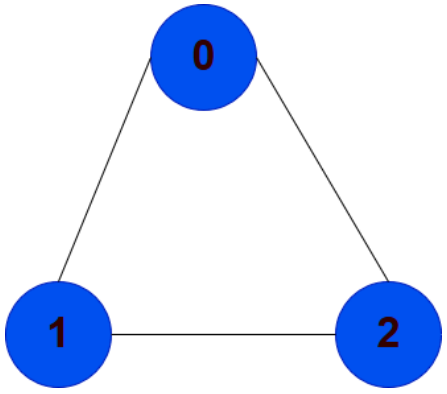
Ονοματεπώνυμο	Αλέξανδρος Φώτιος Ντογραματζής
ΑΜ	CS2.18.0010
Σχολή	Τμήμα Πληροφορικής & Τηλεπικοινωνιών
Όνομα Μεταπτυχιακού	ΠΜΣ Πληροφορικής
Ειδίκευση	Διαχείριση Δεδομένων, Πληροφορίας και Γνώσης
Ακαδημαϊκό Έτος	2018-2019
Μάθημα	Τεχνικές Ιδιωτικότητας
Είδος Εργασίας	Απαλλακτική Εργασία

Γενικά

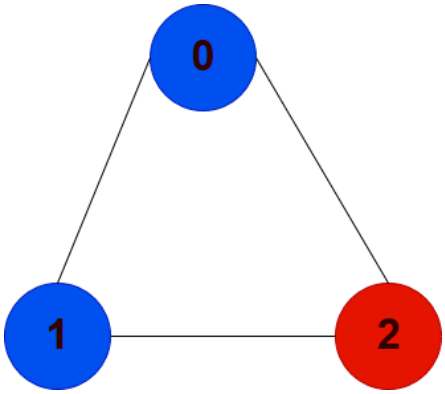
Συστήματα –Γράφοι Δικτύου

Κάθε σύστημα αναπαρίσταται από ένα γράφο-δίκτυο όπου οι κόμβοι του αναπαριστούν τους χρήστες τους δικτύου και οι ακμές του τις μεταξύ τους συνδέσεις. Στο πρόγραμμα οι γράφοι αναπαρίσταται από ένα αρχείο txt όπου κάθε γραμμή του αναπαριστά τις συνδέσεις ενός χρήστη με τους υπόλοιπους του δικτύου(0 αν δεν συνδέονται, 1 αν υπάρχει σύνδεση). Γίνεται η σύμβαση ότι οι corrupted χρήστες με όποιους χρήστες και να συνδέονται και σε οποιοδήποτε σημείο και να είναι στο δίκτυο, οι συνδέσεις τους αναπαρίστανται από τις τελευταίες γραμμές του αρχείου. Στις παρακάτω εικόνες οι corrupted χρήστες αναπαρίστανται από κόκκινους κόμβους και εικόνες οι honest χρήστες αναπαρίστανται από μπλε κόμβους.

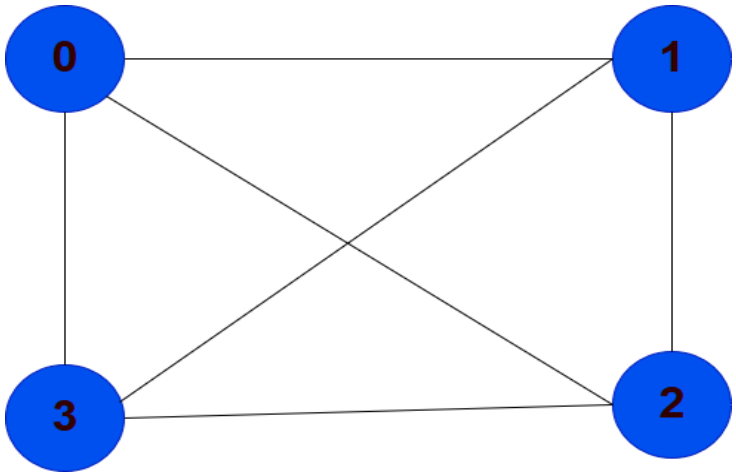
1. **3 χρήστες σε πλήρη γράφο, 0 corrupted user(s)**



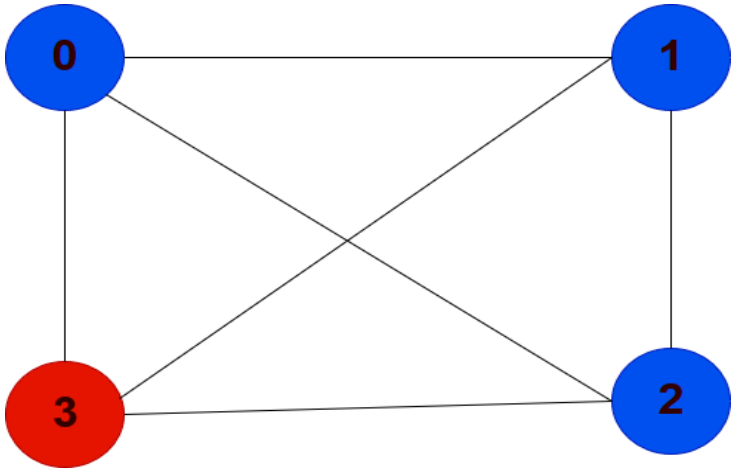
2. **3 χρήστες σε πλήρη γράφο , 1 corrupted user(s)**



3. **4 χρήστες σε πλήρη γράφο, 0 corrupted**

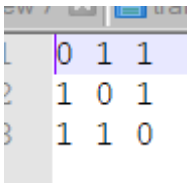


4. **4 χρήστες σε πλήρη γράφο, 1 corrupted**

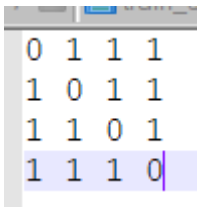


5. **4 χρήστες σε γραμμή, 1 corrupted user(s) (από τους μεσαίους)**

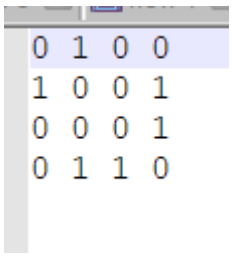
Τα 1,2 σε μορφή αρχείου:



Τα 3,4 σε μορφή αρχείου:



Το 5 σε μορφή αρχείου:



Τύποι Αρχικής Γνώσης Αντιπάλου

Η αρχική γνώστη του αντιπάλου αναπαρίσταται από μια κατανομή πιθανοτήτων και είναι ένα αρχείο txt με μια γραμμή όπου κάθε στοιχείο απεικονίζει την πιθανότητα να έχει στείλει ο χρήστης i (παίρνει τιμές 0-n-1,n:αριθμός έντιμων χρηστών δικτύου). Εδώ θα χρησιμοποιηθούν οι παρακάτω κατανομές:

