Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής - Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Εφαρμογή για κινητές συσκευές "CoronaTracker" με στόχο την ιχνηλάτηση των επαφών των κρουσμάτων του COVID-19.** **Application for mobile devices "CoronaTracker" with the aim of tracing the contacts of COVID-19 cases.**
Ονοματεπώνυμο Φοιτητή	**Αναστάσιος Κόλλιας**
Πατρώνυμο	**Βασίλειος**
Αριθμός Μητρώου	**ΜΠΣΠ/ 19018**
Επιβλέπων	**Αλέπης Ευθύμιος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Απρίλιος 2021**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Αλέπης Ευθύμιος	Πατσάκης Κωνσταντίνος	Βίρβου Μαρία
Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής	Καθηγήτρια

Ευχαριστίες

Θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέπων καθηγητή μου για την παρούσα Μεταπτυχιακή Διατριβή κ. Αλέπη Ευθύμιο, που μου έδωσε αυτή την ευκαιρία να εργαστώ πάνω σε ένα θέμα τόσο μείζονος σημασίας για την σημερινή εποχή την οποία βιώνουμε, όπως και τον συμφοιτητή μου Βασίλειο Ζωγράφο, με τον οποίον και συνεργαστήκαμε για την ανάλυση και την υλοποίηση της παρούσας εφαρμογής.

Περίληψη

Τη σήμερον ημέρα, οι εποχές οι οποίες βιώνουμε αποτελούν κάτι πρωτόγνωρο για το σύνολο της ανθρωπότητας. Η πανδημία της νόσου του κορονοϊού, χωρίς να κάνει καμία διάκριση σε φυλή, φύλο και ηλικία, έχει οδηγήσει στην απώλεια πολυάριθμων συνανθρώπων μας, έχει μεταβάλλει δραματικά την καθημερινότητα όλων μας, ενώ ταυτόχρονα έχει επιφέρει τεράστιες οικονομικές και υγειονομικές καταστροφές. Ως συνέπεια, καλούμαστε όλοι μας να συνεισφέρουμε σε οποιοδήποτε βαθμό δύναται στην καταπολέμηση αυτού του φαινομένου.

Επομένως, ο στόχος της παρούσας εργασίας κινείται γύρω από τον άξονα των εφαρμογών smartphone που πραγματοποιούν ιχνηλάτηση των επαφών των κρουσμάτων του COVID-19.

Από την μία πλευρά, θα καλύψουμε κάποια σημεία μείζονος σημασίας αναφορικά με τέτοιου είδους εφαρμογές όπως, τις διάφορες στρατηγικές και αρχιτεκτονικές αυτών, καθώς και τους κινδύνους που εγκυμονούν τόσο σε επίπεδο ιδιωτικότητας όσο και σε επίπεδο επιθέσεων.

Από την άλλη πλευρά, αλλά εξίσου σημαντικά, θα παρουσιάσουμε την δική μας προσπάθεια για την ανάπτυξη μιας τέτοιας εφαρμογής ιχνηλάτησης για Android smartphones εν ονόματι “CoronaTracker”, η οποία έχει ως στόχο να αποτελέσει μια πλήρη υλοποίηση, από την πλευρά της λειτουργικότητας, μιας εφαρμογής ιχνηλάτησης που θα μπορούσε κάλλιστα να προσθέσει ένα μικρό λιθαράκι στην μάχη ενάντια αυτής της πρωτοφανούς πανδημίας.

Λέξεις Κλειδιά: COVID-19, ιχνήλατηση επαφών, εφαρμογές smartphone

Abstract

Today, the times we are experiencing are something unprecedented for all of the humanity. The pandemic of coronavirus, without making any distinction in race, gender and age, has led to the loss of many of our fellow human beings, has dramatically changed the daily lives of all of us, while at the same time has caused enormous economic and health disasters. As a result, we are all called upon to contribute to the best of our ability to combat this phenomenon.

Therefore, the aim of the present thesis revolves around the axis of smartphone applications that track COVID-19 case contacts.

On the one hand, we will cover some points of major importance regarding those applications such as, their various strategies and architectures, as well as the risks involved both in terms of privacy and in terms of attacks.

On the other hand, but equally important, we will present our own effort to develop such a tracking application for Android smartphones called "CoronaTracker", which aims to be a complete implementation, in terms of functionality, of a tracking application that could well play its part to the battle against this unprecedented pandemic.

Keywords: COVID-19, contact tracking, smartphone applications

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1° :

Ανίχνευση επαφών COVID-19 μέσω εφαρμογών smartphone

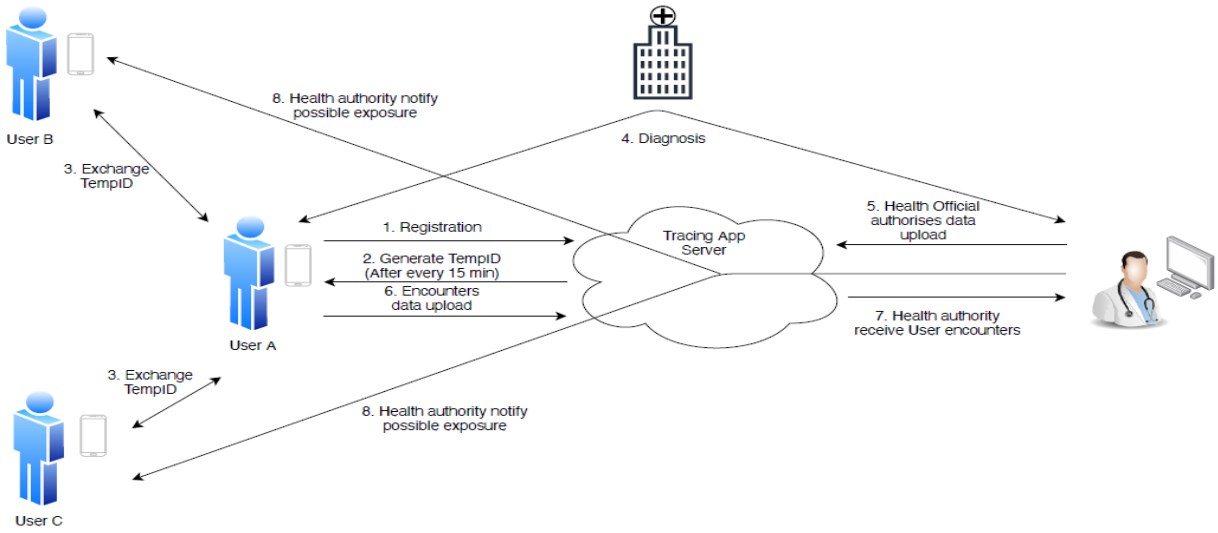
Οι εφαρμογές ιχνηλάτησης επαφών COVID-19, παρ' ότι αποτελούν ένα πολύ σημαντικό εργαλείο για την αντιμετώπιση της πανδημίας, θα πρέπει να λάβουμε υπόψη και πολυάριθμα σοβαρά ζητήματα σχετικά με τις αρχιτεκτονικές αυτών, τον βαθμό της ιδιωτικότητας που προσφέρουν, καθώς και τις πιθανές επιθέσεις που μπορούν να δεχθούν.

1.1 Αρχιτεκτονική Συστήματος

Ο τύπος της αρχιτεκτονικής που υιοθετήθηκε για τις πτυχές της συλλογής δεδομένων της ανίχνευσης εφαρμογών αποτέλεσε αντικείμενο πολλών συζητήσεων λόγω τόσο των ζητημάτων ασφάλειας όσο και απορρήτου. Θα συζητήσουμε τρεις ξεχωριστές αρχιτεκτονικές συστήματος που χρησιμοποιούνται συνήθως ή προτείνονται για την ανάπτυξη εφαρμογών ανίχνευσης COVID-19. Αυτές είναι οι κεντρικές, οι αποκεντρωμένες και οι υβριδικές προσεγγίσεις που συνδυάζουν χαρακτηριστικά τόσο από την κεντρική όσο και από τις αποκεντρωμένες αρχιτεκτονικές.

1.1.1 Κεντρική Αρχιτεκτονική

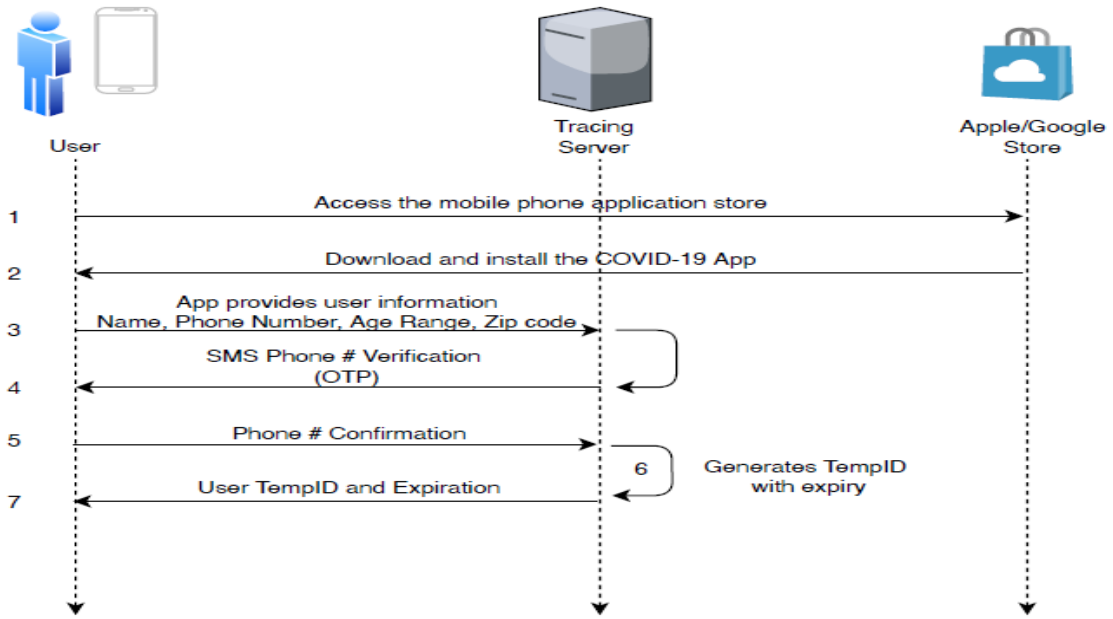
Η εικόνα 1 δείχνει τις κύριες οντότητες και αλληλεπιδράσεις μιας κεντρικής αρχιτεκτονικής. Σημειώνουμε ότι η κεντρική αρχιτεκτονική που περιγράφουμε βασίζεται στο πρωτόκολλο Bluetrace. Η αρχική απαίτηση για την εφαρμογή είναι ότι ένας χρήστης πρέπει να προ-εγγραφεί στον κεντρικό διακομιστή. Ο διακομιστής δημιουργεί ένα προσωρινό αναγνωριστικό που διατηρεί το απόρρητο (TempID) για κάθε συσκευή. Αυτό το TempID στη συνέχεια κρυπτογραφείται με ένα μυστικό κλειδί (γνωστό μόνο στην κεντρική αρχή διακομιστή) και αποστέλλεται στη συσκευή. Οι συσκευές ανταλλάσσουν αυτά τα TempID (σε μηνύματα συναντήσεων Bluetooth) όταν έρχονται σε στενή επαφή μεταξύ τους. Μόλις ένας χρήστης διαγνωστεί θετικός, μπορεί να μεταφορτώσει εθελοντικά όλα τα αποθηκευμένα μηνύματα συνάντησής του στον κεντρικό διακομιστή. Ο διακομιστής χαρτογραφεί τα TempID σε αυτά τα μηνύματα σε άτομα για να εντοπίσει τις επαφές σε κίνδυνο. Δίδονται τώρα περισσότερες λεπτομέρειες σχετικά με τις βασικές διαδικασίες της κεντρικής αρχιτεκτονικής.



Εικόνα 1: Κεντρική Αρχιτεκτονική Εφαρμογών Ιχνηλάτησης

1) *Φάση Εγγραφής*

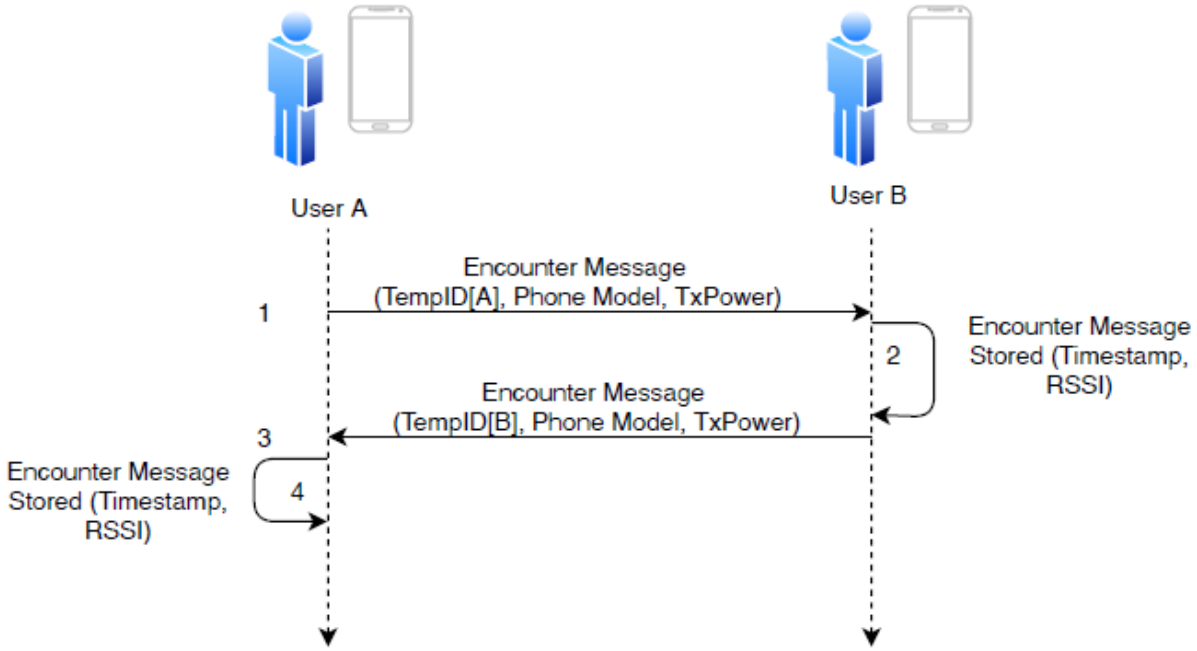
Η εικόνα 2 δείχνει τα βήματα που απαιτούνται για την εγγραφή ενός χρήστη σε μια κεντρική αρχιτεκτονική. Ένας χρήστης κάνει λήψη της εφαρμογής (βήματα 1 και 2) και καταχωρεί λεπτομέρειες όπως όνομα, αριθμό κινητού τηλεφώνου, ηλικιακή ομάδα και ταχυδρομικό κώδικα στον διακομιστή (βήμα 3). Ο διακομιστής επαληθεύει τον αριθμό του κινητού στέλνοντας έναν κωδικό μίας χρήσης (OTP) μέσω SMS (βήματα 4 και 5). Κατά την επαλήθευση, ο διακομιστής υπολογίζει ένα TempID (βήμα 6), το οποίο ισχύει μόνο για σύντομο χρονικό διάστημα (ο προτεινόμενος χρόνος λήξης του Bluetrace είναι 15 λεπτά). Το TempID και ο χρόνος λήξης μεταδίδονται στη συνέχεια στην εφαρμογή του χρήστη.



