Exercise 9

Program Analysis for System Security and Reliability
ETH Zürich

May 14, 2018

Problem 1. In this task, we consider a secret password (for simplicity a number between 1 and 10, generated by `prior`) and a leak of that password, leaked by `leak`.

```
def password_prior (){
    // select a password uniformly at random from {1, ..., 10}
    return uniformInt(1, 10);
}
def leak (password){
    // original program leaks full password
    val := password
    // put enforcement here
    return val;
}
```

In the following questions, we investigate how to prevent the leakage of the password by different privacy policies.

1. Encode the privacy enforcement ξ defined by the following equivalence relation, as a program:
$$\xi := \{\{1,2,3\},\{4,5,6\},\{7,8,9\},\{10\}\}$$

2. What is the probability that the attacker can correctly guess the password, given the privacy enforcement ξ above?

3. What is the permissiveness of ξ? What is the precision of ξ?

4. Provide a privacy enforcement (this time as an equivalence relation), that enforces the following privacy policy:
$$\forall o \in O.\, \Pr[I \in \{1\} \mid O \in [o]_\xi\}] \in [0,1]$$
What is the most permissive enforcement that achieves this policy? What is the most precise one?

5. Provide a privacy enforcement (as an equivalence relation), that enforces the following privacy policy:

$$\forall o \in O.\, \Pr[I \in \{1\} \mid O \in [o]_{\xi}\}] \in \left[\frac{1}{10}, \frac{1}{10}\right]$$

   What is the most permissive enforcement that achieves this policy? What is the most precise one?

6. In general, is the most permissive enforcement unique? If so, why? If not, provide a counterexample (i.e., a privacy policy that does not have a single most permissive enforcement).

7. What about the most precise enforcement? Is it unique?

Problem 2. Consider the following program, that describes a prior knowledge on patients (`patient_prior`) and the medial cost of a patient, depending on the diseases that patient has (`medical_cost`). Here, `flip(p)` returns 1 with probability p, and 0 with probability $1 - p$.

```
// a patient is captured by an array of real numbers
def patient_prior: R[] (){
    // probability that patient has aids: 1%
    aids := flip(0.01);
    // probability that patient has the flu depends whether or not the
       patient has aids
    flu := 0;
    if aids{
        flu = flip(0.5);
    }else{
        flu = flip(0.1);
    }
    return [aids, flu];
}
def medical_cost (patient:R[]){
    cost := 0;
    if patient[0]{
            // cost for aids
            cost += 10000;
    }
    if patient[1]{
        // cost for flu
        if flip(0.5){
            // probability that doctor treats flu: 50%
            cost += 70;
        }
    }
    // put enforcement here
    return cost;
}
```

```
def run(){
    P := patient_prior()
    return medical_cost(P);
}
```

In the following, we want to protect the information whether a patient has AIDS, while releasing information on that patient's medial cost.

1. Let P be a random patient constructed by `patient_prior()`. What is the distribution of P?

2. Let C be the cost computed by selecting a random patient from P and then computing their medical cost `medial_cost(P)`, What is the distribution of C?

3. What is the posterior distribution of `medial_cost()`? Concretely, what is $\Pr[P = p \mid C = c]$? Provide these probabilities for all patients p and all costs c.

4. Run the greedy algorithm for permissive enforcement (from class) to derive a privacy enforcement for the following privacy policy (the policy limits the probability that the attacker can infer that the patient has AIDS):

   $$\forall o \in O.\, \Pr[I \in \{[1,0],[1,1]\} \mid O \in [o]_\xi\}] \in \left[0, \frac{1}{2}\right]$$

   If the most violating class is not unique, pick the class that contains the smallest cost. To simplify the algorithm, do not consider the distance of the merge candidates to the enforcement policy. Instead, pick the merge candidate that contains the smallest cost.