- Για 2 έντιμους χρήστες: 0 και 1 έχουν πιθανότητα να στείλουν 0.6 και 0.4 αντίστοιχα(biased)

- Για 3 έντιμους χρήστες:
    1. Uniform: Θεωρείται ότι όλοι οι χρήστες έχουν πιθανότητα 1/3 να στείλουν
    2. Biased:Ο 0 και ο 2 πιθανότητα 0.5 και ο 1 πιθανότητα 0
- Για 4 έντιμους χρήστες: ομοιόμορφή όλοι οι χρήστες πιθανότητα 0.25

Οι παραπάνω κατανομές σε μορφή αρχείου:

```
0.6 0.4
```

```
0.5 0 0.5
```

```
0.33333 0.33333 0.33333
```

```
0.25 0.25 0.25 0.25
```

Θεωρητική Προσέγγιση

Για το σύστημα 1:

- Με γνώση αντιπάλου ομοιόμορφη κατανομή $\pi=\{p(x_0), p(x_1), p(x_2)\}=$ $\{0.33333,0.33333,0.33333\}$ και για πρωτόκολλο dc με δίκαια νομίσματα $cr=\{0,1\}=\{y_0, y_1\}$:$P(0)=P(1)=0.5$ προκύπτει ο πίνακας C με πιθανότητες $p(y_j|x_i)$:

$$\begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

και joint distribution περιέχει όλες τις πιθανότητες $p(y_j,x_i)=\pi_{x_i} C_{x_i y_j}$:

$$\begin{bmatrix} 1/12 & 1/12 & 1/12 & 1/12 \\ 1/12 & 1/12 & 1/12 & 1/12 \\ 1/12 & 1/12 & 1/12 & 1/12 \end{bmatrix}$$

και output distribution πC με $\delta_{y_j} = p(y_j) = \sum_{x_i} p(y_j, x_i)$:

$$\delta = \begin{bmatrix} \frac{3}{12} & \frac{3}{12} & \frac{3}{12} & \frac{3}{12} \end{bmatrix}$$

και προκύπτουν οι posterior distributions [π,C] με στοιχεία $\sigma_{x_i}^{y_j} = p(x_i|y_j)$:

$$\begin{bmatrix} 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 \end{bmatrix}$$

Υπολογίζεται το posterior vulnerability(πιθανότητα επιτυχίας αντιπάλου):

$V[\pi, C] = \sum_{y_j} \delta_{y_j} V(\sigma^{y_j})$ η οποία για Bayes vulnerability γίνεται:

$V[\pi, C] = \sum_{y_j} max_{x_i} \pi_{x_i} C_{x_i y_j}$ $= \frac{1}{3}\frac{1}{4}4 = \frac{1}{3}$

Για το σύστημα 2:

Το σύστημα 1 και το σύστημα 2 έχουν τον ίδιο γράφο και η μόνη τους διαφορά είναι ότι στο σύστημα 2 ο χρήστης 2 είναι corrupted.

Με γνώση αντιπάλου π={p(x_0), p(x_1)}={0.6,0.4 } biased και για πρωτόκολλο dc με δίκαια νομίσματα cr={0,1}={y_0, y_1}:P(0)=P(1)=0.5 προκύπτει ο πίνακας C με πιθανότητες p(y_j|x_i):

$$\begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

και joint distribution περιέχει όλες τις πιθανότητες p(y_j,x_i)=$\pi_{x_i} C_{x_i y_j}$:

$$\begin{bmatrix} 0.15 & 0.15 & 0.15 & 0.15 \\ 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}$$

και output distribution πC με $\delta_{y_j} = p(y_j) = \sum_{x_i} p(y_j, x_i)$:

$$\delta = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

και προκύπτουν οι posterior distributions [π,C] με στοιχεία $\sigma_{x_i}^{y_j} = p(x_i|y_j)$:

$$\begin{bmatrix} 0.6 & 0.6 & 0.6 & 0.6 \\ 0.4 & 0.4 & 0.4 & 0.4 \end{bmatrix}$$

Υπολογίζεται το posterior vulnerability:

$V[\pi, C] = \sum_{y_j} \delta_{y_j} V(\sigma^{y_j})$ η οποία για Bayes vulnerability γίνεται:

$V[\pi, C] = \sum_{y_j} max_{x_i} \pi_{x_i} C_{x_i y_j} = \frac{6}{10}\frac{1}{4} 4 = 0.6$

Η τιμή του posterior vulnerability που αντιστοιχεί στην πιθανότητα επιτυχία του αντιπάλου είναι 0.6 και είναι μεγαλύτερη από ότι στο σύστημα 1 που ο χρήστης 2 δεν ήταν corrupted.

Για τις παραπάνω περιπτώσεις όπου χρησιμοποιείται πρωτόκολλο DC, είναι γνωστό από τη θεωρία ότι το συγκεκριμένο πρωτόκολλο προσφέρει strong anonymity σε κάθε connected component του γράφου και δεν θα μπορούσαν να αποφευχθούν οι παραπάνω υπολογισμοί και να υπολογιστεί με πιο απλό τρόπο η πιθανότητα επιτυχίας του αντιπάλου. Υπολογίζεται η εκ των υστέρων γνώση του αντιπάλου δεδομένου ότι ξέρει σε ποιο component είναι ο χρήστης (οι χρήστες των άλλων connected components έχουν προφανώς πιθανότητα 0). Και μετά παίρνουμε το vulnerability αυτής της κατανομής (δηλαδή τη μεγαλύτερη πιθανότητα).

Για το σύστημα 3:

Για πρωτόκολλο crowds με πιθανότητα προώθησης φ=0.75 και m συνολικό αριθμό χρηστών,c:corrupted χρήστες και n=m-c(έντιμοι χρήστες, γενικά ισχύουν:

$$p(\perp | x_i) = a = C_{x_i, s} = 1 - \beta - (n-1)\gamma = \frac{n - n\varphi}{m - n\varphi}$$

$$p(y_i | x_i) = \beta = C_{x_i, y_i} = \frac{c(m - \varphi(n-1))}{m(m - \varphi n)}$$

$$p(y_i | x_j) = \gamma = C_{x_j, y_i} = \beta - \frac{c}{m} = \frac{c\varphi}{m(m - \varphi n)}$$

- Με γνώση αντιπάλου ομοιόμορφη κατανομή π={p(x_0), p(x_1), p(x_2),p(x_3)}=

{0.25,0.25,0.25,0.25}. Εδώ έχουμε m=n=4 και c=0 οπότε έχουμε :

α=1

β=0

γ=0

Επομένως το κανάλι-πίνακας του συστήματος C έχει μορφή:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