Εικόνα 2: Διαδικασία Εγγραφής Εφαρμογής Ιχνηλάτησης Κεντρικής Αρχιτεκτονικής

2) *Εγγραφή συναντήσεων/ πληροφοριών επαφών*

Μόλις ένας χρήστης έρθει σε επαφή με έναν άλλο χρήστη της εφαρμογής, ανταλλάσσουν ένα «μήνυμα συναντήσεως» χρησιμοποιώντας το Bluetooth, όπως παρουσιάζεται στην εικόνα 3. Ένα μήνυμα συνάντησης περιλαμβάνει την ανταλλαγή TempID, μοντέλου τηλεφώνου και ισχύος μετάδοσης (TxPower) (βήματα 1 και 3). Κάθε συσκευή καταγράφει επίσης την ένδειξη ισχύος λήψης σήματος (RSSI) και τη χρονική σήμανση της παράδοσης μηνυμάτων (βήματα 2 και 4). Είναι σημαντικό να σημειωθεί ότι οι αριθμοί τηλεφώνων δεν περιλαμβάνονται σε αυτά τα μηνύματα. Δεδομένου ότι τα TempID δημιουργούνται και κρυπτογραφούνται από το διακομιστή δεν αποκαλύπτεται κανένα από τα προσωπικά στοιχεία του χρήστη της εφαρμογής. Έτσι, και οι δύο χρήστες της εφαρμογής έχουν συμμετρική εγγραφή της συνάντησης που είναι αποθηκευμένη στον τοπικό χώρο αποθήκευσης των αντίστοιχων τηλεφώνων τους. Το πρωτόκολλο χρησιμοποιεί μια προσωρινή μαύρη λίστα για να αποφευχθεί η εγγραφή διπλών επαφών από έναν χρήστη. Έτσι, μόλις ένας χρήστης λάβει ένα μήνυμα συναντήσεως, η εφαρμογή θα τοποθετήσει αυτόματα σε μια μαύρη λίστα τον αποστολέα για μικρό χρονικό διάστημα.



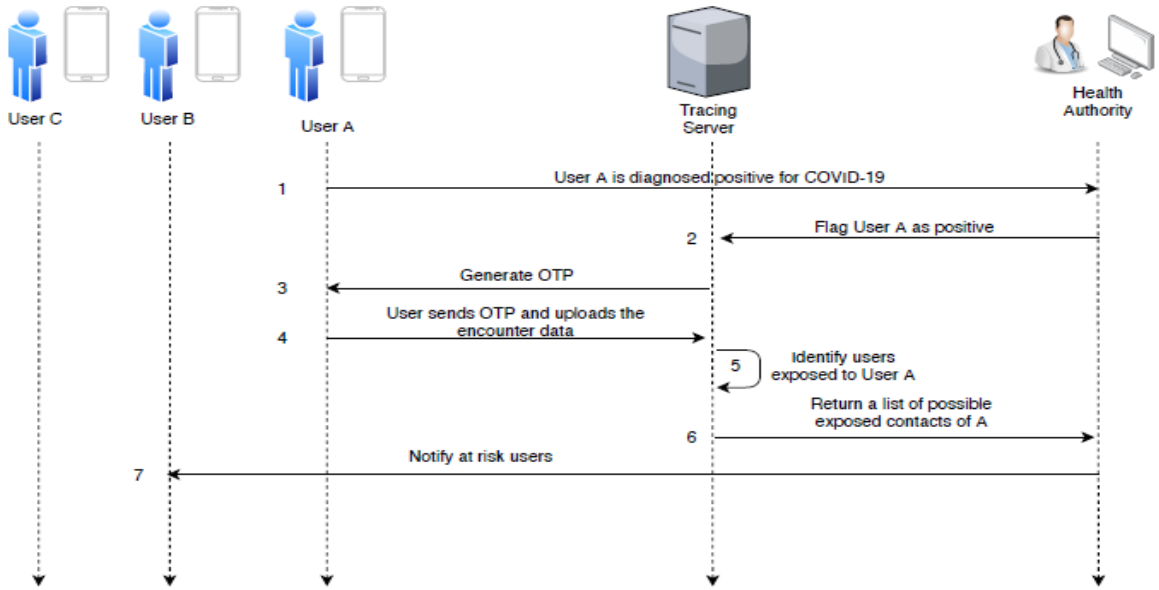
Εικόνα 3: Λειτουργία Ανταλλαγής Επαφών Εφαρμογής Ιχνηλάτησης Κεντρικής Αρχιτεκτονικής

3) *Μεταφόρτωση δεδομένων συναντήσεων*

Όλες οι εγγραφές συναντήσεων αποθηκεύονται τοπικά και δεν φορτώνονται αυτόματα στο διακομιστή. Η εικόνα 4 δείχνει τη ροή της εφαρμογής όταν ένας χρήστης διαγνωστεί θετικός για τον COVID-19 (βήμα 1). Ο υπάλληλος υγείας επιβεβαιώνει εάν ο χρήστης έχει εγκαταστήσει την εφαρμογή ανίχνευσης και επισημαίνει τον χρήστη ως μολυσμένο (βήμα 2). Η μεταφόρτωση των δεδομένων συνάντησης είναι προαιρετική. Εάν ο χρήστης συμφωνήσει να ανεβάσει τα δεδομένα, ο υπεύθυνος υγείας το προωθεί στον διακομιστή backend και ο διακομιστής δημιουργεί ένα OTP για επαλήθευση (βήμα 3). Μόλις επαληθευτεί, τα δεδομένα συνάντησης μεταφορτώνονται στον διακομιστή (βήμα 4).

4) *Επεξεργασία των μεταφορτωμένων δεδομένων από την πλευρά του διακομιστή*

Ο διακομιστής παραλαμβάνει την λίστα μηνυμάτων συνάντησης, αποκρυπτογραφώντας κάθε TempID με το μυστικό κλειδί του. Αυτό το TempID αντιστοιχεί στη συνέχεια στον αριθμό κινητού του χρήστη. Ο διακομιστής χρησιμοποιεί τις τιμές TxPower και RSSI για να προσεγγίσει την απόσταση (εγγύτητα) που διαχωρίζει τους χρήστες κατά τη διάρκεια της αναφερόμενης συνάντησης. Η εκτίμηση εγγύτητας μπορεί επίσης να πραγματοποιηθεί τοπικά στο τηλέφωνο, αλλά αυτό έχει επιπτώσεις στη χρήση της μπαταρίας. Αυτά τα δεδομένα εγγύτητας, σε συνδυασμό με τις χρονικές σημάνσεις, χρησιμοποιούνται για να εξακριβωθεί το προφίλ κινδύνου (εγγύτητα και διάρκεια) της συνάντησης (βήμα 5, εικόνα 4). Μια λίστα καταρτίζεται με όλες τις απαιτούμενες πληροφορίες (βήμα 6) για περαιτέρω επεξεργασία από τον αρμόδιο υπάλληλο υγείας (βήμα 7).



Εικόνα 4: Ειδοποίηση Εφαρμογής Ιχνηλάτησης Κεντρικής Αρχιτεκτονικής

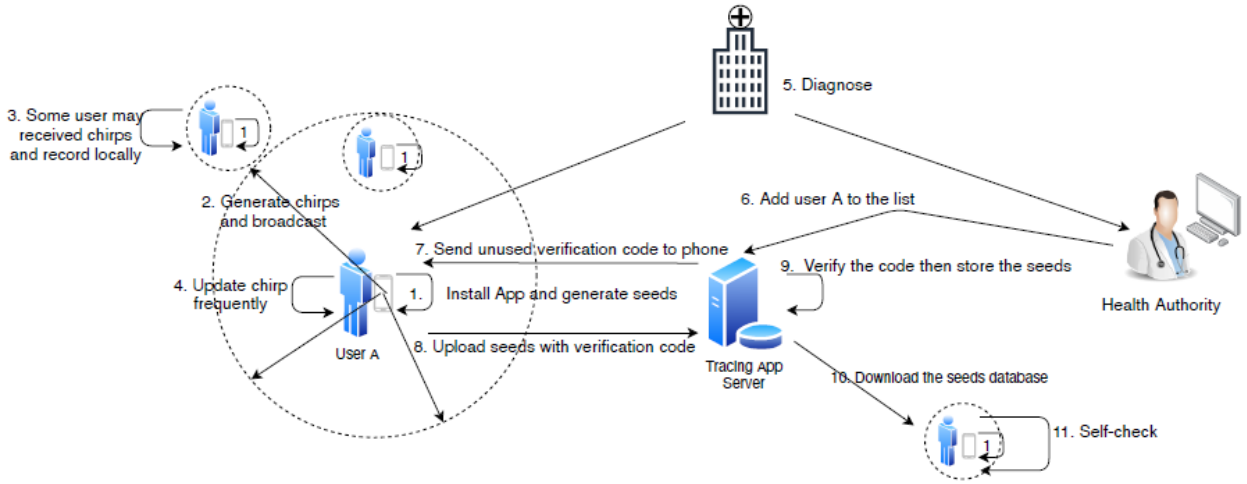
Συνοψίζοντας, στην κεντρική αρχιτεκτονική, ο κεντρικός διακομιστής παίζει βασικό ρόλο στην εκτέλεση βασικών λειτουργιών, όπως η αποθήκευση κρυπτογραφημένων πληροφοριών PII, η δημιουργία ανώνυμων TempIDs, η ανάλυση κινδύνου και οι ειδοποιήσεις για στενές επαφές. Αυτή η συσσώρευση ευθυνών εγείρει ανησυχίες σχετικά με το απόρρητο. Ο διακομιστής θεωρείται αξιόπιστος σε αυτήν την αρχιτεκτονική, με ορισμένες χώρες να εισάγουν αυστηρούς κανονισμούς προστασίας της ιδιωτικής ζωής για τη διασφάλιση της χρήσης και του κύκλου ζωής των συλλεγόμενων δεδομένων.

1.1.2 Αποκεντρωμένη Αρχιτεκτονική

Σε αντίθεση με την κεντρική αρχιτεκτονική, η αποκεντρωμένη αρχιτεκτονική προτείνει τη μεταφορά βασικών λειτουργιών στις συσκευές του χρήστη, αφήνοντας τον διακομιστή με ελάχιστη συμμετοχή στη διαδικασία ανίχνευσης επαφών. Η ιδέα είναι να ενισχυθεί το απόρρητο των χρηστών δημιουργώντας ανώνυμα αναγνωριστικά στις συσκευές χρηστών (διατηρώντας τις ταυτότητες των πραγματικών χρηστών μυστικές από τους άλλους χρήστες καθώς και από τον διακομιστή) και την επεξεργασία των ειδοποιήσεων έκθεσης σε μεμονωμένες συσκευές αντί για τον κεντρικό διακομιστή.

Λαμβάνουμε το πρωτόκολλο Private Automated Contact Tracing (PACT) ως βάση για την περιγραφή της αποκεντρωμένης αρχιτεκτονικής. Η αποκεντρωμένη προσέγγιση δεν απαιτεί από τους χρήστες της εφαρμογής να «προ-εγγραφούν» πριν από τη 1^{η} χρήση, αποφεύγοντας έτσι την αποθήκευση οποιουδήποτε PII στον διακομιστή. Οι συσκευές δημιουργούν τα τυχαία αναγνωριστικά τους (χρησιμοποιούνται ως είσοδος για μια ψευδοτυχαία συνάρτηση), τα οποία χρησιμοποιούνται σε συνδυασμό με την τρέχουσα ώρα για τη δημιουργία ψευδωνύμων διατήρησης της ιδιωτικής ζωής ή “chirps” με πολύ σύντομη διάρκεια ζωής περίπου 1 λεπτό (βλ. Εικόνα 5). Αυτά τα “chirps” ανταλλάσσονται στη συνέχεια περιοδικά με άλλες συσκευές που έρχονται σε στενή επαφή. Μόλις ένας χρήστης διαγνωστεί θετικός με COVID-19, μπορεί να προσφερθεί εθελοντικά να ανεβάσει τα “seeds” του

και τις σχετικές πληροφορίες χρόνου σε έναν κεντρικό διακομιστή. Αυτό έρχεται σε αντίθεση με την κεντρική αρχιτεκτονική όπου φορτώνεται ο πλήρης κατάλογος των μηνυμάτων συνάντησης. Η μεταφόρτωση “seeds”, αντί όλων των χρησιμοποιημένων “chrips”, βελτιώνει το latency και παρέχει βελτιωμένη χρήση του bandwidth.

