Privacy Technologies Project Report

Στέφου Θεόδωρος - ΠΜΣ Πληροφορική 2
ΑΜ: CS2190002
e-mail: cs2190002@di.uoa.gr / theo.stefou@gmail.com

Σε αυτό το project report θα εξηγηθούν τεχνικές επιλογές για το project και θα αναλυθούν αποτελέσματα ανάλυσης της εξόδου από το εργαλείο fbleau.

Ο κώδικας είναι διαθέσιμος σε αυτό το github repository:
https://github.com/TheoStefou/crowds-protocol-simulation

ΤΟ ΠΡΩΤΟΚΟΛΛΟ CROWDS

Το πρωτόκολλο CROWDS είναι ένα πιθανοτικό πρωτόκολλο ανωνυμίας το οποίο προσφέρει αρκετά ισχυρή ανωνυμία (και ικανοποιητική απόδοση σε αντίθεση με άλλα πρωτόκολλα όπως το dining cryptographers το οποίο προσφέρει ισχυρή αντωνυμία αλλά η απόδοσή του αποτρέπει την εκτέλεσή του σε ρεαλιστικά σενάρια) στους χρήστες ως προς τον τελικό προορισμό του μηνύματος υπό τη συνθήκη ότι ο αντίπαλος δεν ελέγχει κάποιον άλλο κόμβο-χρήστη. Σε περίπτωση που κάποιος από τους χρήστες ελέγχεται από τον αντίπαλο, τότε το πρωτόκολλο προσφέρει στους χρήστες το λεγόμενο probable innocence, δηλαδή τη δυνατότητα να ισχυριστούν με κάποια πιθανότητα ότι δεν ήταν οι ίδιοι ο αρχικός αποστολέας, αλλά ότι απλά προωθούσαν το μήνυμα μέσα στο δίκτυο. Προφανώς το πρωτόκολλο δεν μπορεί να προσφέρει κανενός είδους ανωνυμία σε κάποιον χρήστη που ελέγχεται από τον αντίπαλο. Από αυτό το σημείο και μέχρι το τέλος αυτής της αναφοράς, θα αναφερόμαστε σε χρήστες που ελέγχονται από τον αντίπαλο ως corrupt χρήστες ή το σύνολο αυτών ως “αντίπαλο”. Όλες οι επικοινωνίες κατά τη διάρκεια του πρωτοκόλλου θεωρούνται ισχυρά κρυπτογραφημένες και η αναγνώριση χρηστών από το περιεχόμενο των μηνυμάτων δεν αποτελεί ποτέ πρόβλημα - μας ενδιαφέρει μόνο η ανωνυμία του αποστολέα ως προς τους υπόλοιπους συμμετέχοντες.

Η λειτουργία του πρωτοκόλλου έχει ως εξής:
Έστω ο μη κατευθυνόμενος γράφος G(V,E) που περιγράφει το δίκτυο χρηστών που συμμετέχουν στο πρωτόκολλο. Οι κόμβοι του γράφου αντιστοιχούν σε χρήστες, ενώ οι ακμές αντιστοιχούν σε αμφίδρομους διαύλους επικοινωνίας μεταξύ ζευγαριών χρηστών. Το πρωτόκολλο χαρακτηρίζεται από μία δεκαδική παράμετρο $\varphi \in [0,1)$ η οποία ορίζει την πιθανότητα προώθησης. Όταν ένας χρήστης επιθυμεί να στείλει ένα μήνυμα σε κάποιο τελικό προορισμό-server, θα πρέπει ξεκινήσει μία διαδικασία η οποία θα δημιουργήσει μία αλυσίδα προωθήσεων. Σύμφωνα με το πρωτόκολλο, ο πρώτος χρήστης-initiator προωθεί με πιθανότητα 100% το μήνυμα διαλέγοντας ομοιόμορφα ανάμεσα στους χρήστες με τους οποίους είναι συνδεδεμένος συμπεριλαμβανομένου και του εαυτού του. Από εκεί και πέρα, κάθε επόμενος χρήστης προωθεί με τον ίδιο τρόπο με πιθανότητα φ το μήνυμα σε δικούς του γείτονες (συμπεριλαμβανομένου και του εαυτού του ομοιόμορφα) και με πιθανότητα 1-φ παραδίδει το μήνυμα στον προορισμό.