και joint distribution περιέχει όλες τις πιθανότητες $p(y_j,x_i)=\pi_{x_i}C_{x_iy_j}$:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 0 & 0.25 \\ 0 & 0 & 0 & 0 & 0.25 \end{bmatrix}$$

και output distribution πC με $\delta_{y_j} = p(y_j) = \sum_{x_i} p(y_j, x_i)$:

$$\delta = [0 \quad 0 \quad 0 \quad 0 \quad 1]$$

Εδώ οι posterior distributions [π,C] με στοιχεία $\sigma_{x_i}^{y_j} = p(x_i|y_j)$ ορίζονται μόνο όταν $y_j = \perp = s$ όπου $\delta_{y_j} \neq 0$ και επομένως έχουμε:

$$\sigma^s = \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{bmatrix}$$

Υπολογίζεται το posterior vulnerability(πιθανότητα επιτυχίας αντιπάλου)

$V[\pi, C] = \sum_{y_j} \delta_{y_j} V(\sigma^{y_j})$ η οποία για Bayes vulnerability γίνεται:

$V[\pi, C] = \sum_{y_j} max_{x_i} \pi_{x_i} C_{x_i y_j}$ =0.25

Για το σύστημα 4:

- Με γνώση αντιπάλου ομοιόμορφη κατανομή $\pi=\{p(x_0), p(x_1), p(x_2)\}=$ {0.33333,0.33333,0.33333} . Εδώ έχουμε m=4, n=3 και c=1 οπότε έχουμε :

α=3/7

β=5/14

γ=3/28

Επομένως το κανάλι-πίνακας του συστήματος C έχει μορφή:

$$\begin{bmatrix} 5/14 & 3/28 & 3/28 & 3/7 \\ 3/28 & 5/14 & 3/28 & 3/7 \\ 3/28 & 3/28 & 5/14 & 3/7 \end{bmatrix}$$

και joint distribution περιέχει όλες τις πιθανότητες $p(y_j,x_i)=\pi_{x_i}C_{x_iy_j}$:

$$\begin{bmatrix} 5/42 & 1/28 & 1/28 & 1/7 \\ 1/28 & 5/42 & 1/28 & 1/7 \\ 1/28 & 1/28 & 5/42 & 1/7 \end{bmatrix}$$

και output distribution πC με $\delta_{y_j} = p(y_j) = \sum_{x_i} p(y_j, x_i)$:

$$\delta = [4/21 \quad 4/21 \quad 4/21 \quad 3/7]$$

και προκύπτουν οι posterior distributions [π,C] με στοιχεία $\sigma_{x_i}^{y_j} = p(x_i|y_j)$:

$$\begin{bmatrix} 5/8 & 3/16 & 3/16 & 1/3 \\ 3/16 & 5/8 & 3/16 & 1/3 \\ 3/16 & 3/16 & 5/8 & 1/3 \end{bmatrix}$$

Υπολογίζεται το posterior vulnerability(πιθανότητα επιτυχίας αντιπάλου)

$V[\pi, C] = \sum_{y_j} \delta_{y_j} V(\sigma^{y_j})$ η οποία για Bayes vulnerability γίνεται:

$V[\pi, C] = \sum_{y_j} max_{x_i} \pi_{x_i} C_{x_i y_j} = 3\frac{5*4}{8*21} + \frac{3}{21} = \frac{1}{2}$

Για το σύστημα 5:

- Με γνώση αντιπάλου ομοιόμορφη κατανομή $\pi=\{p(x_0), p(x_1), p(x_2)\}=$ {0.33333,0.33333,0.33333} και για πρωτόκολλο dc με δίκαια νομίσματα $cr=\{0,1\}=\{y_0, y_1\}$:$P(0)=P(1)=0.5$ προκύπτει ο πίνακας C με πιθανότητες $p(y_j|x_i)$:

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 \end{bmatrix}$$

και joint distribution περιέχει όλες τις πιθανότητες $p(y_j,x_i)=\pi_{x_i}C_{x_iy_j}$:

$$\begin{bmatrix} 1/27 & 1/27 & 1/27 & 1/27 & 1/27 & 1/27 & 1/27 & 1/27 \\ 1/27 & 1/27 & 1/27 & 1/27 & 1/27 & 1/27 & 1/27 & 1/27 \\ 1/27 & 1/27 & 1/27 & 1/27 & 1/27 & 1/27 & 1/27 & 1/27 \end{bmatrix}$$

και output distribution πC με $\delta_{y_j} = p(y_j) = \sum_{x_i} p(y_j, x_i)$:

$$\delta = [1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9]$$

και προκύπτουν οι posterior distributions [π,C] με στοιχεία $\sigma_{x_i}^{y_j} = p(x_i|y_j)$:

$$\begin{bmatrix} 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 & 1/3 \end{bmatrix}$$

Υπολογίζεται το posterior vulnerability:

$V[\pi, C] = \sum_{y_j} \delta_{y_j} V(\sigma^{y_j})$ η οποία για Bayes vulnerability γίνεται:

$V[\pi, C] = \sum_{y_j} max_{x_i} \pi_{x_i} C_{x_i y_j}$ $= \frac{1}{3}\frac{1}{9}9 = \frac{1}{3}$

Η θεωρητική ανάλυση εδώ δίνει τα ίδια αποτελέσματα με το σύστημα 1 δηλαδή ο αντίπαλος έχει την ίδια πιθανότητα επιτυχία στα δύο συστήματα, παρόλο που το σύστημα 5 έχει ένα χρήστη corrupted

- Με γνώση αντιπάλου biased κατανομή $\pi=\{p(x_0), p(x_1), p(x_2)\}=$

  {0.5,0.0,0.5} και για πρωτόκολλο dc με δίκαια νομίσματα $cr=\{0,1\}=\{y_0, y_1\}$:P(0)=P(1)=0.5 προκύπτει ο πίνακας C με πιθανότητες $p(y_j|x_i)$:

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 \end{bmatrix}$$

και joint distribution περιέχει όλες τις πιθανότητες $p(y_j,x_i)=\pi_{x_i}C_{x_iy_j}$:

$$\begin{bmatrix} 1/18 & 1/18 & 1/18 & 1/18 & 1/18 & 1/18 & 1/18 & 1/18 & 1/18 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/18 & 1/18 & 1/18 & 1/18 & 1/18 & 1/18 & 1/18 & 1/18 & 1/18 \end{bmatrix}$$