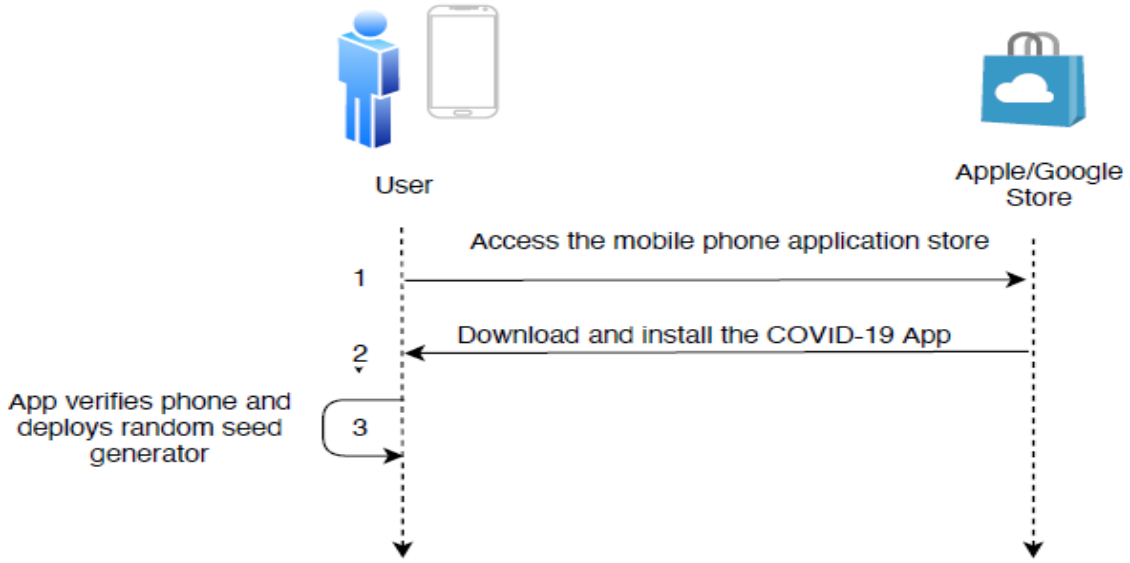


Εικόνα 5: Αποκεντρωμένη Αρχιτεκτονική Εφαρμογών Ιχνηλάτησης

Ο κεντρικός διακομιστής λειτουργεί μόνο ως σημείο συνάντησης, παρόμοια με έναν πίνακα ανακοινώσεων για τη διαφήμιση των “seeds” των μολυσμένων χρηστών. Αυτός ο διακομιστής θεωρείται ως «ειλικρινής αλλά περίεργος». Άλλοι χρήστες της εφαρμογής μπορούν να κατεβάσουν αυτά τα “seeds” για να αναδημιουργήσουν τα “chirps” (χρησιμοποιώντας χρονικές σημάνσεις) που στάλθηκαν από τους μολυσμένους χρήστες. Ο διακομιστής, καθώς και άλλοι χρήστες, δεν μπορούν να αντλήσουν στοιχεία αναγνώρισης, παρά μόνο γνωρίζοντας τα “seeds” και τα “chirps”. Μόνο οι άλλοι χρήστες της εφαρμογής μπορούν να πραγματοποιήσουν ανάλυση κινδύνου για να ελέγξουν εάν εκτίθενται για αρκετά μεγάλο χρονικό διάστημα. Αυτή η μονόδρομη αναζήτηση σε σχέση με τα ληφθένα “seeds” περιορίζει τη λειτουργικότητα του διακομιστή και ανακουφίζει ορισμένους από τους κινδύνους απορρήτου. Παρακάτω δίνονται περισσότερες λεπτομέρειες σχετικά με τις βασικές διαδικασίες της αποκεντρωμένης αρχιτεκτονικής.

1) *Εγκατάσταση εφαρμογής*

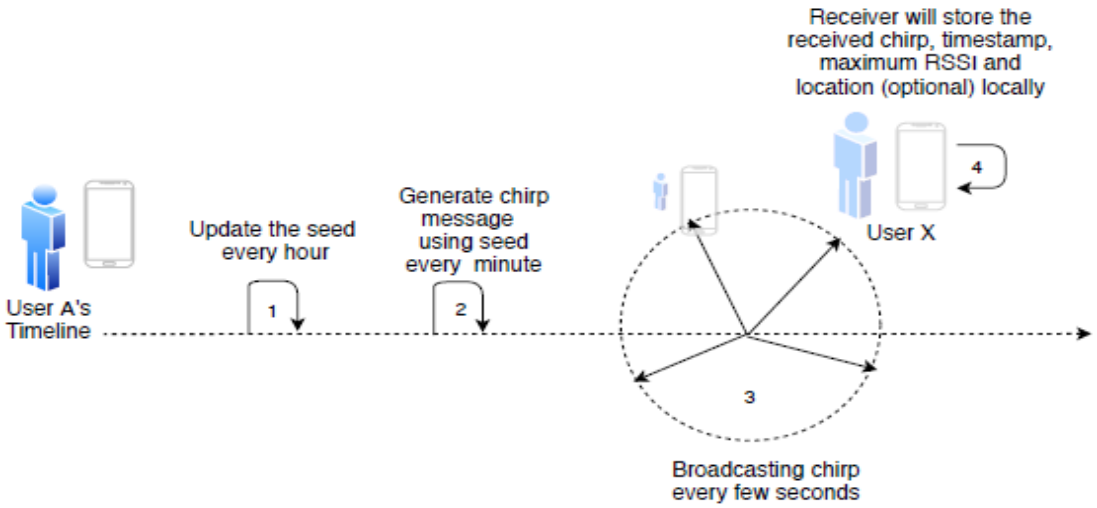
Οι εφαρμογές ανίχνευσης COVID-19 που υιοθετούν την αποκεντρωμένη αρχιτεκτονική δεν απαιτούν απαραίτητα διαδραστική διαδικασία εγγραφής κατά το στάδιο εγκατάστασης της εφαρμογής. Η διαδικασία εγκατάστασης της εφαρμογής επαληθεύει μόνο το smartphone ενός χρήστη και αναπτύσσει έναν αλγόριθμο δημιουργίας τυχαίων “seeds” που δεν είναι συνδεδεμένα με το τηλέφωνο (βλ. Εικόνα 6).



Εικόνα 6: Διαδικασία Εγκατάστασης Εφαρμογής Ιχνηλάτησης Αποκεντρωμένης Αρχιτεκτονικής

2) *Δημιουργία "seeds", "chirps" και ανταλλαγή "chirps"*

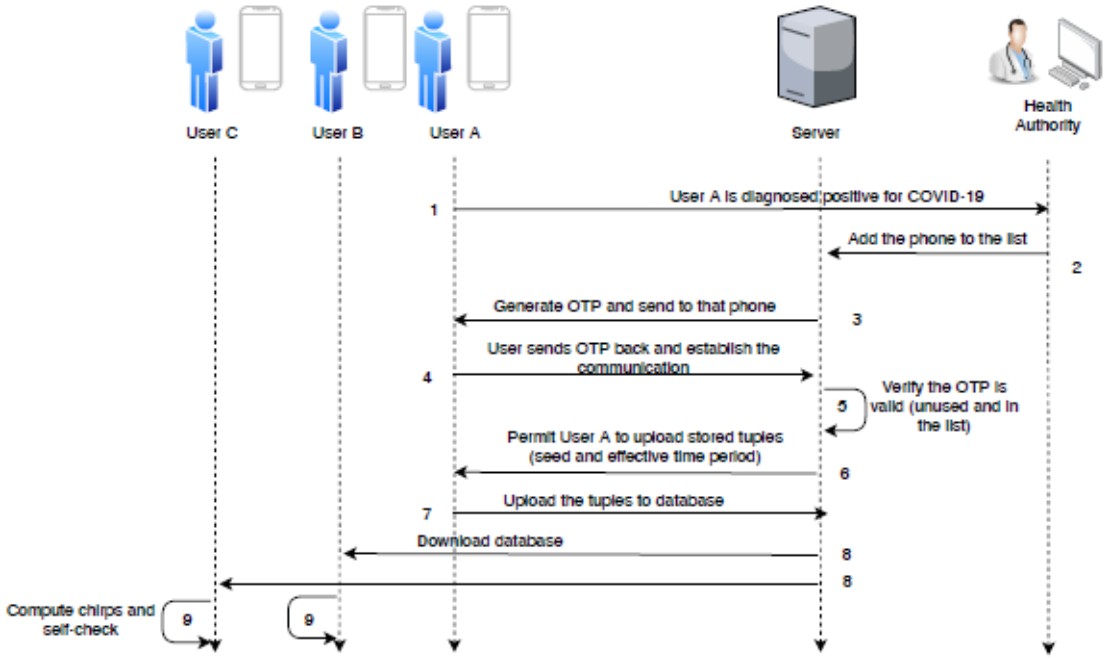
Μόλις εγκατασταθεί η αποκεντρωμένη εφαρμογή παρακολούθησης, το "seed" δημιουργείται (με περίοδο λήξης μιας ώρας) από τη συσκευή του χρήστη (βλ. Εικόνα 7). Αυτό το "seed" μαζί με τον τρέχων χρόνο, στη συνέχεια χρησιμοποιούνται σε μια ψευδοτυχαία συνάρτηση για να δημιουργήσουν το "chirp". Τα "chirps" δεν συνδέονται με ένα άτομο ή το τηλέφωνό τους - επομένως εξ' ορισμού, είναι ανώνυμα. Έπειτα, η εφαρμογή δημιουργεί νέα "chirps" με χρονικό βαθμό λεπτομέρειας το 1 λεπτό. Αυτά μεταδίδονται κάθε λίγα δευτερόλεπτα μέσω του σήματος Bluetooth. Στο τηλέφωνο του «ακροατή», η εφαρμογή αποθηκεύει αυτόματα όλα τα "chirps" που λαμβάνονται (βήμα 4 στην εικόνα 7). Οι πληροφορίες που αποθηκεύονται στην εφαρμογή λήψης περιλαμβάνουν το "chirp", τη χρονική σήμανση κατά τη λήψη του chirp και τη μέγιστη τιμή RSSI. Αγνοούνται πανομοιότυπα "chirps" εντός 1 λεπτού. Σημειώνεται πως η κρίσιμη διαφορά από την κεντρική αρχιτεκτονική είναι το γεγονός ότι τα TempIDs δημιουργούνται από τον διακομιστή - στην αποκεντρωμένη περίπτωση, τα "seeds" και τα "chirps" δημιουργούνται στην ίδια την συσκευή.



Εικόνα 7: Ανταλλαγή Συναντήσεων Εφαρμογής Ιχνηλάτησης Αποκεντρωμένης Αρχιτεκτονικής

3) *Μεταφόρτωση δεδομένων συναντήσεων*

Εάν ένας χρήστης διαγνωστεί θετικός, του δίνεται ένας μοναδικός «αριθμός άδειας» από την αρμόδια αρχή για να εξουσιοδοτήσει τη μεταφόρτωση όλων των χρησιμοποιημένων “seeds” που αποθηκεύονται τοπικά στο τηλέφωνό τους (απεικονίζεται στην εικόνα 8), καθώς και τους χρόνους δημιουργίας και λήξης των “seeds”. Σημειώστε ότι ο διακομιστής στην αποκεντρωμένη αρχιτεκτονική παίρνει μόνο τα “seeds” που σχετίζονται με έναν μόνο αναγνωρισμένο χρήστη. Αυτό πρέπει να συγκριθεί με την κεντρική αρχιτεκτονική όπου μεταφορτώνεται στον διακομιστή η πλήρης λίστα επαφών (με TempIDs) όλων των ατόμων που συναντώνται.



Εικόνα 8: Διαδικασία Ανίχνευσης Εφαρμογής Ιχνηλάτησης Αποκεντρωμένης Αρχιτεκτονικής

4) *Η διαδικασία ανίχνευσης επαφών*

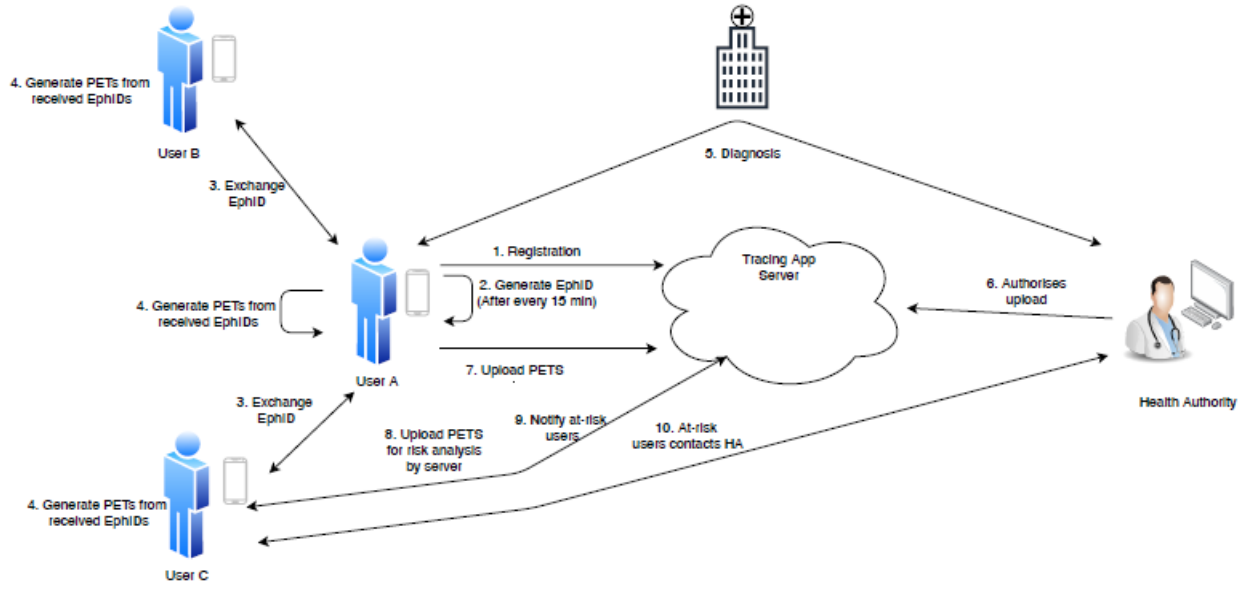
Σε αντίθεση με την κεντρική αρχιτεκτονική, η διαδικασία εντοπισμού κατά την αποκεντρωμένη αρχιτεκτονική εκτελείται τοπικά από τον χρήστη της εφαρμογής στη συσκευή του (αντί για τον κεντρικό διακομιστή). Οι χρήστες της εφαρμογής μπορούν να επικοινωνήσουν με τον διακομιστή, συνήθως μία φορά την ημέρα, για να κατεβάσουν τυχόν “seeds” που ανέβηκαν από μολυσμένους χρήστες. Δεδομένου ότι έχουν ληφθεί τέτοια “seeds” (βήμα 8 στην εικόνα 8), η εφαρμογή του χρήστη στη συνέχεια ανακατασκευάζει όλα τα αντίστοιχα “chirps” (χρησιμοποιώντας ψευδοτυχαίους υπολογισμούς με βάση τα “seeds” και διακριτά χρονικά διαστήματα μεταξύ του χρόνου έναρξης και λήξης). Τέλος, η εφαρμογή πραγματοποιεί αναζήτηση για να ελέγξει εάν κάποια από τις ανακατασκευασμένες πληροφορίες “chirp” εμφανίζεται στο τοπικό αρχείο καταγραφής “chirp”. Εάν ναι, τότε προκύπτουν οι χρόνοι εγγύτητας και διάρκειας (με βάση τις χρονικές σημάνσεις και τις τιμές RSSI) για σκοπούς ανάλυσης κινδύνου, ενώ δεν απαιτείται κάποια ανθρώπινη παρέμβαση.

