup (H_Q)$ and $|X_i| = (k-1)$.
b) For all i, $(X_i \cap H_Q) \neq \emptyset$ and $(\bigcap_i X_i) \cap H_P = \emptyset$
c) Finally, $X_Q \subset (\cup_i X_i)$.

The second player announces $(H_P \cup (\bigcup_i D_i))$. To choose X_i, we have $\binom{2k-1}{k-1}$ possibilities and it's clear that one can choose a collection of such sets satisfying the conditions. In fact, one very straightforward way is to ensure that $X_1 \subset H_Q$ in addition to the second condition above. We claim that the above is a safe (even with second order reasoning) protocol.

1.3 Three Steps and Beyond

The advantage of the previous strategies was that the information exchange was short and took place in exactly two meaningful announcements (ignoring the announcements that said "pass"). However, as we can observe none of the runs are actually strongly safe and leaks information to the Eavesdroppper. In order to rectify this, we need to go back to the first announcement and analyse it more carefully which we have done in the next section.

2 Analysis

2.1 Strategy 1

In this section, we tabulate some of the results of running the above strategies for various values of k ranging from 2 to 17 as shown in Table 1. We observe that as we increase k, the number of positive propositions leaked is is close to k whereas the number of negative propositions learnt is about three to five times the value of k.

k	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
+ve	2	3	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-ve	8	12	15	19	23	27	31	35	39	43	48	51	55	59	63	67
#deals	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Table 1: Information leakage across deal types

2.2 Strategy 2

For strategy 2, we ran some experiments for various deal types of the form $\langle i, i, i, 0\rangle$ and the results are tabulated in Table 2. The fact that the number of positive propositions leaked is 0 conforms to our assumption that it is safe.

k	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
+ve	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-ve	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51
#deals	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

Table 2: Information leakage with Strategy 2

2.3 Three Steps and Beyond

In what follows, we assume that the eavesdropper has knowledge of the protocol. This in turn means we can restrict our attention to protocols such that each agent "says what he means and means what he says".

To further strengthen the protocol, we need to go back to the first announcement. We note that it is not strongly safe and hence we need to change it if we need to strengthen it further. In order to maintain the information content transferred by Eave in the very first step starting from the minimal deal d_0, we will have to choose a particular card (say 5) and pair it with other cards to complete a disjunct. If we have D_1, D_2 in the first announcement such that $D_1 \cap D_2 = \emptyset$, then, there will be deals such that B and C will be uncertain.

We set up the experiment as follows,

```
> dLst = ut.allDeals([4,4,4], [a,b,c], range(12))
> len(dLst)
34650
> cl  = ut.allHands(4, range(12))
> cl5 = filterHands(cL, [5])
> ann1L = [[0,1,2,3]] + cl5
> len(ann1L)
166
> ann1 = ('a', ann1L)
```

As noted above, ann1 is our required first announcement that we are going to analyze with respect to second order reasoning for the agents B and C. It's clear that ann1 is informative to at least one of B or C as long as A has the same hand as he holds in the minimal deal. However, what happens when he holds other hands that are a part of ann1?

```
> dL1 = filterDeals(dLst, a, cl5 + [[0,1,2,3]])
> len(dL1)
11620
```

Basically, we first obtain the set of all deals where A may announce $ann1$. There are exactly $70 * (165 + 1) = 11620$ of such deals and we obtained that set as shown above. Now, we may analyze this set for leakage to each of the agents B and C. The code for computing the leakage across deals is as given below,

```
> def computeLeakage(s0, dLst, ann1, agt):
```

```
    kb = []
    for d in dLst:
      r0 = s0.newState()
      r0.deal = d
      rF = r0.updateAnn(ann1)
      kb.append(rF.getCards(agt))
      kc.append(rF.getCards(c))
    return kb, kc
> %time kb.kc = computeLeakage(s0, dL1, ann1)


In [20]: fAnn
Out[20]:
('a',
 [[0, 1, 2, 3],
  [0, 1, 2, 11],
  [0, 1, 3, 11],
  [0, 2, 3, 11],
  [1, 2, 3, 11],
  [0, 3, 4, 11],
  [2, 5, 6, 11],
  [4, 6, 7, 11]])
> kB = computeLeakage(s0, dL, fAnn, b)
> kC = computeLeakage(s0, dL, fAnn, c)

In [31]: %time kB = computeLeakage(s0, dL, fAnn, b)
CPU times: user 47min 14s, sys: 3.3 s, total: 47min 17s
Wall time: 47min 24s

In [32]: %time kc = computeLeakage(s0, dL, fAnn, b)
CPU times: user 47min 21s, sys: 4.15 s, total: 47min 25s
Wall time: 50min 47s

In [33]: fAnn
Out[33]:
('a',
 [[0, 1, 2, 3],
  [0, 1, 2, 7],
  [0, 1, 3, 7],
  [0, 2, 3, 7],
  [1, 2, 3, 7],
  [4, 6, 7, 9],
  [0, 2, 7, 9],
  [0, 3, 4, 7],
  [4, 5, 7, 11],
  [2, 7, 8, 11],
  [1, 4, 7, 10]])
```