και output distribution πC με $\delta_{y_j} = p(y_j) = \sum_{x_i} \mathrm{p}(\mathrm{y}_j, x_i)$:

$$\delta = [1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9 \quad 1/9]$$

και προκύπτουν οι posterior distributions [π,C] με στοιχεία $\sigma_{x_i}^{y_j} = p(x_i|y_j)$:

$$\begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \end{bmatrix}$$

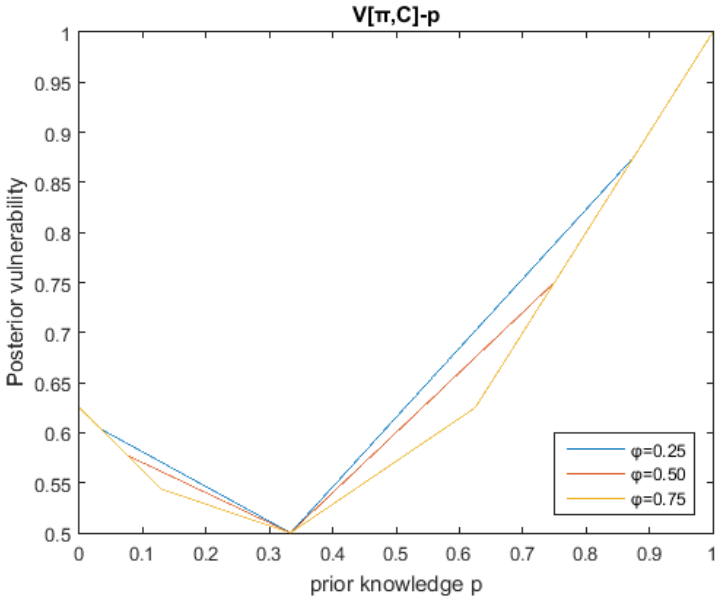
Υπολογίζεται το posterior vulnerability:

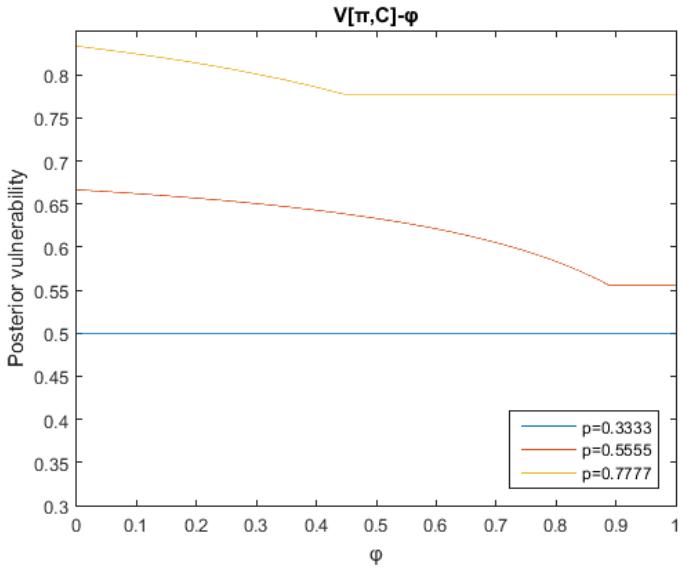
$V[\pi, C] = \sum_{y_j} \delta_{y_j} V(\sigma^{y_j})$ η οποία για Bayes vulnerability γίνεται:

$V[\pi, C] = \sum_{y_j} max_{x_i} \pi_{x_i} C_{x_i y_j} = \frac{1}{2}\frac{1}{9}9 = \frac{1}{2}$

Παρατηρούμε ότι για ίδιο σύστημα, η πιθανότητα επιτυχίας του αντιπάλου είναι μεγαλύτερη όταν η αρχική του γνώση είναι μια biased κατανομή από ότι όταν είναι ομοιόμορφη κατανομή.

Γραφικές παραστάσεις για σύστημα 4





Προγραμματιστικό Σκέλος

Όλα τα προγράμματα που περιγράφονται παρακάτω υλοποιήθηκαν με τη χρήση της γλώσσας προγραμματισμού java. Επίσης χρησιμοποιήθηκε μια εξωτερική βιβλιοθήκη της java(jars files) κυρίως για τις περιπτώσεις όπου πρέπει να γίνει επιλογή μιας ενέργειας με πιθανοτικό τρόπο. Η βιβλιοθήκη λέγεται: commons-math3-3.6.1 και είναι διαθέσιμη:

https://www-eu.apache.org/dist//commons/math/binaries/commons-math3-3.6.1-bin.zip

Το project περιέχει 3 main(μία για κάθε πρόγραμμα που πρέπει να υλοποιηθεί. Στον ίδιο φάκελο υπάρχουν:

- ο φάκελος files με τα αρχεία των γράφων συστημάτων που χρησιμοποιήθηκαν, των πιθανοτικών κατανομών αποστολής μηνυμάτων από τους χρήστες που αποτελούν την αρχική γνώστη του αντιπάλου, αρχείο με τους αποστολείς μηνυμάτων, αρχείο με πληροφορίες για κάθε προσομοίωση του πρωτοκόλου που είναι γνωστές για τον αντίπαλο, αρχείο με τις προβλέψεις του αντιπάλου για τους αποστολείς μηνυμάτων.
- Τα αρχεία jar της εξωτερικής βιβλιοθήκης
- Υπόλοιπες κλάσεις που χρησιμοποιούνται από τις main.

Simulation

Το πρώτο πρόγραμμα που υλοποιήθηκε, κάνει προσομοίωση ενός πρωτοκόλλου(dc ή crowds) σε ένα σύστημα-δίκτυο που αναπαρίσταται από γράφους της μορφής που είδαμε παραπάνω. Σε πρώτη φάση γίνεται αρχικοποίηση των παραμέτρων του προγράμματος. Οι περισσότεροι δίνονται σαν arguments μαζί με την εντολή εκτέλεσης στον command line:

- Είδος πρωτοκόλλου
- Όνομα αρχείου γράφου-δικτύου που θα χρησιμοποιηθεί στις προσομοιώσεις
- Αριθμός corrupted users
- Αριθμός προσομοιώσεων-δεν δίνεται ως argument στην εντολή εκτέλεσης στο command line αλλά προσδιορίζεται στο πρόγραμμα.
- Αρχείο με αποστολείς μηνυμάτων για τις προσομοιώσεις. Το αρχείο έχει τόσες γραμμές όσες ο αριθμός των προσομοιώσεων και κάθε γραμμή περιέχει το id-αριθμό αποστολέα μηνύματος. Για λόγους απλότητας έχει δημιουργηθεί μια συνάρτηση η οποία παράγει με τυχαίο τρόπο τόσους αποστολείς(ids αποστολέων) όσους και το πλήθος των προσομοιώσεων. Οι τιμές των ids που παράγονται προφανώς εξαρτώνται από των αριθμών χρηστών δικτύου(πλήθος κόμβων γράφου(userfile.txt)

Αφού προσδιοριστούν οι παραπάνω παράμετροι, διαβάζεται από το αντίστοιχο δοσμένο αρχείο ο γράφος-δίκτυο του συστήματος. Στη συνέχεια ελέγχεται αν η παράμετρος με το όνομα πρωτοκόλλου αντιστοιχεί σε κάποιο γνωστό πρωτόκολλο(εδώ dc ή crowds) και δημιουργεί ένα αντικείμενο της αντίστοιχης κλάσης. Σε διαφορετική περίπτωση τυπώνεται ένα μήνυμα λάθους. Οι κλάσεις DiningCryptographers και Crowds ουσιαστικά περιέχουν όλες τις μεθόδους εκείνες που προσομοιώνουν τον αλγόριθμο και τη διαδικασία του εκάστοτε πρωτοκόλλου πάνω στο γράφο-δίκτυο ενός δοσμένου αρχείου.