1.1.3 Υβριδική Αρχιτεκτονική

Στην κεντρική αρχιτεκτονική, ο διακομιστής εκτελεί όλες τις πολύπλοκες εργασίες, π.χ. υπολογισμούς TempID, κρυπτογράφηση, αποκρυπτογράφηση, ανάλυση κινδύνου και ειδοποιήσεις ειδοποιήσεων για τις επαφές υψηλού κινδύνου. Από την άλλη πλευρά, στην περίπτωση της αποκεντρωμένης αρχιτεκτονικής, όλες αυτές οι λειτουργίες ανατίθενται στην συσκευή του εκάστοτε χρήστη, διατηρώντας τον διακομιστή μόνο ως πίνακα ανακοινώσεων για σκοπούς αναζήτησης. Η υβριδική αρχιτεκτονική προτείνει αυτές οι λειτουργίες να χωρίζονται μεταξύ του διακομιστή και των συσκευών. Πιο συγκεκριμένα, η δημιουργία και η διαχείριση του TempID παραμένουν αποκεντρωμένες (δηλαδή, διαχειρίζονται από τις συσκευές) για τη

διασφάλιση του απορρήτου και της ανωνυμοποίησης, ενώ η ανάλυση κινδύνου και οι ειδοποιήσεις πρέπει να αποτελούν ευθύνη του κεντρικού διακομιστή. Υπάρχουν τρεις κύριοι λόγοι για την εκτέλεση της διαδικασίας ανίχνευσης στον διακομιστή: i) Στην αποκεντρωμένη αρχιτεκτονική, ο διακομιστής δεν γνωρίζει τον αριθμό των χρηστών που κινδυνεύουν καθώς οι συσκευές κάνουν αυτήν την ανάλυση κινδύνου χωρίς να λαμβάνουν υπόψη τον διακομιστή. Έτσι, ο διακομιστής δεν διαθέτει στατιστικές πληροφορίες και δεν είναι σε θέση να εκτελέσει οποιαδήποτε ανάλυση δεδομένων για τον εντοπισμό ομάδων έκθεσης. ii) Η ανάλυση κινδύνου και οι ειδοποιήσεις θεωρούνται μια ευαίσθητη διαδικασία που πρέπει να αντιμετωπίζονται από τις αρχές, έχοντας κατά νου τους υπάρχοντες πόρους υποδομής και την κατάσταση της πανδημίας. iii) Οι μεταφορτωμένες πληροφορίες συνάντησης από μολυσμένους χρήστες δεν διατίθενται στους άλλους χρήστες, αλλά διατηρούνται μόνο στον διακομιστή. Αυτό γίνεται για να αποφευχθούν πιθανές επιθέσεις εύρεσης της ταυτότητας των χρηστών στην αποκεντρωμένη αρχιτεκτονική.

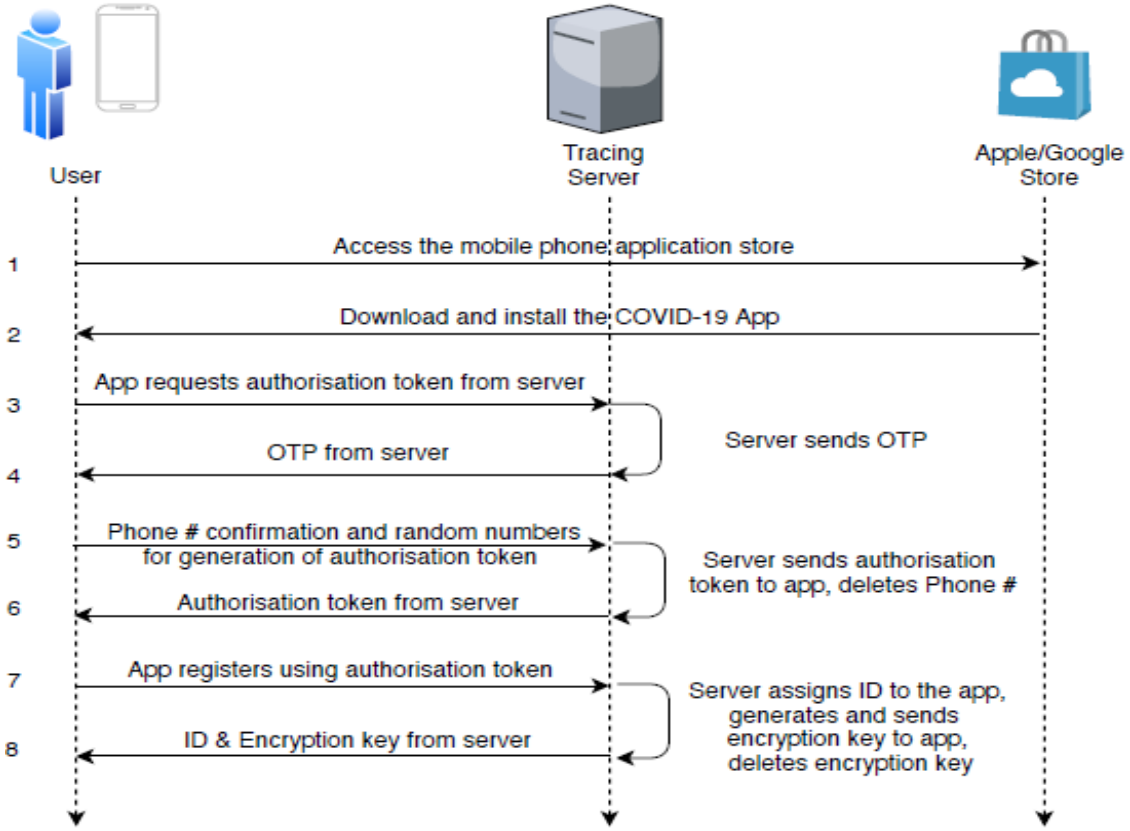
Η εικόνα 9 δείχνει την αλληλουχία αλληλεπιδράσεων στην υβριδική αρχιτεκτονική με βάση το πρωτόκολλο Desire. Αυτό το πρωτόκολλο απαιτεί από τη διαδικασία εγγραφής της εφαρμογής του χρήστη να εκχωρήσει ένα μοναδικό αναγνωριστικό συσκευής χωρίς την καταγραφή PII. Στη συνέχεια, οι συσκευές δημιουργούν κρυπτογραφικά και ανταλλάσσουν ταυτότητες Ephemeral με άλλες συσκευές μέσω BLE. Για κάθε ληφθέν EphID, δημιουργούνται και αποθηκεύονται δύο μη συνδεδεμένα ιδιωτικά διακριτικά (PET) για να αντιπροσωπεύουν μια συνάντηση. Μόλις ένας χρήστης διαγνωστεί θετικός, μια λίστα με τα τοπικά δημιουργημένα PET μεταφορτώνεται στον διακομιστή. Οποιαδήποτε συσκευή μπορεί τώρα να στείλει τα δεύτερα δημιουργημένα διακριτικά PET στον διακομιστή, ο οποίος στη συνέχεια εκτελεί την ανάλυση κινδύνου και τις ειδοποιήσεις. Ο διακομιστής δεν μπορεί να συμπεράνει πληροφορίες αναγνώρισης από τα PET, και όλη η επικοινωνία μεταξύ του διακομιστή και των συσκευών δρομολογείται μέσω ενός διακομιστή μεσολάβησης ή «ανωνυμοποίησης». Παρατίθενται παρακάτω περισσότερες λεπτομέρειες σχετικά με τις βασικές διαδικασίες της υβριδικής αρχιτεκτονικής.



Εικόνα 9: Διαδικασία Ανίχνευσης Εφαρμογής Ιχνηλάτησης Υβριδικής Αρχιτεκτονικής

1) *Εγκατάσταση και εγγραφή*

Η διαδικασία εγγραφής στην υβριδική αρχιτεκτονική απαιτεί μια διαδικασία ελέγχου ταυτότητας δύο βημάτων, όπου ο αριθμός τηλεφώνου επαληθεύεται από το OTP και η εφαρμογή επαληθεύεται μέσω ενός διακριτικού εξουσιοδότησης που εκδίδεται από τον διακομιστή. Η εικόνα 10 παρουσιάζει την λεγόμενη διαδικασία. Καθώς ο διακομιστής δεν επιτρέπεται να αποθηκεύει κανένα PII, διαγράφει τον αριθμό τηλεφώνου μετά την επαλήθευση. Στη συνέχεια, ο διακομιστής εκχωρεί στην εφαρμογή ένα μοναδικό αναγνωριστικό και δημιουργεί ένα κλειδί κρυπτογράφησης που αποστέλλεται στην εφαρμογή. Στη συνέχεια, ο διακομιστής διαγράφει το κλειδί κρυπτογράφησης. Ο client, μελλοντικά, θα ταυτοποιηθεί χρησιμοποιώντας αυτό το αναγνωριστικό.

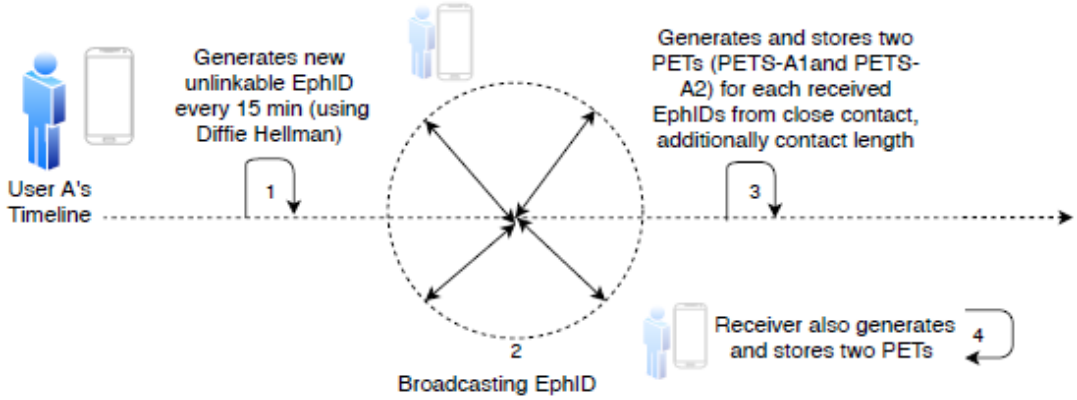


Εικόνα 10: Διαδικασία Εγγραφής Εφαρμογής Ιχνηλάτησης Υβριδικής Αρχιτεκτονικής

2) *Δημιουργία και ανταλλαγή Ephemeral IDs*

Στη φάση λειτουργίας, η συσκευή δημιουργεί ένα νέο Ephemeral ID (EphID) χρησιμοποιώντας τον μηχανισμό ανταλλαγής κλειδιών Diffie Hellman, ο οποίος διαρκεί για συνήθως 15 λεπτά και συγχρονίζεται με το διάστημα περιστροφής της διεύθυνσης Bluetooth MAC. Η συσκευή ξεκινά τη μετάδοση αυτού του EphID μέσω Bluetooth. Μόλις ληφθεί ένα EphID από άλλη συσκευή, η εφαρμογή δημιουργεί δύο PET. Η εφαρμογή διατηρεί δύο

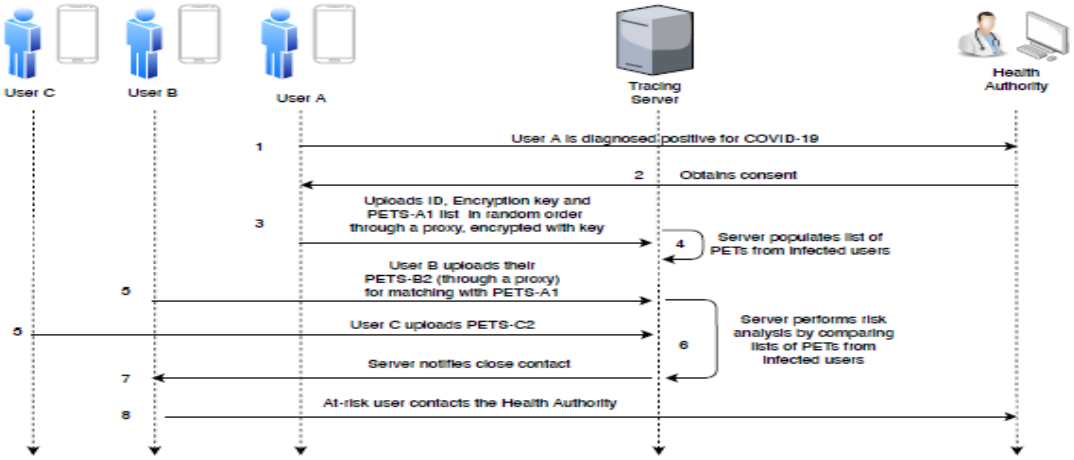
πίνακες, που αναφέρονται ως πίνακες μεταφόρτωσης και ερωτημάτων. Το ένα PET αποθηκεύεται στον πίνακα ερωτημάτων και το άλλο, μαζί με την τρέχουσα ώρα και τη διάρκεια της επαφής, αποθηκεύεται στον πίνακα μεταφόρτωσης.



Εικόνα 11: Διαδικασία Συνάντησης Εφαρμογής Ιχνηλάτησης Υβριδικής Αρχιτεκτονικής

3) *Μεταφόρτωση δεδομένων συναντήσεων*

Μόλις ένας χρήστης διαγνωστεί με COVID-19, απαιτείται ρητά η συγκατάθεσή του για τη μεταφόρτωση των δεδομένων. Ο χρήστης ανεβάζει το αναγνωριστικό, το κλειδί κρυπτογράφησης και το PET στον πίνακα μεταφόρτωσης μαζί με τις τιμές χρόνου και διάρκειας (βήματα 1-3 Εικόνα 12). Ο διακομιστής καταγράφει αυτές τις τιμές PET και τα σχετικά δεδομένα και χρησιμοποιεί το κλειδί κρυπτογράφησης για την ενημέρωση της εγγραφής χρήστη (κατάσταση).