ΤΕΧΝΙΚΕΣ ΕΠΙΛΟΓΕΣ

Η προσομοίωση του πρωτοκόλλου για το project αυτό έγινε στη γλώσσα Python, έκδοση 3.5.2 και δεν έχει χρησιμοποιηθεί κάποια version specific βιβλιοθήκη ή προγραμματιστική σύνταξη, οπότε μπορεί να εκτελεστεί με οποιαδήποτε έκδοση της Python 3 (Δεν έχει επιβεβαιωθεί πειραματικά). Επίσης, χρησιμοποιήθηκε λειτουργικό Linux Ubuntu 16.04 LTS.

Για την ανάλυση της εξόδου των προσομοιώσεων χρησιμοποιήθηκε το εργαλείο fbleau του οποίου η εγκατάσταση έγινε με την εντολή: `cargo install fbleau`
Χρησιμοποιήθηκε δηλαδή ως linux utility και όχι μέσω του python module που διατίθεται.

ΕΠΕΞΗΓΗΣΗ ΑΡΧΕΙΩΝ ΚΩΔΙΚΑ

Στο github repository του project αυτού βρίσκονται τα εξής python scripts:

1. **simulate.py**
   Περιλαμβάνει την ανάλυση-parsing των παραμέτρων του συστήματος και την εκτέλεση αυτού. Μπορεί να εκτελεστεί ως εξής:
   `python3 simulate.py <phi> <graph-file> <corrupted-file> <users-file> <broken-paths> <fix-strategy>`
   - **phi**: Η παράμετρος πιθανότητας προώθησης φ
   - **graph-file**: Μονοπάτι αρχείου με τον πίνακα γειτνίασης που περιγράφει τον γράφο. Ο πίνακας είναι συμμετρικός ως προς τη διαγώνιο και παρά το ότι ένας χρήστης μπορεί να προωθήσει στον εαυτό του, τα στοιχεία της διαγωνίου είναι μηδενικά. Ο γράφος δεν είναι βεβαρημένος, άρα τα υπόλοιπα στοιχεία μπορούν να είναι 1 ή 0.
     Παράδειγμα:
     0 1 0
     1 0 1
     0 1 0
   - **corrupted-file**: Μονοπάτι αρχείου με τα ids των χρηστών που είναι corrupt (ένα id ανά γραμμή, όχι διπλότυπα)
   - **users-file:** Μονοπάτι αρχείου με τα ids των χρηστών που έχουν να στείλουν κάποιο μήνυμα. Δεν μπορεί να περιέχει id corrupt χρήστη. (ένα id ανά γραμμή, κάθε χρήστης μπορεί να στείλει πολλά μηνύματα, η προσομοίωση γίνεται με τη σειρά που δίνεται στο αρχείο, η κατανομή αποστολέων που υπονοείται στο αρχείο αυτό αποτελεί την αρχική γνώση του αντιπάλου)
   - **broken-paths**: Θετικός ακέραιος που ορίζει πόσες φορές ο αντίπαλος μπορεί να σπάσει ένα μονοπάτι κατά τη διάρκεια μίας προσομοίωσης αποστολής. Όταν ο αντίπαλος “σπάει” ένα μονοπάτι, ουσιαστικά εννοούμε ότι καταγράφει τον εντοπισμό του χρήστη που του προώθησε το μήνυμα και δεν κάνει τίποτα για αυτό, άρα το πρωτόκολλο υπο ρεαλιστικές συνθήκες “κολλάει” και πρέπει να επανεκτελεστεί.
   - **fix-strategy**: Μπορεί να είναι “initiator” ή “last-honest”. Ορίζει την στρατηγική επανεκτέλεσης όταν ένας αντίπαλος σπάει ένα μονοπάτι. Κατά την στρατηγική “initiator”, το πρωτόκολλο επανεκτελείται από την αρχή από τον ίδιο τον αποστολέα, ενώ κατά την στρατηγική “last-honest”, ο τελευταίος μη corrupt χρήστης αναλαμβάνει την επανεκτέλεση από το

σημείο που είχε μείνει. Δε λαμβάνεται υπ' όψιν για broken-paths=0 αλλά πρέπει να δοθεί παρ' όλα αυτά.