Πιο συγκεκριμένα για το πρωτόκολλο Dining Cryptographers, παράγονται τόσα δίκαια νομίσματα(50% 50% 1) όσες και οι ακμές του γράφου και στη συνέχεια για κάθε κόμβο οι τιμές των νομισμάτων που αντιστοιχούν στις ακμές που συνδέονται μαζί του περνάνε ανά ζεύγη σε πύλες xor και οι έξοδοι των πυλών αυτών πάλι περνάνε σαν είσοδοι ανά ζεύγη σε πύλες xor και η διαδικασία επαναλαμβάνεται μέχρι να βγει μία μοναδική έξοδος με τιμή 0 ή 1. Στο τέλος για κάθε κόμβο, υπάρχει μια τιμή εξόδου. Κόμβοι χωρίς συνδέσεις έχουν έξοδο 0. Για τον κόμβο αποστολέα η έξοδος του γίνεται xor με 1. Στη συνέχεια το πρόγραμμα παράγει ανάλογα με το πλήθος των corrupted users ένα αρχείο εξόδου(adversaryinfo.txt) που περιέχει πληροφορίες που είναι γνωστές στον αντίπαλο και χρησιμοποιούνται από αυτόν για

να προβλέψει τον αποστολέα του μηνύματος. Το αρχείο αυτό έχει τόσες γραμμές όσες και το πλήθος των προσομοιώσεων, Κάθε γραμμή αντιστοιχεί και σε μια εκτέλεση του πρωτοκόλλου .Κάθε γραμμή περιέχει m δυαδικές τιμές χωρισμένες με κενό μεταξύ τους(m πλήθος χρηστών) και στην συνέχεια περιέχει ένα αριθμό n δυαδικών χρηστών που αντιστοιχούν στα γνωστά νομίσματα στους corrupted users(ουσιαστικά ένα για κάθε σύνδεση των corrupted users). Προφανώς το n εξαρτάται από το συνολικό πλήθος ακμών του γράφου που συνδέονται στους corrupted users.

Από την άλλη στο Crowds, στο πρώτο βήμα ο αποστολέας προωθεί το μήνυμα σε κάποιο από τους κόμβους του γράφου με τους οποίους συνδέεται ή στον εαυτό του και κάθε επόμενος κόμβος επιλέγει με πιθανότητα 75% να προωθήσει το μήνυμα σε κάποιον άλλον κόμβο ή στον εαυτό του από αυτούς που συνδέεται ή σε διαφορετική περίπτωση(25%) να το στείλει στο server. Το μονοπάτι με τα id των χρηστών από τον αποστολέα στο server, αποθηκεύεται σε μια λίστα. Στη συνέχεια σαρώνεται αυτή η λίστα και εντοπίζουμε στο μονοπάτι τους corrupted users και ποιους έκαναν detected(προηγούμενο στοιχεία από τους corrupted στο μονοπάτι). Αυτές οι πληροφορίες αποθηκεύονται στο αρχείο εξόδου adversaryinfo.txt. Για κάθε εκτέλεση του πρωτοκόλλου υπάρχει μια γραμμή στο αρχείο . Κάθε γραμμή έχει την παρακάτω μορφή:

-1:honest_node1,…,honestn0 corrupted_node_id1:honest_node1,…,honestn1 corrupted_node_id2: honest_node1,…,honestn2… corrupted_node_idc: honest_node1,…,honestnc

Διευκρινήσεις:

- Με -1 συμβολίζεται ο server
- Κάθε γραμμή εκτέλεσης έχει το πολύ c+1 στοιχεία τα οποία χωρίζονται μεταξύ τους με ένα κενό
- Κάθε στοιχείο έχει δύο τμήματα τα οποία διαχωρίζονται με ‘’:’’. Στο πρώτο τμήμα έχουμε τα id του corrupted user ή -1 για το server και στο δεύτερο τα id των honest users χωρισμένα με ‘’,’’ που έχουν γίνει detected από τον συγκεκριμένο corrupted user.
- Παραδείγματα φαίνονται παρακάτω για crowd με m=4,n=3 και c=1:

```
-1:1 3:0,1,3,3,2
-1:3 3:1,1
-1:0
```

✓ Στην πρώτη γραμμή ο server έκανε detect τον 1 και ο 3 έκανε detect 2 φορές τον εαυτό του και από μία φορά τον 0 τον 1 και τον 2

✓ Στη δεύτερη γραμμή ο server κάνει detect τον 3 και ο 3 έκανε 2 φορές detect τον 1

✓ Στη τρίτη γραμμή ο server κάνει detect τον 0 . Ο corrupted user 3 δεν έκανε detect κάποιον χρήστη και για αυτό απουσιάζει.

Attack

Το δεύτερο πρόγραμμα που υλοποιήθηκε, μοντελοποεί ένα αντίπαλο οποίος χρησιμοποιεί τις πληροφορίες που συλλέγει από το σύστημα και την αρχική γνώση προσπαθεί να βρει τον αποστολέα του μηνύματος. Οι πληροφορίες που ξέρει ο αντίπαλος δίνονται σαν arguments στην εντολή εκτέλεσης του προγράμματος στο terminal:

- Είδος πρωτοκόλλου
- Όνομα αρχείου γράφου δικτύου πάνω στο οποίο θα γίνει η επίθεση
- Όνομα αρχείου που περιέχει μια κατανομή με n πιθανότητες. Κάθε i στοιχείο της κατανομής εκφράζει την πιθανότητα ο χρήστης i να είναι ο αποστολέας του μηνύματος
- Το όνομα του αρχείου εξόδου του simulation
- Επίσης σαν argument δίνεται ο αριθμός των corrupted users
- Ο αριθμός των προσομοιώσεων μπορεί να είναι ίδιος ή μικρότερος από ότι στο simulation και προσδιορίζεται πάλι στην αρχή του κώδικα

Αφού προσδιοριστούν οι παραπάνω παράμετροι, διαβάζεται πάλι από το αντίστοιχο δοσμένο αρχείο ο γράφος-δίκτυο του συστήματος. Στη συνέχεια ελέγχεται αν η παράμετρος με το όνομα πρωτοκόλλου αντιστοιχεί σε κάποιο γνωστό πρωτόκολλο(εδώ dc ή crowds). Σε περίπτωση μη αντιστοιχίας με κάποιο γνωστό πρωτόκολλο εμφανίζεται μήνυμα λάθους και το πρόγραμμα τερματίζει.