Εικόνα 12: Διαδικασία Ειδοποίησης Εφαρμογής Ιχνηλάτησης Υβριδικής Αρχιτεκτονικής

4) *Διαδικασία ανίχνευσης επαφών*

Κάθε χρήστης που θέλει να ελέγξει την έκθεση κινδύνου του σε μολυσμένες περιπτώσεις ανεβάζει τα PET στον πίνακα ερωτημάτων στον διακομιστή, χρησιμοποιώντας ένα διακομιστή μεσολάβησης ή ανωνυμοποίησης (βήμα 5 Εικόνα 12). Ο διακομιστής εκτελεί την ανάλυση κινδύνου αντιστοιχίζοντας τα PET από τον πίνακα ερωτημάτων με τα PET που ανέβασε ο μολυσμένος χρήστης. Χρησιμοποιώντας τις τιμές χρόνου και διάρκειας, ο διακομιστής αξιολογεί

εάν ο χρήστης κινδυνεύει ή όχι. Κάθε χρήστης που κινδυνεύει μπορεί να ειδοποιηθεί μέσω της εφαρμογής για να επικοινωνήσει με την υγειονομική αρχή. Να σημειωθεί ότι ο διακομιστής δεν είναι σε θέση να αναγνωρίσει κανέναν χρήστη μιας και βασίζεται αποκλειστικά στις τιμές PET του.

1.2 Διαχείριση Δεδομένων, Ιδιωτικότητα και Ασφάλεια

Ένα από τα σημαντικότερα ζητήματα σε οποιαδήποτε εφαρμογή ανίχνευσης είναι η διαχείριση, το απόρρητο και η ασφάλεια των δεδομένων που συλλέγονται. Το Ευρωπαϊκό Συμβούλιο Προστασίας Δεδομένων εξέδωσε δήλωση σχετικά με τη σημασία της προστασίας των προσωπικών δεδομένων κατά την καταπολέμηση του COVID-19 και επισημασμένων άρθρων του Γενικού Κανονισμού Προστασίας Δεδομένων που παρέχουν τους νομικούς λόγους για την επεξεργασία προσωπικών δεδομένων στο πλαίσιο επιδημιών. Επιπλέον, ορισμένες κυβερνήσεις έχουν θεσπίσει ειδικούς νόμους περί προστασίας της ιδιωτικής ζωής που αποσκοπούν στην αντιμετώπιση ζητημάτων απορρήτου. Για να συμμορφωθούν με αυτές τις απαιτήσεις, οι εφαρμογές παρακολούθησης πρέπει να χρησιμοποιούν πλήθος τεχνικών, στις τρεις διαφορετικές φάσεις της λειτουργίας τους:

i) Εγγραφή, ii) Λειτουργία και iii) Φάσεις αναγνώρισης θετικών περιπτώσεων, ανάλογα με:

• Ποια δεδομένα παράγονται και από ποιον;

• Ποια δεδομένα ανταλλάσσονται ανάμεσα σε ποιον και πότε;

• Ποια δεδομένα αποθηκεύονται πού και από ποιον;

• Ποιος μπορεί να έχει πρόσβαση σε ποιο κομμάτι δεδομένων;

1.2.1 Διαχείριση Δεδομένων

Ο διακομιστής είναι υπεύθυνος για i) την αποθήκευση του PII που συλλέγεται όταν ένας χρήστης εγγράφεται, ii) την δημιουργία, αποθήκευση και μεταφορά των TempIDs σε όλους τους εγγεγραμμένους χρήστες περιοδικά (για παράδειγμα μετά από κάθε 15 λεπτά) και iii) τη διατήρηση μιας λίστας όλων των ατόμων που έχουν διαγνωστεί ως θετικά και τις στενές επαφές τους. Αντίθετα, η συσκευή ενός χρήστη, αφού λάβει το TempID από το διακομιστή, εκτελεί τις ακόλουθες δύο εργασίες: i) δημιουργεί, ανταλλάσσει και αποθηκεύει τις επαφές που είχε με άλλους για ένα καθορισμένο χρονικό διάστημα, συνήθως 21 ημέρες, και ii) κατόπιν αιτήματος, κοινοποιεί τα δεδομένα επαφής που έχει αποθηκεύσει στον διακομιστή με τη συγκατάθεση του χρήστη.

Οι συσκευές χρηστών είναι υπεύθυνες για τη δημιουργία του ωριαίου “seed” και τον υπολογισμό των “chirps” με βάση το “seed” και την τρέχουσα ώρα. Επιπλέον, είναι υπεύθυνοι για την ανταλλαγή και την αποθήκευση αυτών των “chirps”, του RSSI και των ληφθέντων πληροφοριών χρονικής σήμανσης με άλλους. Υπάρχει επίσης η επιλογή για τη συσκευή να αποθηκεύει πρόσθετα μεταδεδομένα, όπως πληροφορίες τοποθεσίας. Ο διακομιστής παίζει περιορισμένο ρόλο σε σύγκριση με την κεντρική αρχιτεκτονική. Αυτός ενεργοποιείται μόνο όταν ένας χρήστης διαγνωστεί ως θετικός στον COVID-19 και ανεβάζει εθελοντικά τα δεδομένα “seeds” του και της χρονικής εγκυρότητας. Αυτά τα δεδομένα, αποθηκευμένα στον διακομιστή, μπορούν τώρα να χρησιμοποιηθούν για αναζήτηση από άλλους χρήστες που έχουν έρθει σε επαφή με τον μολυσμένο χρήστη, ανακατασκευάζοντας τα «chirps” χρησιμοποιώντας τα “seeds”.

Στη φάση λειτουργίας, οι συσκευές αποθηκεύουν όλες τις συναντήσεις ως καταχωρήσεις PET σε δύο διαφορετικούς πίνακες. Ο διακομιστής καταγράφει τα αναγνωριστικά της συσκευής (με κενά μεταδεδομένα, όπως βαθμολογία κινδύνου, εάν έχει σταλεί ειδοποίηση ή όχι κ.λπ.). Ο διακομιστής λαμβάνει μόνο τα PET από χρήστες που έχουν διαγνωστεί θετικοί και εθελοντικά έχουν αποφασίσει να ανεβάσουν αυτές τις πληροφορίες. Μια άλλη σημαντική διαφορά από την αποκεντρωμένη αρχιτεκτονική είναι ότι αυτά τα PET δεν μεταφέρονται σε άλλες συσκευές. Αντίθετα, άλλες συσκευές ανεβάζουν τα PET τους από τον πίνακα ερωτημάτων τους για ανάλυση κινδύνου από τον διακομιστή.

1.2.2 Ιδιωτικότητα

Η επιτυχία οποιασδήποτε εφαρμογής αυτόματης παρακολούθησης επαφών εξαρτάται από διάφορους παράγοντες, όπως: πόσο απρόσκοπτα και με τι ακρίβεια μπορεί να συλλάβει στενές επαφές. Ένας άλλος παράγοντας είναι η εμπιστοσύνη που έχουν οι χρήστες σχετικά με το απόρρητο και την ασφάλειά τους κατά τη χρήση της εφαρμογής. Μια αφελής προσέγγιση για την ανίχνευση επαφών θα μπορούσε να είναι η ανάπτυξη ενός απορρήτου-αγνωστικού συστήματος που διαφημίζει και ανταλλάσσει τους αριθμούς κινητών τηλεφώνων των συμμετεχόντων και καταγράφει περιοδικά την τοποθεσία τους με έναν κεντρικό διακομιστή. Μια τέτοια εφαρμογή θα εγείρει σοβαρές ανησυχίες για το απόρρητο και πιθανότατα δεν θα γίνει αποδεκτή από τους χρήστες. Επομένως, όλες οι αρχιτεκτονικές έχουν ενσωματωμένο σύστημα προστασίας της ιδιωτικής ζωής. Ωστόσο, το ποσό της παρεχόμενης προστασίας διαφέρει σημαντικά και εξαρτάται από τα μοντέλα επίθεσης, τις παραδοχές εμπιστοσύνης και τα μέτρα προστασίας που υιοθετείται από την εκάστοτε λύση.

Από τη σκοπιά του απορρήτου, ταξινομούμε τα δεδομένα που πρόκειται να αποθηκευτούν σε τρεις κατηγορίες: i) PII των συμμετεχόντων (π.χ. ονόματα, αριθμοί τηλεφώνου, εάν έχουν διαγνωστεί θετικοί στον ιό ή όχι κ.λπ.), ii) μηνύματα διαφήμισης επαφών (ψευδώνυμα που ανταλλάσσονται μεταξύ συσκευών) και iii) κοινωνικά γραφήματα/γραφήματα εγγύτητας. Μια ένδειξη δηλαδή των αλληλεπιδράσεων μεταξύ των χρηστών και των ατόμων με τα οποία ήρθαν σε στενή επαφή. Κάθε κατηγορία δεδομένων έχει διαφορετικές επιπτώσεις στο απόρρητο.

Υπάρχουν πολλές επιπτώσεις απορρήτου σε ένα smartphone, καθώς είναι συνήθως λιγότερο ασφαλές από έναν διακομιστή. Σε αυτήν την περίπτωση, επιθέσεις όπως κλοπή, θα έχει ως αποτέλεσμα την αποκάλυψη του περιεχομένου που είναι αποθηκευμένο στο smartphone. Αυτός ο τύπος απειλής υπάρχει σε όλες τις αρχιτεκτονικές. Ωστόσο, η διαφορά μεταξύ των διαφορετικών αρχιτεκτονικών είναι αυτό που είναι αποθηκευμένο στα smartphone. Τα δεδομένα που ενδέχεται να αποθηκευτούν σε συσκευές, όπως λεπτομέρειες των μηνυμάτων συνάντησης, θεωρούνται λιγότερο ευαίσθητα, καθώς αυτές οι πληροφορίες δεν μπορούν να χρησιμοποιηθούν για την άμεση αναγνώριση των επαφών.

Σε μια κεντρική αρχιτεκτονική, οι διακομιστές έχουν πρόσβαση και στους τρεις τύπους δεδομένων. Επομένως, εάν η πρόσβαση στους διακομιστές διακυβεύεται από κακόβουλους χρήστες, θα ήταν δυνατό να προσδιοριστούν όλα τα άτομα και οι επαφές τους, θέτοντας έτσι σε κίνδυνο το απόρρητό τους. Ως εκ τούτου, οι κεντρικές αρχιτεκτονικές πρέπει να παρέχουν επαρκή προστασία των διακομιστών για την εγγύηση του απορρήτου των χρηστών.

Στην αποκεντρωμένη αρχιτεκτονική, όλοι οι χρήστες μπορούν να αποκτήσουν πρόσβαση στον δημόσιο διακομιστή για να κατεβάσουν τη λίστα των “seeds” και να υπολογίσουν τα “chirps” που χρησιμοποιούνται από έναν μολυσμένο χρήστη. Ωστόσο, καθώς αυτά τα “seeds” μεταφορτώνονται μαζί με τις περιόδους λήξης τους, μπορούν να οδηγήσουν σε μη εξουσιοδοτημένη αναγνώριση μολυσμένων ατόμων που χρησιμοποιούν άλλες πληροφορίες κάποιου πλευρικού καναλιού. Για παράδειγμα,

κακόβουλα άτομα / εφαρμογές / οργανισμοί μπορούν να συνεχίσουν να συλλέγουν τα Ephermal IDs και τα “seeds” από τις αναφερόμενες περιπτώσεις και να συνδέσουν τα αναγνωριστικά /τα “chirps με τις προσπελάσιμες βοηθητικές πληροφορίες. Επίσης, καθώς μόνο ένας μολυσμένος χρήστης ανεβάζει “seeds” στο διακομιστή, μια επίθεση ανάλυσης κίνησης, η οποία ξεκίνησε από έναν κακόβουλο χρήστη που μπορεί να κατασκοπεύει, θα μπορούσε να εντοπίσει έναν θετικό χρήστη COVID-19 που ανεβάζει “seeds” στον διακομιστή.

Η υβριδική αρχιτεκτονική υιοθετεί πρόσθετες προηγμένες μεθόδους βελτίωσης της ιδιωτικής ζωής, όπως η κοινή χρήση μυστικών, το Diffie-Hellman (DDH) και η διασταύρωση ιδιωτικών σετ. Γενικά, το μυστικό ενός χρήστη κοινοποιείται από τον χρήστη και τον διακομιστή. Επιπλέον, μέρος της ανάλυσης κινδύνου της λοίμωξης υπολογίζεται στον διακομιστή χρησιμοποιώντας μια μυστική διατήρηση της προστασίας της ιδιωτικής ζωής. Επομένως, εάν ένα συμβαλλόμενο μέρος έχει τεθεί σε κίνδυνο, δεν θα αποκαλυφθεί ολόκληρο το μυστικό ή το αποτέλεσμα της ανάλυσης κινδύνου. Αυτές οι μέθοδοι βελτίωσης του απορρήτου συμβάλλουν στην προστασία της ταυτότητας των μολυσμένων χρηστών από το μήκος κύματος κακόβουλων χρηστών ή παραβιασμένων διακομιστών. Ωστόσο, αυτές οι βελτιωμένες προστασίες απορρήτου εξακολουθούν να μην μπορούν να εμποδίσουν το PII των χρηστών από το να αποκαλυφθεί εάν ένας κακόβουλος χρήστης μπορεί να αποκτήσει επιτυχώς πρόσβαση σε δεδομένα που συλλέγονται από πληροφορίες περιβάλλοντος ενός πλευρικού καναλιού.

1.2.3 Ασφάλεια

Η έννοια της ασφάλειας περιλαμβάνει τον περιορισμό των δυνατοτήτων ενός αντιπάλου να εισάγει ψευδώς αρνητικά και ψευδώς θετικά στοιχεία στο σύστημα, πέρα και από τη διασφάλιση της ακεραιότητας και της διαθεσιμότητας του συστήματος. Το κίνητρο για την επίθεση ποικίλλει και μπορεί να κυμαίνεται από πολιτικό και ιδεολογικό έως οικονομικό. Στο πλαίσιο της ανίχνευσης επαφών, ένας εισβολέας μπορεί να στοχεύει στην εισχώρηση εσφαλμένων καταχωρήσεων ή να προκαλέσει απενεργοποίηση της υπηρεσίας.

Δεδομένου ότι και οι τρεις αρχιτεκτονικές που έχουν περιγραφεί περιλαμβάνουν έναν κεντρικό διακομιστή, είναι σκόπιμο να διερευνηθούν οι συγκεκριμένες απειλές ασφαλείας για καθεμία από τις αρχιτεκτονικές. Η πιθανή απειλή ασφάλειας εξαρτάται από τα δεδομένα που προέρχονται από έναν διακομιστή, ποια δεδομένα κοινοποιούνται και είναι προσβάσιμα σε έναν διακομιστή και με ποια μορφή συλλέγονται και αποθηκεύονται τα δεδομένα (π.χ. ψευδώνυμο, κρυπτογραφημένο, μη κρυπτογραφημένο). Επιπλέον, εξαρτάται και από τον τρόπο λειτουργίας του διακομιστή, δηλαδή εάν είναι i) ένας αξιόπιστος διακομιστής, ii) ένας «ειλικρινής αλλά περίεργος διακομιστής», iii) ένας παραβιασμένος / κακόβουλος διακομιστής, ή iv) ένας «συναδελφικός» διακομιστής.