2. **crowds.py**
   Κάνει expose την κλάση CrowdsSimulation η οποία περιέχει μεθόδους για την εκτέλεση του πρωτοκόλλου. Το μικρό αυτό "API", διαθέτει στον χρήστη του έναν constructor για την αρχικοποίηση και ορισμό παραμέτρων του πρωτοκόλλου, καθώς και μία μέθοδο simulate για την προσομοίωση μίας εκτέλεσης ξεκινώντας από έναν συγκεκριμένο χρήστη του οποίου το id λαμβάνει ως όρισμα. Οι υπόλοιπες μέθοδοι χρησιμοποιούνται εσωτερικά για καλύτερη οργάνωση του κώδικα.
   Η έξοδος για κάθε κλήση της μεθόδου simulate είναι ένα παρατηρήσιμο γεγονός από τον αντίπαλο. Κάθε γραμμή αντιστοιχεί σε μία εκτέλεση και περιέχει τα ids των χρηστών που έγιναν detect από τον αντίπαλο χωρισμένα με έναν κενό χαρακτήρα.

3. **generate_users_file.py**
   Μπορεί να εκτελεστεί ως εξής:
   `python3 <apriori-file> <num-iterations>`
   Αναλαμβάνει τη δημιουργία του users-file.
   - **apriori-file**: Περιέχει την κατανομή συχνότητας αποστολής για κάθε χρήστη. Αντιπροσωπεύει την κατανομή προϋπάρχουσας γνώσης του αντιπάλου. (ένας δεκαδικός ανά γραμμή, αθροίζουν στο 1 με tolerance ανακρίβειας 10^{-5} για αποφυγή θεμάτων ακρίβειας αναπαράστασης δεκαδικών αριθμών στο δυαδικό)
   - **num-iterations**: Θετικός ακέραιος που ορίζει τον συνολικό αριθμό αποστολών που πρέπει να παραχθούν από όλους τους χρήστες.

4. **distribution.py**
   Μπορεί να εκτελεστεί ως εξής:
   `python3 <num-users>`
   Βοηθητικό script για την αυτόματη παραγωγή ομοιόμορφης κατανομής **num-users** αποστολέων. Οι corrupt χρήστες θα πρέπει να έχουν πιθανότητα αποστολής ίση με το μηδέν, κάτι που δεν αναλαμβάνει αυτό το script, οπότε σε περίπτωση που χρησιμοποιηθεί θα πρέπει να γίνουν αλλαγές χειροκίνητα.

5. **map_observables.py**
   Μπορεί να εκτελεστεί ως εξής:
   python3 <simulation-out> <users-file> <fbleau-input>
   Χρησιμοποιείται για την αντιστοίχιση της εξόδου της προσομοίωσης σε αναγνωριστικά παρατηρήσιμων. Δημιουργεί ένα αρχείο csv με δύο κολώνες, όπου η πρώτη είναι ο χρήστης που έκανε την αποστολή, ενώ η δεύτερη είναι το αναγνωριστικό του αντίστοιχου παρατηρήσιμου για την εκτέλεση. Αν η έξοδος του simulate.py ήταν ως εξής (χωρίς τα βελάκια και τον αριθμό δεξιά τους):
   0 1 1 2 -> 0
   1 2 3 4 -> 1
   0 1 1 2 -> 0
   4 4 -> 2
   4 4 -> 2
   τότε το script αυτό θα αντιστοίχιζε τα παρατηρήσιμα με τα αναγνωριστικά που φαίνονται μετά τα βελάκια. Αυτό γίνεται καθώς το fbleau περιμένει είσοδο με δύο κολώνες. Σημαντικό να αναφερθεί ότι η σειρά των detections παίζει ρόλο και μπορεί να διαχωρίσει δύο παρατηρήσιμα ακόμα και αν ως σύνολα έχουν τα ίδια στοιχεία.
   - **simulation-out**: Μονοπάτι αρχείου που περιέχει την έξοδο εκτέλεσης της προσομοίωσης από το script simulate.py
   - **users-file**: Μονοπάτι προς το αρχείου αποστολέων. Το ίδιο που χρησιμοιήθηκε για το simulation.py
   - **fbleau-input**: Μονοπάτι προς το αρχείο csv που θα αποθηκευτούν τα αποτελέσματα.