Στη συνέχεια δημιουργείται στο πρόγραμμα ένα αντικείμενο τύπου Attacker ο οποίος ανάλογα με το είδος του πρωτοκόλλου εκτελεί την αντίστοιχη μέθοδο επίθεσης. Το πρώτο πράγμα που κάνουν και οι δύο μέθοδοι είναι να διαβάσουν το αρχείο που παράχθηκε σαν έξοδο από το simulation program να επεξεργαστούν κατάλληλα το περιεχόμενο του και να το αποθηκεύσουν σε κατάλληλες δομές δεδομένων . Στη συνέχεια, κάθε μέθοδος ακολουθεί τη στρατηγική που θεωρήθηκε η βέλτιστη για το κάθε πρωτόκολλο για τον εντοπισμό του αποστολέα του μηνύματος.

Στο Dining Cryptographers, που είναι γνωστό ότι προσφέρει ισχυρή ανωνυμία, θεωρήθηκε ότι δεν θα είχε μεγάλο νόημα να υλοποιηθεί μια σύνθετη στρατηγική η οποία θα βασίζεται σε υπολογισμούς πιθανοτήτων σαν αυτές που είδαμε παραπάνω προκειμένου να επιλέξει τον έντιμο χρήστη που έχει τη μεγαλύτερη πιθανότητα να στείλει. Αντίθετα η στρατηγική που υλοποιήθηκε, δουλεύει όχι με μαθηματικό αλλά με εμπειρικό τρόπο. Αρχικά ελέγχεται αν ισχύει κάποια από τις παρακάτω ειδικές περιπτώσεις που επιτρέπουν τον σίγουρο και άμεσο εντοπισμό του αποστολέα του μηνύματος:

1. Αποστολέας του μηνύματος είναι ένας corrupted χρήστης.
2. Ένας απομονωμένος κόμβος δηλαδή ένας κόμβος που δεν συνδέεται με κάποιον άλλο και δίνει σαν έξοδο 1, είναι προφανές ότι αυτός είναι ο αποστολέας γιατί αν δεν ήταν , θα έδινε κανονικά σαν έξοδο 0(δεν έχει νομίσματα) αλλά αυτός δίνει έξοδο 1. Άρα ισχύει (output xor message)=(0 xor 1)=1
3. Ο αποστολέας του μηνύματος είναι ένας χρήστης κόμβος που συνδέεται αποκλειστικά μόνο με corrupted users. Σε τέτοιες περιπτώσεις ο αντίπαλος ξέρει από το αρχείο adversary.txt το output και τα νομίσματα του συγκεκριμένου κόμβου και από τη δομή του δικτύου επιβεβαιώνει ότι δεν υπάρχουν άλλα νομίσματα για το συγκεκριμένο κόμβο. Επομένως

παίρνοντας τα νομίσματα σε ζεύγη και κάνοντας τους xor προκύπτουν κάποιοί έξοδοι στους οποίους επίσης παίρνουμε ανά ζεύγη και τους κάνουμε xor και η διαδικασία αυτή συνεχίζεται μέχρι να προκύψει μία μοναδική έξοδος η οποία αν έχει τιμή που ταυτίζεται με την έξοδο του αντίστοιχου κόμβου από το αρχείο adversaryinfo.txt σημαίνει ότι ο κόμβος αυτός δεν είναι αποστολέας. Σε διαφορετική περίπτωση είναι ο αποστολέας.

Αν δεν ισχύει καμία από τις παραπάνω περιπτώσεις, τότε ο αντίπαλος κοιτάει στην αρχική του γνώση ποιος έντιμος χρήστης στέλνει με μεγαλύτερη συχνότητα μήνυμα(είναι πιο πιθανό να στείλει μήνυμα) και λέει ότι αυτός είναι ο αποστολέας. Σε περίπτωση που 2 ή περισσότεροι χρήστες έχουν την ίδια πιθανότητα να στείλουν μήνυμα τότε επιλέγεται κάποιος από αυτούς τυχαία.

Σε αντίθεση με το Dining Cryptographers στο Crowds ακολουθείται μια πιο σύνθετη πολιτική πρόβλεψης του πιο πιθανού αποστολέα του μηνύματος .Πρώτα ελέγχονται αν ισχύουν οι παρακάτω ειδικές περιπτώσεις(παρόμοιες με τις περιπτώσεις 1,2 στο dc):

1. Αποστολέας του μηνύματος είναι κάποιος corrupted users
2. Αποστολέας είναι κάποιος απομονωμένος κόμβος τότε αυτός μην έχοντας άλλη επιλογή θα προωθεί το μήνυμα συνεχώς στον εαυτό του μέχρι κάποια στιγμή να σταλεί στο server.Ο αντίπαλος βλέπει ότι έγινε ο συγκεκριμένος χρήστης detect από το server και ξέρει από το γράφο του δικτύου ότι ο συγκεκριμένος χρήστης δεν συνδέεται με κάποιον άλλο κόμβο του δικτύου και επομένως δεν υπάρχει πιθανότητα να προωθεί το μήνυμα κάποιου άλλου χρήστη-αποστολέα .Συνεπώς αυτός είναι ο αποστολέας

Αν δεν ισχύει καμιά από τις 2 παραπάνω περιπτώσεις, τότε ελέγχεται αν έχει γίνει detect κάποιος άλλος χρήστης εκτός από αυτόν που έκανε detect ο server. Αν δεν έχει γίνει τότε ακολουθείται η ίδια στρατηγική με αυτή που περιγράφεται στο dc, δηλαδή θεωρείται αποστολέας του μηνύματος ο έντιμος χρήστης που με βάση την κατανομή έχει μεγαλύτερη πιθανότητα να στείλει. Αν με βάση την κατανομή, υπάρχουν πολλοί χρήστες που έχουν την ίδια μέγιστη πιθανότητα αποστολής τότε επιλέγεται τυχαία ένας από αυτούς. Σε διαφορετική περίπτωση γίνεται η παρακάτω μαθηματική ανάλυση που οδηγεί στο χρήστη που έχει τη μεγαλύτερη πιθανότητα να είναι αποστολέας του μηνύματος με βάση γνώσης την πληροφορία ότι ένα συγκεκριμένο σύνολο έντιμων χρηστών που έγιναν detected. Αρχικά υπολογίζονται οι πιθανότητες α, β, γ που είδαμε και παραπάνω τι ακριβώς αντιπροσωπεύουν . Στη συνέχεια υπολογίζονται οι πιθανότητες $p(det_i)$, όπου i:id ενός έντιμου χρήστη(στο θεωρητικό κομμάτι οι τιμές αυτές δίνονται από την output distribution). Οι πιθανότητες αυτές υπολογίζονται με τη χρήση του θεωρήματος ολικής πιθανότητας:

$$p(det_i) = p(det_i|user_i)p(user_i) + \sum_{j=0, j\neq i}^{n} p(det_i|user_j)p(user_j) = \beta\, p(user_i)$$
$$+ \gamma \sum_{j=0, j\neq i}^{n} p(user_j) = \beta p(user_i) + \gamma\left(\sum_{j=0}^{n} p(user_j) - p(user_i)\right)$$
$$= \beta p(user_i) + \gamma\big(1 - p(user_i)\big), j: \textit{επίσης id honest user}$$

Στη συνέχεια υπολογίζονται με εφαρμογή του κανόνα Bayes οι παρακάτω πιθανότητες(posterior distribution στη θεωρία):

$$p(user_i|det_i) = \frac{p(det_i|user_i)p(user_i)}{p(det_i)} = \frac{\beta p(user_i)}{p(det_i)}$$

$$p(user_i|det_j) = \frac{p(det_j|user_i)p(user_i)}{p(det_j)} = \frac{\gamma p(user_i)}{p(det_j)}$$

Στο σημείο αυτό πρέπει να γίνουν 2 σημαντικές παρατηρήσεις:

- Υπολογίζονται μόνο οι πιθανότητες εκείνες για τις οποίες οι τιμές που παίρνουν οι τυχαίες μεταβλητές det_i , det_j στους προηγούμενους τύπους είναι id κόμβων που έχουν όντως γίνει detect κατά την συγκεκριμένη αποστολή του συγκεκριμένου μηνύματος. Με άλλα λόγια δεν υπολογίζουμε όλες τις πιθανότητες του πίνακα του posterior distribution αλλά μόνο εκείνες τις στήλες του που αφορούν κόμβους που έχουν όντως γίνει detected κατά την εκτέλεση του προγράμματος
- Υπάρχουν και οι πιθανότητες $p(\perp|user_i)=p(user_i)$ αφού $p(\perp)=\alpha$

Θεωρώντας ότι οι δεσμευμένες πιθανότητες σε κάθε γραμμή των μειωμένων διαστάσεων πίνακα posterior distribution, είναι ανεξάρτητες μεταξύ τους , παίρνουμε την τομή τους, πολλαπλασιάζοντας τις πιθανότητες κάθε γραμμής και έτσι προκύπτουν n πιθανότητες μία για κάθε έντιμο χρήστη. Τέλος αυτές οι πιθανότητες πολλαπλασιάζονται με τις αντίστοιχες $p(\perp|user_i)$. Ο αποστολέας του μηνύματος είναι αυτός που έχει τη μεγαλύτερη πιθανότητα από αυτές.

Τέλος για κάθε προσομοίωση, ο αποστολέας που προβλέπει ο αντίπαλος καταγράφεται σαν μια νέα γραμμή στο αρχείο guess.txt.

Vulnerability

Το πρόγραμμα αυτό ουσιαστικά είναι συνδυασμός των 2 προηγουμένων με κάποιες τροποποιήσεις. Πρέπει να δοθούν σαν arguments στην εντολή εκτέλεσης του προγράμματος από το command line τα παρακάτω:

- Είδος πρωτοκόλλου
- Όνομα αρχείου γράφου δικτύου πάνω στο οποίο θα γίνει η επίθεση
- Αριθμός corrupted users
- Όνομα αρχείου που περιέχει μια κατανομή με n πιθανότητες. Κάθε i στοιχείο της κατανομής εκφράζει την πιθανότητα ο χρήστης i να είναι ο αποστολέας του μηνύματος
- Αριθμός προσομοιώσεων

Διαβάζεται το αρχείο με το γράφο δίκτυο όπως προηγουμένως όμως δεν παράγονται τυχαία αποστολείς μηνυμάτων αλλά με βάση την κατανομή της αρχικής γνώσης του αντιπάλου. Συνεπώς στο πρόγραμμα αυτό σε αντίθεση με αυτό του simulation δεν θα έχουμε σαν αποστολείς μηνυμάτων corrupted users.Στη συνέχεια γίνεται προσομοίωση του πρωτοκόλλου αν είναι ένα από τα 2 γνωστά με τον ίδιο τρόπο όπως στο πρόγραμμα simulation.Στη συνέχεια δημιουργείται ένας Attacker που κάνει τόσες επιθέσεις στο σύστημα με το πρωτόκολλο ανώνυμης επικοινωνίας όσες και το πλήθος των προσομοιώσεων. Στη συνέχεια γίνεται σύγκριση των τιμών της λίστας με τους πραγματικούς

αποστολείς με τις αντίστοιχες τιμές της λίστας με τους αποστολείς που προέβλεψε ο αντίπαλος και στο τέλος καταγράφεται το ποσοστό επιτυχίας του αντιπάλου.

Αποτελέσματα

Τρέξαμε τα προγράμματα που περιέγραφηκαν για όλα τα συστήματα για τα οποία κάναμε σε προηγούμενη ενότητα θεωρητική ανάλυση. Στον παρακάτω πίνακα φαίνονται τα αποτελέσματα με τα θεωρητικά και πειραματικά ποσοστά επιτυχίας του αντιπάλου(posterior Bayes vulnerability) για κάθε μια από τις 6 περιπτώσεις που παρουσιάστηκαν παρακάτω:

Αριθμός προσομοιώσεων:10000

Πρωτόκολλο	Αριθμός users	Αριθμός corrupted users	κατανομή αρχικής γνώσης	Θεωρητικό ποσοστό επιτυχίας αντιπάλου	Πειραματικό Ποσοστό επιτυχίας αντιπάλου
dc	3	0	uniform	0.3333	~0.3333
dc	3	1	biased->0.6,0.4	0.6000	~0.6000
crowds	4	0	uniform	0.2500	~0.2500
crowds	4	1	uniform	0.5000	~0.4400
dc	4	1	uniform	0.3333	~0.5555
dc	4	1	biased ->0.5,0,0.5	0.5000	~0.7500