Ένας κακόβουλος / παραβιασμένος διακομιστής μπορεί να διακόψει όλους τους τύπους επικοινωνιών ή να εισάγει ψευδείς ειδοποιήσεις έκθεσης σε όλες τις αρχιτεκτονικές. Ομοίως, ένας «συναδελφικός διακομιστής» μπορεί να επικοινωνήσει με άλλες κακόβουλες οντότητες για να εκτελέσει την απο-ανωνυμοποίηση των χρηστών.

Στην κεντρική αρχιτεκτονική, ο διακομιστής θεωρείται αξιόπιστος. Είναι υπεύθυνος για την αποθήκευση του PII των χρηστών και τη διαχείριση των κλειδιών ασφαλείας που χρησιμοποιούνται για την κρυπτογράφηση / αποκρυπτογράφηση TempID. Αυτό θέτει τον κίνδυνο κλοπής δεδομένων εάν ο διακομιστής παραβιαστεί, μια γενική απειλή ενάντια σε οποιονδήποτε κεντρικό διακομιστή. Σε αυτό το πλαίσιο, η εφαρμογή διακομιστή πρέπει να εκτελείται σε ένα αξιόπιστο περιβάλλον και να χρησιμοποιεί κατάλληλους μηχανισμούς ελέγχου ταυτότητας και ελέγχου πρόσβασης. Όλες οι πληροφορίες που

ανταλλάσσονται μεταξύ του διακομιστή και του smartphone του χρήστη καθώς και μεταξύ του διακομιστή και των υπευθύνων υγείας πρέπει να είναι εξουσιοδοτημένες και ασφαλείς. Έτσι, οι κεντρικές αρχιτεκτονικές θεωρούν κακόβουλους τους χρήστες στα μοντέλα επίθεσης τους και στοχεύουν να διατηρήσουν τις πληροφορίες όλων των χρηστών ασφαλείς για να αποφευχθεί η απώλεια του απορρήτου των χρηστών. Αυτό διασφαλίζει ότι κανένα κακόβουλο τρίτο μέρος δεν μπορεί να έχει πρόσβαση σε πληροφορίες που αποστέλλονται / λαμβάνονται ή εξαιρούνται πληροφορίες. Ωστόσο, κακόβουλοι χρήστες σε κεντρικές αρχιτεκτονικές θα μπορούσαν να εκμεταλλευτούν τις μη επαληθευμένες πληροφορίες επαφής BLE που ανταλλάσσονται μεταξύ συσκευών για τη διάδοση λανθασμένων πληροφοριών επαφής με τη μετάδοση ή την αναπαραγωγή. Αυτός ο τύπος επίθεσης θα είχε ως αποτέλεσμα ψευδώς θετικά κατά τη διαδικασία ανίχνευσης επαφών, αναγκάζοντας τους χρήστες να ενημερώνονται λανθασμένα ως στενές επαφές.

Οι αποκεντρωμένες και υβριδικές αρχιτεκτονικές, από την άλλη πλευρά, αναλαμβάνουν έναν «ειλικρινές αλλά περίεργο διακομιστή» που εκτελεί όλες τις εργασίες που του έχουν ανατεθεί και συλλέγει παθητικά ευαίσθητα δεδομένα, εάν υπάρχουν. Αυτό το μοντέλο επίθεσης θεωρεί την κυβέρνηση και τον διακομιστή ως αναξιόπιστους και αποκαλύπτει μόνο τις ταυτότητες των χρηστών στις αρχές υγείας. Όπως αναφέρθηκε προηγουμένως, η κύρια ανησυχία του χρήστη σχετίζεται με την κυβέρνηση που μπορεί να χρησιμοποιεί τα δεδομένα και για σκοπούς διαφορετικούς από την ανίχνευση επαφών. Επομένως, αυτές οι αρχιτεκτονικές στοχεύουν στην απόκρυψη των ταυτοτήτων των χρηστών και στη δημιουργία ανώνυμων αναγνωριστικών για τις συσκευές, αποτρέποντας έτσι τη δυνατότητα του διακομιστή να συνδέει αναγνωριστικά με πληροφορίες χρήστη. Η αποκεντρωμένη αρχιτεκτονική μεταβιβάζει τη διαχείριση δεδομένων στα smartphone των χρηστών, καθιστώντας τη λύση πιο ισχυρή έναντι ενός σημείου αποτυχίας / επίθεσης, όπως ο κεντρικός διακομιστής. Ωστόσο, η αποκεντρωμένη αρχιτεκτονική εξακολουθεί να απαιτεί έναν κεντρικό διακομιστή με ελάχιστη λειτουργία. Επομένως, θα είναι ευάλωτο σε έναν πολύ μικρότερο αριθμό επιθέσεων που βασίζονται σε διακομιστές. Σε αποκεντρωμένες αρχιτεκτονικές, ανώνυμα αναγνωριστικά μεταφορτώνονται στον διακομιστή, τα οποία στη συνέχεια είναι πιθανώς προσβάσιμα από άλλα smartphone για αντιστοίχιση. Έτσι, ένας «ειλικρινής αλλά περίεργος διακομιστής» δεν θα μπορεί να μάθει κανένα PII, να συνδέει τα ανώνυμα αναγνωριστικά ή να δημιουργεί κοινωνικά γραφήματα, εκτός εάν έχει πρόσβαση σε κάποιες πληροφορίες καναλιού. Σε περίπτωση παραβίασης δεδομένων, δεν θα υπάρξει καμία επίπτωση, καθώς οι εισβολείς έχουν πρόσβαση μόνο στα “seeds”/”chirps” των μολυσμένων χρηστών, οι οποίοι είναι ήδη δημόσιοι. Ένας κακόβουλος χρήστης, από την άλλη πλευρά, εξακολουθεί να μπορεί να προκαλέσει ψευδώς θετικά στοιχεία μεταβιβάζοντας τα “chirps” και να ξεκινήσει επιθέσεις Denial of Service (DoS), μεταδίδοντας ψεύτικες αλλά σωστά μορφοποιημένες διαφημίσεις.

Η υβριδική αρχιτεκτονική πραγματοποιεί την ανάλυση κινδύνου επαφής και τις διαδικασίες κοινοποίησης στον διακομιστή. Αυτό αποτρέπει τυχόν επιθέσεις επαναπροσδιορισμού / απο-ανωνυμοποίησης. Επιπλέον, η υβριδική αρχιτεκτονική παρέχει πρόσθετους μηχανισμούς για την απόκρυψη των ταυτοτήτων των χρηστών από το διακομιστή, ενώ επιτρέπει την κεντρική αντιστοίχιση των επαφών. Παρόμοια με τις αποκεντρωμένες αρχιτεκτονικές, προτείνει τη δημιουργία Ephemeral IDs στις συσκευές. Η λογική είναι ότι οι συσκευές διατηρούν τον πλήρη έλεγχο των μυστικών τους αναγνωριστικών, καθιστώντας τις λιγότερο ευαίσθητες σε παραβιάσεις στο διακομιστή.

1.3 Υπολογισμός Εγγύτητας

Η πιθανότητα μόλυνσης με COVID-19 αυξάνεται με την παρατεταμένη και στενή επαφή με ένα μολυσμένο άτομο. Ως εκ τούτου, οι εκτιμήσεις της απόστασης (γνωστή ως εγγύτητα) και η διάρκεια της επαφής είναι σημαντικά στοιχεία που χρησιμοποιούνται για την αξιολόγηση της πιθανής εξάπλωσης της μόλυνσης. Οι εφαρμογές ιχνηλάτησης επαφών στοχεύουν στην καταγραφή τέτοιων συναντήσεων. Ένα αυτοματοποιημένο σύστημα εντοπισμού επαφών βασισμένο σε smartphone χρησιμοποιεί κυρίως δύο τεχνολογίες (αισθητήρα): GPS και Bluetooth, για την εκτίμηση εγγύτητας. Το GPS που χρησιμοποιείται σε συστήματα πλοήγησης μπορεί να παρέχει αρκετά ακριβείς πληροφορίες τοποθεσίας εντός του περιθωρίου σφάλματος, ειδικά όταν χρησιμοποιείται σε εξωτερικούς χώρους. Ωστόσο, η αποθήκευση πληροφοριών απόλυτης θέσης συνοδεύεται από κόστος απορρήτου, εάν μεταφερθεί στον διακομιστή. Επιπλέον, το GPS δεν είναι κατάλληλο για εκτίμηση της εγγύτητας σε εφαρμογές COVID για διάφορους λόγους: i) Το GPS έχει χαμηλή απόδοση σε έντονα συνωστισμένους εξωτερικούς χώρους. ii) Το GPS δεν λειτουργεί γενικά σε εσωτερικούς χώρους και, όταν λειτουργεί, παρέχει πολύ χαμηλή ακρίβεια και iii) καταναλώνει ενέργεια γρήγορα και μπορεί να εξαντλήσει γρήγορα την μπαταρία ενός smartphone, εάν χρησιμοποιηθεί για μεγάλο χρονικό διάστημα.

Η διασύνδεση Bluetooth που υπάρχει στα περισσότερα σύγχρονα κινητά τηλέφωνα μπορεί να καταγράψει τις τιμές του δείκτη έντασης ισχύος σήματος (RSSI), βοηθώντας στην εκτίμηση της εγγύτητας. Το ασύρματο σήμα που εκπέμπεται από έναν πομπό αποσυντίθεται / εξασθενεί καθώς ταξιδεύει μέσω του αέρα. Ο δέκτης μπορεί να εκτιμήσει περίπου πόσο μακριά βρίσκεται ο πομπός καταγράφοντας τις τιμές RSSI. Ωστόσο, υπάρχουν πολλά ζητήματα με την εκτίμηση της εγγύτητας μόνο με βάση τις τιμές RSSI. Το ασύρματο σήμα μπορεί να επηρεαστεί από διάφορους παράγοντες εκτός από την απόσταση. Τα αντικείμενα στο περιβάλλον λειτουργίας, όπως έπιπλα, τοίχοι, άνθρωποι κ.λπ. στη διαδρομή μεταξύ ενός αποστολέα και ενός δέκτη επηρεάζουν τη μείωση του σήματος. Άλλα ασύρματα σήματα που χρησιμοποιούν την ίδια συχνότητα μπορεί επίσης να προκαλέσουν παρεμβολές και να εξασθενίσουν περαιτέρω το σήμα. Εκτός αυτού, διαφορετικά κινητά τηλέφωνα μεταδίδουν σήματα Bluetooth με διαφορετικά επίπεδα ισχύος, επηρεάζοντας την εκτίμηση της απόστασης. Τέλος, τα μοτίβα μετάδοσης από το ίδιο τηλέφωνο διαφέρουν ανάλογα με τον προσανατολισμό του τηλεφώνου (συγκεκριμένα την κεραία του) και την παρουσία ή απουσία θήκης τηλεφώνου. Αυτά τα ζητήματα μπορούν να μετριαστούν κάπως με βαθμονόμηση των τιμών RSSI / εξασθένησης σήματος με βάση γνωστές δυνάμεις μετάδοσης.

Ωστόσο, πρέπει να είμαστε σαφείς ότι κάθε στρατηγική μετριασμού που αποσκοπεί στην αντιμετώπιση των προαναφερθέντων αρνητικών επιπτώσεων στην ακρίβεια εγγύτητας μπορεί να έχει περιορισμένο αντίκτυπο μόνο σε πραγματικές συνθήκες καναλιών. Ενώ είναι αλήθεια ότι μπορεί να επιτευχθεί μια τυπική ανάλυση σφάλματος τοποθεσίας σε ένα κανονικό κανάλι σκίασης log ακόμη και παρουσία άγνωστων τιμών εκ των προτέρων για τον εκθέτη απώλειας διαδρομής και τη διακύμανση θορύβου, πρέπει να κατανοήσουμε τους περιορισμούς αυτών των πληροφοριών - θεωρητικές κατασκευές. Οι ισχυρισμοί για «εγγυημένη» ακρίβεια της τάξης του 1 μέτρου από οποιαδήποτε τρέχουσα εφαρμογή θα πρέπει επομένως να εξεταστούν με κάποιο σκεπτικισμό. Επιπλέον, θα πρέπει να σημειώσουμε ότι θα μπορούσαν να υπάρχουν και άλλα μοντέλα καναλιών που περιγράφουν καλύτερα πώς η ισχύς του σήματος ποικίλλει ανάλογα με την απόσταση σε πραγματικά σενάρια. Θα μπορούσαν κάλλιστα να απαιτούνται μοντέλα με περισσότερες από τις δύο ελεύθερες παραμέτρους. Επιπλέον, είναι πολύ πιθανό ότι η χρησιμότητα οποιουδήποτε τέτοιου μοντέλου εξαρτάται τόσο από τον χρόνο όσο και από τη γεωγραφική θέση. Ωστόσο, , θα πρέπει να είναι εφικτό στις μελλοντικές εφαρμογές ανίχνευσης, που είναι χτισμένες σε νέες πλατφόρμες υλικού (παροχή ακρίβειας χρονικού νανοδευτερολέπτου), να παρέχουν πραγματικά ακρίβεια 1m με πολύ υψηλό βαθμό εμπιστοσύνης.

Συνοψίζοντας, με τις τεχνικές που χρησιμοποιούνται από τις τρέχουσες εφαρμογές για την εκτίμηση της εγγύτητας, θα εξακολουθούν να υπάρχουν πολλά ψευδώς θετικά και ψευδώς αρνητικά στοιχεία. Η εκτίμηση εγγύτητας μπορεί να υποδεικνύει στενή επαφή, ενώ η πραγματική επαφή να είναι μακριά ή λανθασμένα να σημειώνεται ότι είναι μακριά όταν βρίσκεται κοντά. Ομοίως, μια στενή επαφή όπως γίνεται αντιληπτή από την εκτίμηση της απόστασης δεν μεταφράζεται πάντα σε έκθεση στον ιό, καθώς μπορεί να υπάρχει τοίχος / απόφραξη μεταξύ των δύο ατόμων (π.χ. δύο γειτονικά διαμερίσματα) ή η επαφή έχει συμβεί σε ανοιχτό χώρο όπου η πιθανότητα μόλυνσης είναι πολύ χαμηλότερη. Ωστόσο, η λήψη ψευδών θετικών δεν είναι τόσο καταστροφική, καθώς οδηγεί μόνο σε πρόσθετα test για αυτές τις ψευδείς περιπτώσεις. Τα ψεύτικα αρνητικά είναι ένα πιο σημαντικό ζήτημα, καθώς θεωρούνται χαμένη ευκαιρία να καταχωρηθεί η επαφή σε μια θετική περίπτωση. Ωστόσο, πιστεύεται ότι τα δεδομένα από αυτές τις εφαρμογές, σε συνδυασμό με άλλες πληροφορίες με βάση τα συμφραζόμενα που αποκτήθηκαν κατά τη διάρκεια μιας συνέντευξης, θα βοηθούσαν τους επαγγελματίες υγείας να λάβουν καλύτερες αποφάσεις.