6. **generate_fbleau_input.py**
   Μπορεί να εκτελεστεί ως εξής:
   python3 generate_fbleau_input.py <mapped_csv> <train_csv> <test_csv>
   Σκοπός του script αυτού είναι να χωρίσει τα mapped observables που δημιούργησε το script map_observables.py σε train set και test set ομοιόμορφα. Για κάθε μία γραμμή των mapped observables, την γράφει στο train set με 80% πιθανότητα και στο test set με πιθανότητα 20%. Έτσι χωρίζει ομοιόμορφα την έξοδο και την ετοιμάζει για το classification training που θα κάνει το εργαλείο fbleau.
   - **mapped_csv**: Μονοπάτι csv αρχείου που έχει παραχθεί από το script map_observables.py
   - **train_csv**: Μονοπάτι προς το csv αρχείο όπου θα γραφτεί το train set.
   - **test_csv**: Μονοπάτι προς το csv αρχείο όπου θα γραφτεί το test set.

7. **execute-all.py**
   Μπορεί να εκτελεστεί ως εξής:
   python3 execute-all.py
   Το script αυτό χρησιμοποιήθηκε για την αυτόματη εκτέλεση του συστήματος για διαφορετικές παραμέτρους, καθώς και για την αυτοματοποίηση των ενδιάμεσων βημάτων που περιγράφονται στο 5 και 6. Συγκεκριμένα, αναλαμβάνει την ολοκληρωμένη εκτέλεση και αποθήκευση αποτελεσμάτων για κάθε διαφορετικό συνδυασμό παραμέτρων από τις παρακάτω:
   - Αν ο γράφος περιέχει corrupt users (2 επιλογές)
   - Αν η αρχική κατανομή είναι ομοιόμορφη ή όχι (2 επιλογές)
   - Τη στρατηγική επιδιόρθωσης σπασμένου μονοπατιού (2 επιλογές)
   - Την παράμετρο φ για τις τιμές
     [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.92, 0.94, 0.96, 0.98, 0.99]
     (15 επιλογές)

   Οπότε η προσομοίωση εκτελέστηκε για 2*2*2*15=120 διαφορετικούς συνδυασμούς.

ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Για τις ακόλουθες μετρήσεις, χρησιμοποιήθηκαν:

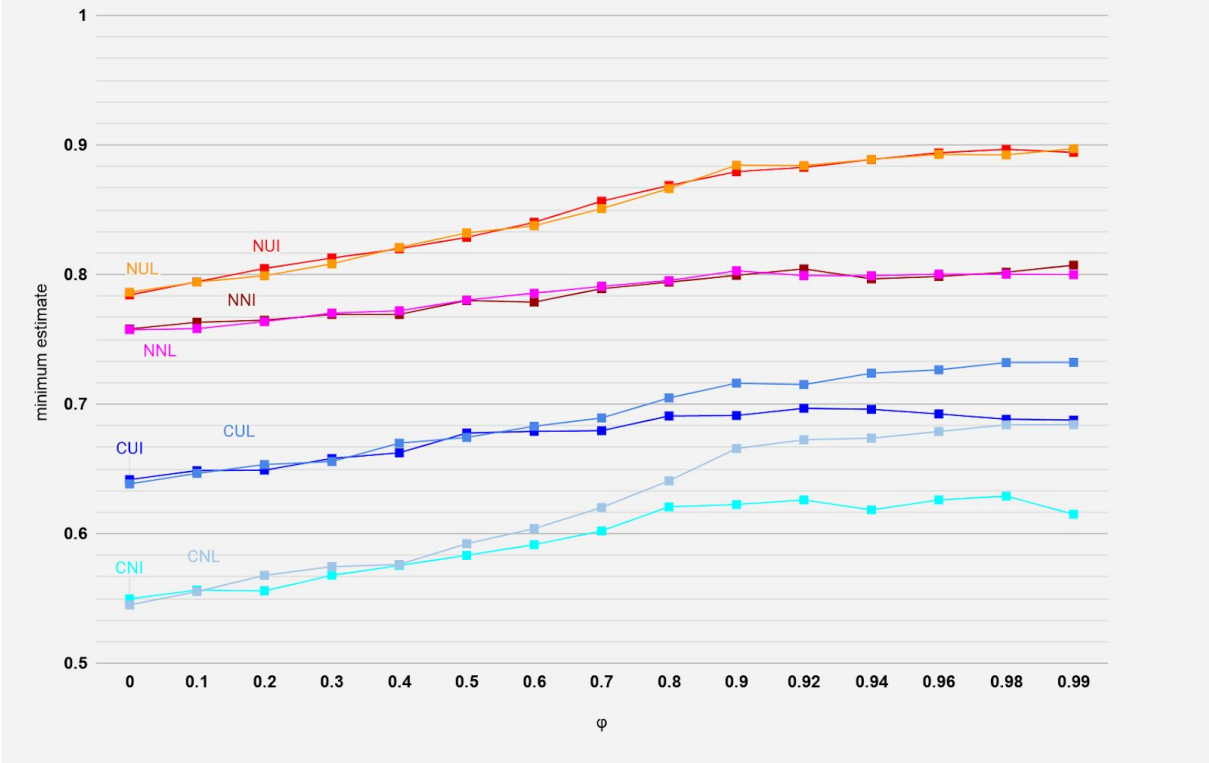
- Δύο γράφοι με 10 χρήστες: ένας χωρίς corrupt και ένας με 2 corrupt χρήστες
- Δύο user-files: Ένα με ομοιόμορφη κατανομή και ένα χωρίς
- Οι 2 επιλογές για το fix-strategy με broken-paths=3 για last-honest

Για τους διάφορους συνδυασμούς των παραπάνω, θα μελετήσουμε πώς το φ συσχετίζεται με τη μέτρηση του εργαλείου fbleau minimum estimate, δηλαδή με την πιθανότητα ο αντίπαλος να μαντέψει λάθος αφού δει την έξοδο του συστήματος, έχοντας οπότε στη διάθεση του την a posteriori γνώση. Ο classification αλγόριθμος που χρησιμοποιήθηκε από το fbleau είναι ο nearest neighbour και οι εκτελέσεις έγιναν γράφοντας στο τερματικό:
`fbleau nn <train.csv> <test.csv>`

Όλα τα user-files είχαν 100.000 συνολικές αποστολές από χρήστες σύμφωνα με την κατανομή του αντίστοιχου apriori-file.

Στο directory /data του repository βρίσκονται τα αρχεία που χρησιμοποιήθηκαν και παρήχθησαν για/από τις εκτελέσεις.

Το παρακάτω γράφημα παρουσιάζει 8 γραμμές κάθε μία από τις οποίες αποτελεί συνάρτηση του φ με το minimum estimate (πιθανότητα ο αντίπαλος να μαντέψει λάθος τον αποστολέα κάποιου μηνύματος) για κάποιο συνδυασμό των εξής:
1ο γράμμα: C (corrupted) ή N (no corrupted), αν περιέχει ή όχι ο γράφος corrupt χρήστες
2ο γράμμα: U (uniform distribution) ή N (not uniform distribution), αν η κατανομή αποστολέων είναι ομοιόμορφη ή όχι.
3ο γράμμα: L (last-honest) ή I (initiator), για το ποια στρατηγική επανόρθωσης μονοπατιού χρησιμοποιήθηκε.