A First Announcement

```
> cl5 = filterHands(cL, [5])
> displayHands(cl5, 5)
[0, 1, 2, 5] [0, 1, 3, 5] [0, 1, 4, 5] [0, 1, 5, 6] [0, 1, 5, 7]
[0, 1, 5, 8] [0, 1, 5, 9] [0, 1, 5, 10] [0, 1, 5, 11] [0, 2, 3, 5]
[0, 2, 4, 5] [0, 2, 5, 6] [0, 2, 5, 7] [0, 2, 5, 8] [0, 2, 5, 9]
[0, 2, 5, 10] [0, 2, 5, 11] [0, 3, 4, 5] [0, 3, 5, 6] [0, 3, 5, 7]
[0, 3, 5, 8] [0, 3, 5, 9] [0, 3, 5, 10] [0, 3, 5, 11] [0, 4, 5, 6]
[0, 4, 5, 7] [0, 4, 5, 8] [0, 4, 5, 9] [0, 4, 5, 10] [0, 4, 5, 11]
[0, 5, 6, 7] [0, 5, 6, 8] [0, 5, 6, 9] [0, 5, 6, 10] [0, 5, 6, 11]
[0, 5, 7, 8] [0, 5, 7, 9] [0, 5, 7, 10] [0, 5, 7, 11] [0, 5, 8, 9]
[0, 5, 8, 10] [0, 5, 8, 11] [0, 5, 9, 10] [0, 5, 9, 11] [0, 5, 10, 11]
[1, 2, 3, 5] [1, 2, 4, 5] [1, 2, 5, 6] [1, 2, 5, 7] [1, 2, 5, 8]
[1, 2, 5, 9] [1, 2, 5, 10] [1, 2, 5, 11] [1, 3, 4, 5] [1, 3, 5, 6]
[1, 3, 5, 7] [1, 3, 5, 8] [1, 3, 5, 9] [1, 3, 5, 10] [1, 3, 5, 11]
[1, 4, 5, 6] [1, 4, 5, 7] [1, 4, 5, 8] [1, 4, 5, 9] [1, 4, 5, 10]
[1, 4, 5, 11] [1, 5, 6, 7] [1, 5, 6, 8] [1, 5, 6, 9] [1, 5, 6, 10]
[1, 5, 6, 11] [1, 5, 7, 8] [1, 5, 7, 9] [1, 5, 7, 10] [1, 5, 7, 11]
[1, 5, 8, 9] [1, 5, 8, 10] [1, 5, 8, 11] [1, 5, 9, 10] [1, 5, 9, 11]
[1, 5, 10, 11] [2, 3, 4, 5] [2, 3, 5, 6] [2, 3, 5, 7] [2, 3, 5, 8]
[2, 3, 5, 9] [2, 3, 5, 10] [2, 3, 5, 11] [2, 4, 5, 6] [2, 4, 5, 7]
[2, 4, 5, 8] [2, 4, 5, 9] [2, 4, 5, 10] [2, 4, 5, 11] [2, 5, 6, 7]
[2, 5, 6, 8] [2, 5, 6, 9] [2, 5, 6, 10] [2, 5, 6, 11] [2, 5, 7, 8]
[2, 5, 7, 9] [2, 5, 7, 10] [2, 5, 7, 11] [2, 5, 8, 9] [2, 5, 8, 10]
[2, 5, 8, 11] [2, 5, 9, 10] [2, 5, 9, 11] [2, 5, 10, 11] [3, 4, 5, 6]
[3, 4, 5, 7] [3, 4, 5, 8] [3, 4, 5, 9] [3, 4, 5, 10] [3, 4, 5, 11]
[3, 5, 6, 7] [3, 5, 6, 8] [3, 5, 6, 9] [3, 5, 6, 10] [3, 5, 6, 11]
[3, 5, 7, 8] [3, 5, 7, 9] [3, 5, 7, 10] [3, 5, 7, 11] [3, 5, 8, 9]
[3, 5, 8, 10] [3, 5, 8, 11] [3, 5, 9, 10] [3, 5, 9, 11] [3, 5, 10, 11]
[4, 5, 6, 7] [4, 5, 6, 8] [4, 5, 6, 9] [4, 5, 6, 10] [4, 5, 6, 11]
[4, 5, 7, 8] [4, 5, 7, 9] [4, 5, 7, 10] [4, 5, 7, 11] [4, 5, 8, 9]
[4, 5, 8, 10] [4, 5, 8, 11] [4, 5, 9, 10] [4, 5, 9, 11] [4, 5, 10, 11]
[5, 6, 7, 8] [5, 6, 7, 9] [5, 6, 7, 10] [5, 6, 7, 11] [5, 6, 8, 9]
[5, 6, 8, 10] [5, 6, 8, 11] [5, 6, 9, 10] [5, 6, 9, 11] [5, 6, 10, 11]
[5, 7, 8, 9] [5, 7, 8, 10] [5, 7, 8, 11] [5, 7, 9, 10] [5, 7, 9, 11]
[5, 7, 10, 11] [5, 8, 9, 10] [5, 8, 9, 11] [5, 8, 10, 11] [5, 9, 10, 11]
```