Σχολιασμός Αποτελεσμάτων

- Τα πειραματικά ποσοστά επιτυχίας του αντιπάλου είναι προσεγγιστικά γιατί σε κάθε εκτέλεση του προγράμματος παράγονται οι αποστολείς το μηνυμάτων με βάση την αρχική γνώση του αντιπάλου- επομένως δεν έχουμε πάντα την ίδια αλληλουχία αποστολέων ούτε το πλήθος των μηνυμάτων που στέλνει ο κάθε έντιμος χρήστης είναι απόλυτα σταθερό.
- Επικεντρωνόμαστε στις περιπτώσεις όπου παρατηρούνται αποκλίσεις ανάμεσα στα θεωρητικά και πειραματικά ποσοστά επιτυχίας του αντιπάλου.
- Είναι γεγονός ότι κατά τον θεωρητικό υπολογισμό του θεωρητικού ποσοστού επιτυχίας του αντιπάλου, γίνονται μόνο υπολογισμοί πιθανοτήτων και δεν εξετάζονται παράμετροι όπως το σχήμα του γράφου και ο τρόπος με τον οποίο κατανέμονται σε αυτόν οι honest και οι corrupted users που ακριβώς

είναι οι corrupted users και με πόσους και ποιους έντιμους συνδέονται. Τέτοια παραδείγματα είναι αυτά που είδαμε με τις προσομοιώσεις του συστήματος 5 με τους 4 χρήστες συνδεδεμένους σε σειρά και τον προτελευταίο χρήστη στη σειρά να είναι corrupted. Παρατηρήθηκε ότι για πρωτόκολλο dining cryptographers με m=4,n=3 και c=1, τα θεωρητικά και πειραματικά ποσοστά έχουν μεγάλη απόκλιση. Αυτό εξηγείται παρακάτω:

- ***<u>Για αρχική γνώση του αντιπάλου ομοιόμορφη κατανομή</u>***: Με βάση το σχήμα του γράφου και τη θέση του corrupted user 3 και την στρατηγική του προγράμματος επίθεσης για dc πρωτόκολλο, είναι δεδομένο ότι σε κάθε περίπτωση ο αντίπαλος θα βρίσκει τον αποστολέα όταν στέλνει ο 2. Αφού η κατανομή είναι ομοιόμορφη ο 2 όπως και κάθε άλλος έντιμος χρήστης έχει πιθανότητα 33.33% να στείλει, οπότε το ποσοστό επιτυχίας του αντιπάλου είναι σίγουρα τόσο. Επιπλέον ο αντίπαλος πρέπει να κάνει για το υπόλοιπο 66.66% τον περιπτώσεων μια πρόβλεψη και αφού η κατανομή είναι ομοιόμορφη επιλέγει ένα χρήστη τυχαία(πιθανότητα 33.33%) οπότε προκύπτει 0.66*0.33=0.22 και έτσι δικαιολογείται τον 55.55% σαν ποσοστό επιτυχίας του αντιπάλου έναντι του θεωρητικού 33.33%
- ***<u>Για αρχική γνώση του αντιπάλου biased κατανομή με [0.5 0 0.5]</u>***: Αντίστοιχα όταν η αρχική γνώση του αντιπάλου για το ίδιο σύστημα με προηγουμένως είναι μια biased κατανομή, προκύπτει πάλι ότι αντίπαλος θα κάνει σωστή πρόβλεψη όταν στέλνει ο 2 (πιθανότητα 50%) και στις άλλες περιπτώσεις επιλέγει τυχαία τον χρήστη με την μεγαλύτερη πιθανότητα αποστολή δηλαδή τον 0 ή τον 2(πιθανότητα 0.5*0.5=0.25).Άρα το ποσοστό επιτυχίας του αντιπάλου είναι περίπου 75%.

- Τέλος για το σύστημα 4 με τον πλήρη γράφο 4 χρηστών με 1 corrupted χρήστη για το πρωτόκολλο crowds, το πειραματικό ποσοστό επιτυχίας του αντιπάλου είναι λίγο μικρότερο από το θεωρητικό (44% έναντι 50%) αλλά θα πρέπει να υπενθυμίσουμε ότι στο πρόγραμμα ο αντίπαλος αποφασίζει με βάση τους κόμβους που έχει όντως εντοπίσει σε μια προσομοίωση και όχι με βάση αυτούς που θεωρητικά θα μπορούσε να έχει εντοπίσει. Για την πρόβλεψη δεν χρησιμοποιείται στο πρόγραμμα όλος ο πίνακας posterior distribution(χρησιμοποιείται στον υπολογισμό του posterior bayes vulnerability) όπως για το θεωρητικό αλλά μόνο εκείνες οι στήλες του που είναι για του χρήστες που έγιναν όντως detect από τον Attacker.

Κώδικας

Ο κώδικας είναι διαθέσιμος στο παρακάτω link:

https://www.dropbox.com/s/5iav9dl99gofe68/PrivacyTechniquesProject.zip?dl=0

Παρακάτω ακολουθούν οδηγίες για την σωστή εκτέλεση των προγραμμάτων:

1. Compile όλα τα αρχεία .java:

```
javac ProtocolType.java

javac Protocol.java

javac PriorKnowledge.java

javac Action.java

javac DiningCryptographers.java

javac Estimator.java

javac FileOperation.java

javac NetworkGraph.java

javac Attacker.java

javac -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Crowds.java

javac -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar User_execution.java

javac -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Simulation.java

javac -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Attack.java

javac -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Vulnerability.java
```

Ουσιαστικά κάνουμε compile κάθε αρχείο κώδικα με την εντολή javac <namefile.java> και για το compile των 5 τελευταίων αρχείων πρέπει να προσδιοριστούν στο classpath τα external jar files με javac –cp.:<jarfile1.jar>:<jarfile2.jar>:…:<jarfilen.jar> <namefile.java>.

2. Για την εκτέλεση των προγραμμάτων:

java cp.:<jarfile1.jar>:<jarfile2.jar>:…:<jarfilen.jar> <namefile> <arguments>

Για το Simulation:

java -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Simulation “protocol name” “graphfilename” corrupted_users_number “senderfilename”

Παράδειγμα:

java -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Simulation "crowds" "graph-4-complete" 1 "userfile.txt"

Για το Attack:

java -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Attack Simulation “protocol name” “graphfilename” corrupted_users_number “prior_knowledgefilename” “simulation_output”

Παράδειγμα:

java -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Attack "crowds" "graph-4-complete" 1 "prior-3-uniform" "adversaryinfo.txt"

Για το Vulnerability:

java -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Vulnerability “protocol name”

“graphfilename” corrupted_users_number “prior_knowledgefilename” simulation_number

java -cp .:commons-math3-3.6.1.jar:commons-math3-3.6.1-javadoc.jar:commons-math3-3.6.1-sources.jar:commons-math3-3.6.1-tests.jar:commons-math3-3.6.1-test-sources.jar:commons-math3-3.6.1-tools.jar Vulnerability "crowds" "graph-4-complete" 1 "prior-3-uniform" 10000