Στο γράφημα αυτό, γραμμές που αντιστοιχούν σε εκτελέσεις που περιλαμβάνουν corrupt χρήστες έχουν τόνους του μπλε, ενώ αυτές που δεν περιλαμβάνουν corrupt χρήστες έχουν τόνους του κόκκινου.
Από το γράφημα μπορούμε να εξάγουμε τα εξής συμπεράσματα, τα οποία συμπίπτουν και με τη θεωρία διαρροής πληροφορίας όσον αφορά το πρωτόκολλο crowds:

1. Η ύπαρξη corrupt χρηστών στο γράφο, έστω και 2 χρηστών από τους 10 συνολικά που χρησιμοποιήσαμε, μειώνει δραματικά την πιθανότητα του αντιπάλου να μαντέψει λάθος. Παρατηρούμε ότι όλες οι μπλε γραμμές είναι κάτω από τις κόκκινες.

2. Η στρατηγική διόρθωσης μονοπατιού last-honest είναι σε όλες τις μπλε γραμμές καλύτερη από τον αντίστοιχο συνδυασμό corrupt-uniform με στρατηγική initiator ή τουλάχιστον τόσο καλή όσο αυτός. Παρατηρούμε ότι για φ=0 ξεκινούν μαζί και καθώς αυτό αυξάνεται, η πιθανότητα λανθασμένης εκτίμησης αυξάνεται περισσότερο για τις γραμμές με last-honest στρατηγική.
3. Το 2 ισχύει μόνο για τις μπλε γραμμές. Στις κόκκινες παρατηρούμε ότι η στρατηγική διόρθωσης δεν επηρεάζει καθόλου, ούτε για NU ούτε για NN. Ακολουθεί λογικά από το ότι δεν υπάρχουν corrupt χρήστες.
4. Οι πιο “ανώνυμες" γραμμές είναι η NUL και η NUI καθώς περιέχουν τον καλύτερο συνδυασμό χαρακτηριστικών που μελετάμε από όλες: δεν έχουν corrupt χρήστες και η κατανομή αποστολέων είναι ομοιόμορφη, άρα ο αντίπαλος (στην περίπτωση αυτή ο server) δεν έχει προϋπάρχουσα γνώση την οποία μπορεί να εκμεταλλευτεί (η στρατηγική διόρθωσης δεν επηρεάζει).
5. Εφόσον στις κόκκινες γραμμές δεν έχουμε corrupt χρήστες και η στρατηγική διόρθωσης δεν επηρεάζει, παρατηρούμε 2 ζευγάρια χωρισμένα σύμφωνα με την ύπαρξη ή όχι ομοιόμορφης κατανομής αποστολέων. Η ομοιόμορφη κατανομή αυξάνει την πιθανότητα λανθασμένης εκτίμησης του αντιπάλου. Τα αντίστοιχα ζευγάρια στις μπλε ξεκινούν από το ίδιο minimum estimate και χωρίζονται σταδιακά καθώς το φ αυξάνεται.
6. Όλες οι γραμμές διατηρούν σταθερή ανοδική πορεία μέχρι το φ να φτάσει στο 0.9, όπου και αρχίζει να σταθεροποιείται γύρω από κάποια μέγιστη τιμή. Αυτό σημαίνει ότι από το σημείο αυτό και μετά, δεν μπορούμε να κάνουμε τον αντίπαλο να μαντέψει χειρότερα αυξάνοντας το φ, δηλαδή φτάσαμε την πιθανότητα λάθους εκτίμησής του όσο πιο κοντά γίνεται στην αρχική γνώση που είχε. (Δεν βγάζει νόημα να γίνει χειρότερη από την αρχική, καθώς απλά δεν θα λάμβανε υπ’ όψιν την καινούρια γνώση που συμπέρανε βλέποντας την έξοδο του συστήματος.)