1.4 Επιθέσεις

Παρακάτω θα καλύψουμε μερικές από τις πιο σημαντικές πιθανές επιθέσεις που μπορούν να εξαπολυθούν εναντίον διαφορετικών αρχιτεκτονικών εφαρμογών ιχνηλάτησης.

1.4.1 Ασύρματη Παρακολούθηση Συσκευής

Ο στόχος του εισβολέα σε αυτόν τον τύπο επίθεσης είναι να παρακολουθεί τη συσκευή από τις πληροφορίες BLE που μεταδίδονται στις εφαρμογές ιχνηλάτησης COVID-19. Για παράδειγμα σε ένα εμπορικό κέντρο για την παρακολούθηση της γενικής κίνησης των πελατών. Ο εισβολέας μπορεί να αναπτύξει κόμβους BLE, όπως τα iBeacons της Apple, στρατηγικά σε ολόκληρο το εμπορικό κέντρο, ακούγοντας παθητικά διαφημίσεις από τις εφαρμογές αυτές. Αυτοί οι κόμβοι μπορούν να στείλουν τα ληφθέντα μηνύματα BLE σε έναν κεντρικό διακομιστή παρακολούθησης για περαιτέρω επεξεργασία. Ο διακομιστής παρακολούθησης μπορεί τώρα να χρησιμοποιεί απλά “chirps” και χρονικές σημάνσεις για να εκτιμήσει τη θέση κάθε συσκευής. Αυτό επιτρέπει την παρακολούθηση, ακόμη και την καταγραφή του χρόνου που περνά κάθε πελάτης (συσκευή) σε κάθε κατάστημα.

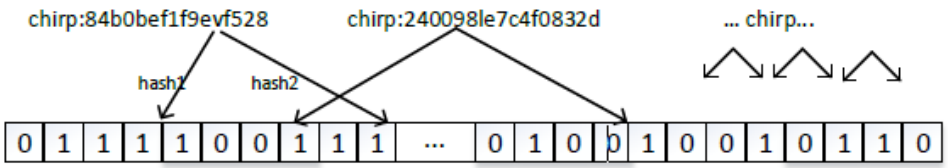
Για εφαρμογές που χρησιμοποιούν την κεντρική αρχιτεκτονική, τα TempID και οι πληροφορίες μοντέλων τηλεφώνου μπορούν να χρησιμοποιηθούν για τον μοναδικό προσδιορισμό μιας συσκευής. Δεδομένου ότι τα TempID αλλάζουν μετά από σύντομο χρονικό διάστημα (συνήθως 10-15 λεπτά), η παρακολούθηση μιας συσκευής πέρα από το σημείο όπου η συσκευή ξεκινά να διαφημίζει ένα νέο TempID θα απαιτούσε επιπλέον ευφυΐα για να συνδέσει τα δύο TempID με την ίδια συσκευή, διαφημίζοντας το ίδιο μοντέλο τηλεφώνου. Στην αποκεντρωμένη αρχιτεκτονική, τα “chirps” με διάρκεια ζωής 1 λεπτού παρέχουν περιορισμένες ευκαιρίες για παρακολούθηση. Ο διακομιστής παρακολούθησης μπορεί ακόμα και να απαριθμεί τον συνολικό αριθμό χρηστών στην περιοχή, ωστόσο είναι δύσκολο να παρακολουθείται η κίνηση μιας συσκευής χωρίς μοντέλο τηλεφώνου. Η παρακολούθηση, σε αυτήν την περίπτωση, θα μπορούσε να εφαρμοστεί σε περιορισμένα σενάρια, π.χ. σε μερικούς πελάτες σε ένα κατάστημα ή εάν η συσκευή του χρήστη βρίσκεται σε στάση. Οι υβριδικές αρχιτεκτονικές συμπεριφέρονται σαν την κεντρική αρχιτεκτονική καθώς οι συσκευές διαφημίζουν το EphID με διάρκεια ζωής 15 λεπτών, καθιστώντας δυνατή την παρακολούθηση μιας συσκευής που βασίζεται σε EphID.

1.4.2 Απόσπαση Τοποθεσίας

Σε αυτήν την επίθεση, ο στόχος των εισβολέων είναι να ανακαλύψει την παρουσία ενός χρήστη σε μια γνωστή τοποθεσία / περιβάλλον, όπως μια γειτονιά. Οι διαφημίσεις BLE και οι πληροφορίες που περιέχονται στην ανταλλαγή μηνυμάτων συνάντησης στην κεντρική αρχιτεκτονική μπορούν να χρησιμοποιηθούν για να επιβεβαιώσουν την τοποθεσία ενός χρήστη. Για παράδειγμα, ας υποθέσουμε ότι η Αλίκη είναι η μόνη στην οικογένειά της που διαθέτει ένα iPhone 9 και αυτό είναι γνωστό σε μία επιτιθέμενο, την Εύα. Η Εύα μπορεί να επιβεβαιώσει εάν η Αλίκη είναι στο σπίτι ακούγοντας τα μηνύματα συνάντησης που περιλαμβάνουν πληροφορίες για το μοντέλο τηλεφώνου της Αλίκης. Ένας απλός τρόπος για να μετριαστεί αυτό το ζήτημα είναι να συμπεριληφθεί και ο αριθμός μοντέλου κινητού στη διαδικασία εγγραφής. Με αυτόν τον τρόπο, ο διακομιστής εξακολουθεί να διαθέτει τις απαιτούμενες πληροφορίες αριθμού μοντέλου για υπολογισμούς εγγύτητας και ο αριθμός μοντέλου τηλεφώνου μπορεί τώρα να αποκλειστεί από τα μηνύματα συνάντησης. Δεν είναι δυνατή η επίθεση επιβεβαίωσης τοποθεσίας σε αποκεντρωμένες ή υβριδικές αρχιτεκτονικές λόγω της χρήσης Ephemeral IDs/”chirps” και της καταστολής πληροφοριών σύνδεσης χρήστη/συσκευής.

1.4.3 Επίθεση Απαρίθμησης

Ο πρωταρχικός στόχος αυτής της επίθεσης είναι να μετρήσει τον αριθμό των χρηστών που έχουν διαγνωστεί θετικοί. Η απαρίθμηση αναφέρεται στην ικανότητα οποιουδήποτε χρήστη να εκτιμήσει τον αριθμό των χρηστών που έχουν προσβληθεί από το COVID-19, οι οποίοι έχουν προσφερθεί εθελοντικά να ανεβάσουν τα δεδομένα ιχνηλάτησης επαφών στον διακομιστή. Να σημειωθεί πως η απαρίθμηση δεν περιλαμβάνει την ικανότητα του διακομιστή να μετρά τον αριθμό των θετικών περιπτώσεων. Στην κεντρική αρχιτεκτονική, οι πληροφορίες σχετικά με τις θετικές περιπτώσεις και τις στενές επαφές τους παραμένουν εντός του διακομιστή, εμποδίζοντας έτσι τους χρήστες να απαριθμήσουν. Στην αποκεντρωμένη αρχιτεκτονική, κάθε θετική περίπτωση ανεβάζει όλα τα “seeds” της από τις τελευταίες 21 ημέρες (21 ημέρες x 24 “seeds” την ημέρα = 504 “seeds”). Όλοι οι χρήστες της εφαρμογής μπορούν να κατεβάσουν τη λίστα όλων των “seeds” από τον διακομιστή και μπορούν να εκτιμήσουν τον αριθμό των θετικών περιπτώσεων. Μία επιλογή για απόκρυψη αυτών των πληροφοριών είναι να υπολογιστούν όλα τα “chirps” στον διακομιστή και να τα αποθηκευτούν σε ένα φίλτρο Bloom. Αυτό το φίλτρο Bloom (βλ. Εικόνα 13) ανακτάται στη συνέχεια από την εφαρμογή για να εκλεχτεί αν ταιριάζει με τα “chirps” της επαφής τους, χωρίς να αποκαλυφθούν άλλες λεπτομέρειες. Η επίθεση απαρίθμησης μπορεί επίσης να μετριαστεί, στην αποκεντρωμένη αρχιτεκτονική, εάν ο μολυσμένος χρήστης έχει τη δυνατότητα να διορθώσει ορισμένα στοιχεία επικοινωνίας ενώ ανεβάζει τις επαφές του. Οι επιθέσεις απαρίθμησης δεν είναι δυνατές στην υβριδική αρχιτεκτονική, καθώς ο διακομιστής αποκρύπτει τη λίστα των μολυσμένων αναγνωριστικών χρήστη από άλλους χρήστες.



Εικόνα 13: Κωδικοποίηση κομματιών σε ένα φίλτρο Bloom

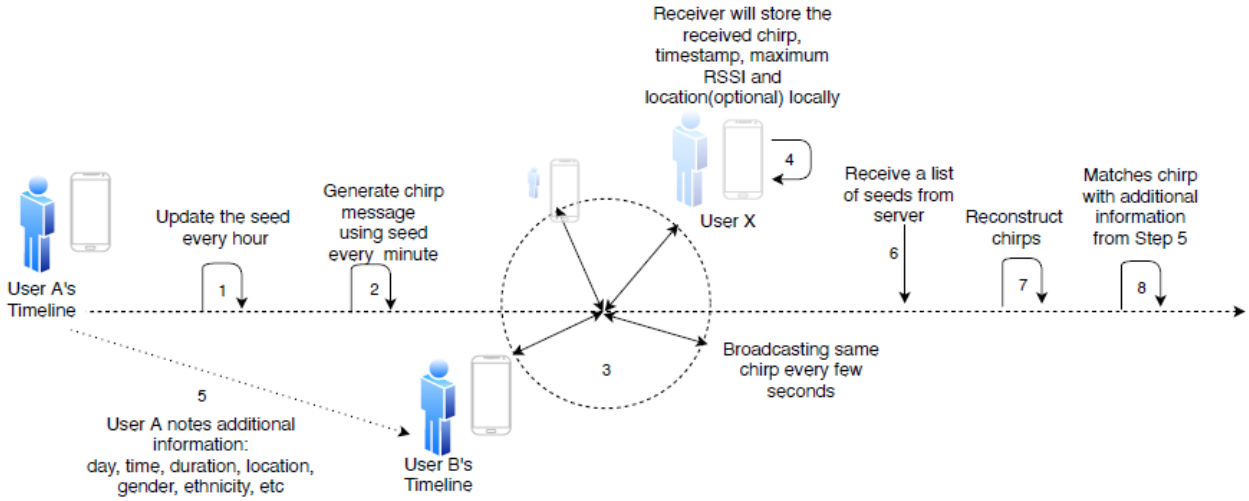
1.4.4 Εύρεση Ταυτότητας Χρήστη

Σε αυτήν την επίθεση, ένας χρήστης στοχεύει στην απο-ανωνυμία της ταυτότητας ενός άλλου χρήστη, συσχετίζοντας τα ανώνυμα δεδομένα εκπομπής με πληροφορίες που συλλέγονται μέσω πλευρικών καναλιών. Αυτό μπορεί να επιτευχθεί συνδέοντας το ανώνυμο αναγνωριστικό με την ταυτότητα του χρήστη σε αυτό που είναι γνωστό ως επίθεση σύνδεσης. Οι περισσότερες εφαρμογές ιχνηλάτησης επαφών έχουν σχεδιαστεί με γνώμονα τα δεδομένα και το απόρρητο των χρηστών. Ωστόσο, στην αποκεντρωμένη αρχιτεκτονική, εξακολουθεί να είναι δυνατή η αναγνώριση των χρηστών όταν αυτοί διαγνωστούν θετικοί στον ιό. Η εικόνα 14 παρουσιάζει τα βήματα που απαιτούνται για την έναρξη της επίθεσης. Ο χρήστης Α χρησιμοποιεί μια αποκεντρωμένη εφαρμογή που καταγράφει τις λεπτομέρειες των συναντήσεων του με άλλα άτομα (ημέρα / ώρα / διάρκεια / τοποθεσία / φύλο κ.λπ.) (βήμα 5). Εάν αυτός ο χρήστης λάβει μια ειδοποίηση (βήμα 6), μπορεί εύκολα να αναγνωρίσει τον μολυσμένο χρήστη συγκρίνοντας τα ανακατασκευασμένα “chirps” (βήμα 7). Αυτό μπορεί να επιτευχθεί εξετάζοντας τη χρονική σφραγίδα (και τη διάρκεια) των “chirps” και συγκρίνοντας τις συλλεγμένες εγγραφές του (βήμα 8). Κάποια κακόβουλη εγγραφή μπορεί επίσης να γίνει αυτόματα από μια τροποποιημένη εφαρμογή που συλλέγει πληροφορίες τοποθεσίας χρησιμοποιώντας GPS / WiFi κ.λπ.

Για την κεντρική αρχιτεκτονική, είναι πιθανό να απο-ανωνυμοποιηθούν οι στενές επαφές, αλλά είναι δύσκολο να απο-ανωνυμοποιηθεί μια θετική περίπτωση, δεδομένου ότι ένας χρήστης της εφαρμογής δεν διαθέτει μια λίστα TempIDs για σύγκριση. Μια θετική περίπτωση μπορεί ακόμα να εντοπιστεί εάν ένας χρήστης που είναι απομονωμένος και έχει συναντήσει μόνο ένα άτομο λαμβάνει μια ειδοποίηση στενής επαφής. Τα TempIDs μπορούν εύκολα να συσχετιστούν με έναν χρήστη, αναφερόμενα στον διαφημιζόμενο αριθμό μοντέλου κινητού. Η διάρκεια της επαφής και μια απομονωμένη συνάντηση θα αυξήσει τις πιθανότητες σύνδεσης των TempID με έναν συγκεκριμένο χρήστη. Παρομοίως, μια επίθεση Sybil μπορεί επίσης να ξεκινήσει, κατά την οποία ένας εισβολέας μπορεί να παρατάξει πολλές συσκευές και να χρησιμοποιήσει μόνο μία συσκευή για μικρό χρονικό διάστημα. Εάν ο χρήστης λάβει μια ειδοποίηση από τον διακομιστή σε μία από τις συσκευές του, τότε μπορεί να περιορίσει την επίθεση σύνδεσης σε ένα σύντομο χρονικό διάστημα, όταν αυτή η συσκευή ήταν ενεργή.

Ένας εισβολέας μπορεί να ξεκινήσει και ένα άλλο είδος επίθεσης σύνδεσης, που ονομάζεται επίθεση Paparazzi σε αποκεντρωμένες εφαρμογές, χρησιμοποιώντας παθητικές συσκευές BLE. Όταν ένας χρήστης διαγνωστεί θετικός στον ιό, ο διακομιστής λαμβάνει τα “seeds”, τα οποία, με τη σειρά τους, αποστέλλονται στους χρήστες, συμπεριλαμβανομένου του εισβολέα. Ο εισβολέας ανακατασκευάζει τα “chirps” και συνδυάζει αυτά τα δεδομένα με εκείνα που λαμβάνονται από τις παθητικές συσκευές BLE. Στη συνέχεια, μπορεί να παρακολουθεί τον χρήστη που είναι θετικός στον ιό καθ 'όλη τη διάρκεια της μετάδοσης. Παρόμοια με την επίθεση Paparazzi, οι επιτιθέμενοι μπορούν να εντοπίσουν μολυσμένους χρήστες, αναπτύσσοντας μεγάλο αριθμό παθητικών συσκευών BLE ενώ συνεργάζονται με τον διακομιστή. Αυτό αναφέρεται ως επίθεση Orwell.

Τα υβριδικά πρωτόκολλα θεωρούνται γενικά μη διασυνδεδεμένα, καθώς δεν μοιράζονται τα PET από μολυσμένους χρήστες με άλλους χρήστες και εκτελούν την ανάλυση κινδύνου και την ειδοποίηση με παρόμοιο τρόπο με την κεντρική αρχιτεκτονική.



Εικόνα 14: Επίθεση σύνδεσης για αποκεντρωμένη αρχιτεκτονική

1.4.5 Κατάχρηση της Εφαρμογής

Ο στόχος αυτού του τύπου επίθεσης είναι να παραπλανήσει την εφαρμογή ιχνηλάτησης με πληροφορίες που παράγονται μέσω εσφαλμένης χρήσης της εφαρμογής. Όλες οι εφαρμογές ιχνηλάτησης, είτε βασίζονται σε κεντρικές, αποκεντρωμένες είτε υβριδικές αρχιτεκτονικές, είναι επιρρεπείς στην κατάχρηση των χρηστών. Αναφέρεται ένα πρόσφατο πείραμα των Χαρτών Google, στο οποίο ένας χρήστης έβαλε 99 κινητά τηλέφωνα σε έναν άδειο δρόμο και οι Χάρτες Google ανταποκρίθηκαν δείχνοντας μεγάλη κυκλοφοριακή συμφόρηση. Ομοίως, μια εφαρμογή ιχνηλάτησης δεν μπορεί να αναγνωρίσει εάν ένα τηλέφωνο μεταφέρεται από τον ιδιοκτήτη του, κάποιον άλλο ή είναι συνδεδεμένο με ένα κατοικίδιο που τρέχει στο πάρκο.

Θα μπορούσαν να εισαχθούν νέα μέτρα για την παράκαμψη αυτών των φαινομένων, όπως η χρήση άλλων υπαρχόντων αισθητήρων στο τηλέφωνο για την καταγραφή δραστηριότητας ή την αναγνώριση βάδισης. Ωστόσο, από τη σκοπιά του απορρήτου, η καταγραφή μεγαλύτερων ποσοτήτων δεδομένων με βάση τα προσωπικά δεδομένα μπορεί να οδηγήσει σε μείωση των εγγυήσεων απορρήτου.

1.5 Κοινές Ανησυχίες των Χρηστών

Οι εφαρμογές ιχνηλάτησης COVID-19 έχουν παρουσιάσει αυξημένη υιοθέτηση από πολλές χώρες. Για παράδειγμα, περισσότερα από 5 εκατομμύρια χρήστες κατέβασαν την εφαρμογή CovidSafe (AU) εντός δύο εβδομάδων από την αρχική της κυκλοφορία της στην Αυστραλία. Ομοίως, οι λήψεις για την ινδική εφαρμογή ανίχνευσης Aarogya Setu έχουν ξεπεράσει το 114 εκατομμύρια. Γενικά, ο αριθμός των λήψεων για μια εφαρμογή χρησιμοποιείται ως ένδειξη αποδοχής από τον χρήστη. Ωστόσο, υποστηρίζουμε ότι αυτό δεν αποτελεί επαρκή μέτρηση για τον υπολογισμό του αντίκτυπου ή της αποτελεσματικότητας μιας εφαρμογής ιχνηλάτησης επαφών. Απαιτούνται πρόσθετες πληροφορίες, όπως ο αριθμός των στενών επαφών που προσδιορίζονται μέσω των δεδομένων που καταγράφονται από μια εφαρμογή και των σχετικών ψευδών θετικών / ψευδών αρνητικών τιμών. Συνήθως, οι υπεύθυνοι υγείας αναλαμβάνουν τον

χειροκίνητο εντοπισμό επαφών σε συνδυασμό με τα δεδομένα της εφαρμογής για να ανακαλύψουν τις στενές επαφές για μια εντοπισμένη περίπτωση. Υπάρχουν τρεις διαφορετικές δυνατότητες από την άποψη αυτή: α) Οι επαφές που προσδιορίζονται από την εφαρμογή ταιριάζουν ακριβώς με τη διαδικασία μη αυτόματης ανίχνευσης, άρα η εφαρμογή πέτυχε τον στόχο της και τα δεδομένα χρησιμοποιούνται ως επιβεβαίωση. β) Οι επαφές που προσδιορίζονται από την εφαρμογή είναι περισσότερες από αυτές που προσδιορίζονται από τη διαδικασία μη αυτόματης ανίχνευσης, άρα η εφαρμογή πέτυχε τον στόχο της και απέδειξε την αποτελεσματικότητά της. γ) Οι επαφές που προσδιορίζονται από την εφαρμογή είναι μικρότερες από αυτές που προσδιορίζονται από τη διαδικασία μη αυτόματης ανίχνευσης, άρα η απόδοση της εφαρμογής είναι αμφισβητήσιμη.

Οι δημιουργοί έχουν εκφράσει πολλές ανησυχίες που σχετίζονται με τη χρήση αυτών των εφαρμογών ιχνηλάτησης επαφών. Ωστόσο, παρακάτω παρατίθενται οι αντίστοιχες ανησυχίες από την οπτική γωνία ενός χρήστη.

1) *Χρήση μπαταρίας*

Η υπερβολική κατανάλωση μπαταρίας είναι ένα επαναλαμβανόμενο πρόβλημα για εφαρμογές σε έξυπνα κινητά. Η κατανάλωση μπαταρίας επηρεάζεται από πολλούς παράγοντες, όπως τη χρήση του επεξεργαστή, τη συχνότητα, τη κλίμακα της διαχείρισης δεδομένων και τον αριθμός των ανταλλαγών μηνυμάτων κ.λπ., καθώς και τις τακτικές συνδέσεις στο δίκτυο κινητής τηλεφωνίας ή Wifi για την επικοινωνία με τους διακομιστές. Το πρωτόκολλο BLE επιτρέπει στην εφαρμογή να ανταλλάσσει περιοδικά μια μικρή ποσότητα δεδομένων με άλλους χρήστες. Από την άλλη πλευρά, η επικοινωνία με τον διακομιστή βασίζεται σε παραδοσιακά πρωτόκολλα ασφαλούς εφαρμογής, π.χ. HTTPs. Ο κύριος αντίκτυπος στη χρήση της μπαταρίας για αυτά τα πρωτόκολλα σχετίζεται με τον αριθμό των ανταλλαγών πληροφοριών με τον διακομιστή. Για τις εφαρμογές που βασίζονται σε μια κεντρική αρχιτεκτονική, ένα μήνυμα σταθερού μεγέθους ανακτάται περιοδικά από τον διακομιστή για να λαμβάνει ένα νέο TempID.

Αντίθετα, για την αποκεντρωμένη έκδοση, δεν υπάρχει περιοδική ανάκτηση δεδομένων κατά το στάδιο λειτουργίας. Τα δεδομένα λαμβάνονται μόνο όταν ο διακομιστής δημοσιεύσει τα “seeds” που ανέβηκαν για μια θετική περίπτωση. Ωστόσο, η εφαρμογή ελέγχει για οποιαδήποτε νέα λήψη μετά από κάθε 24 ώρες. Άρα, το ποσοστό μεταφόρτωσης των αποκεντρωμένων εφαρμογών είναι χαμηλότερο από τις αντίστοιχες εφαρμογές κεντρικών αρχιτεκτονικών. Στις αποκεντρωμένες περιπτώσεις, η μεταφόρτωση αποτελείται από όλα τα “seeds” που χρησιμοποιήθηκαν κατά τις τελευταίες 21 ημέρες, ενώ οι κεντρικές εφαρμογές ανεβάζουν όλα τα μηνύματα συνάντησης που έχουν ληφθεί τις τελευταίες 21 ημέρες. Οι κεντρικές εφαρμογές αποδίδουν καλύτερα όσον αφορά την επεξεργασία στις συσκευές, καθώς αυτό συνεπάγεται ελάχιστη επεξεργασία σε σύγκριση με τις αποκεντρωμένες εφαρμογές που πρέπει να δημιουργούν και να συντηρούν “seeds” (μετά από κάθε 1 ώρα) και τα “chirps” (μετά από κάθε 1 λεπτό) Για την υβριδική αρχιτεκτονική, οι συσκευές δημιουργούν EphIDs καθώς και δύο PET για κάθε λαμβανόμενο EphID. Οι υβριδικές εφαρμογές ανεβάζουν τον υψηλότερο όγκο δεδομένων στο διακομιστή, δηλαδή: τα PET από αναγνωρισμένα θετικά κρούσματα και τα Query PET από όλους τους άλλους χρήστες, που συλλέχθηκαν με σκοπό τον έλεγχο της κατάστασης κινδύνου. Από την άλλη πλευρά, η λήψη από τον διακομιστή είναι χαμηλότερη σε σύγκριση με άλλες αρχιτεκτονικές.

Η κατανάλωση μπαταρίας επηρεάζεται επίσης και από την άποψη εκτέλεσης, καθώς μια εφαρμογή που εκτελείται στο προσκήνιο απαιτεί περισσότερη ισχύ από μια εφαρμογή που εκτελείται στο παρασκήνιο. Συνήθως, αυτή η επιλογή σχεδιασμού εξαρτάται από την υποστήριξη του λειτουργικού συστήματος για αυτές τις εφαρμογές. Για παράδειγμα, οι εφαρμογές CovidSafe και TraceTogether IoS αντιμετωπίζουν αυτά τα ζητήματα όταν η εφαρμογή εκτελείται στο παρασκήνιο. Η Google και η Apple, οι δύο κορυφαίοι πάροχοι λειτουργικών συστημάτων για smartphone, συνεργάστηκαν για την παροχή API ειδοποιήσεων έκθεσης που βελτιώνουν την

ενσωμάτωση των εφαρμογών ιχνηλάτησης με το λειτουργικό σύστημα. Αυτά τα API αναμένεται να βελτιώσουν τη διαδικασία ανάπτυξης αυτών των εφαρμογών και να μειώσουν την κατανάλωση ενέργειας.

2) *Συμβατότητα εκδόσεων λειτουργικού συστήματος και διαφορετικών εφαρμογών*

Η ανάπτυξη μιας εφαρμογής που λειτουργεί σε όλα τα μοντέλα smartphone δεν είναι ασήμαντη εργασία. Τα smartphone διαθέτουν διαφορετικές εκδόσεις λειτουργικού συστήματος (OS), ενώ το Android και το iOS είναι τα δύο κυρίαρχα λειτουργικά συστήματα. Οι περισσότερες από τις εφαρμογές ιχνηλάτησης έχουν αναπτυχθεί για τις νεότερες εκδόσεις λειτουργικού συστήματος. Για παράδειγμα, τόσο το TraceTogether όσο και το CovidSafe απαιτούν έκδοση iOS 10 ή μεταγενέστερη. Το TraceTogether απαιτεί Android 5.1 ή μεταγενέστερη έκδοση, ενώ το CovidSafe λειτουργεί σε Android 6.0 ή μεταγενέστερη έκδοση. Το CovidSafe απαιτεί τις νεότερες εκδόσεις λειτουργικού συστήματος για λόγους ασφαλείας και βελτιωμένες δυνατότητες Bluetooth.

Ένα δευτερεύον πρόβλημα είναι η συμβατότητα μεταξύ εφαρμογών. Για παράδειγμα σε μια περίπτωση κατά την οποία ένας χρήστης μιας εφαρμογής έχει κυκλοφορήσει σε μια συγκεκριμένη γεωγραφική περιοχή σε ένα ταξίδι, όπου η περιοχή αυτή είναι διαφορετική από αυτή για την οποία και έχει αναπτυχθεί η εκάστοτε εφαρμογή. Επιπλέον, δεν είναι σαφές πώς θα συμπεριφερόταν μια εφαρμογή εάν μια δεύτερη εφαρμογή ανίχνευσης είναι εγκατεστημένη στην ίδια συσκευή λόγω των αρχιτεκτονικών διαφορών που συζητήθηκαν νωρίτερα.

3) *Διαφάνεια*

Υπάρχει μια πραγματική ανησυχία των χρηστών σχετικά με τη φύση των πληροφοριών που συλλέγονται από το smartphone τους και τη χρήση τους από διάφορα μέρη. Υπάρχουν δύο βασικές προσεγγίσεις για την επίτευξη διαφάνειας. Η πρώτη επιλογή είναι να είναι ανοιχτός ο πηγαίος κώδικας (open source) της εφαρμογής. Η δημοσίευση του πηγαίου κώδικα της εφαρμογής βελτιώνει τη διαφάνεια και την εμπιστοσύνη στο σύστημα, καθώς τα εφαρμοσμένα χαρακτηριστικά απορρήτου και ασφάλειας μπορούν να ελεγχθούν από την ερευνητική και ακαδημαϊκή κοινότητα. Αν και αυτή είναι η προτιμώμενη επιλογή, ο δημόσιος πηγαίος κώδικας δεν αποτελεί πανάκεια για την εξασφάλιση της ασφάλειας. Πολλοί κίνδυνοι προκύπτουν από τις διαμορφώσεις του συστήματος και τη χρήση ενός ευρύτερου συστήματος που δεν είναι εύκολα παρατηρήσιμο από την ανάλυση του πηγαίου κώδικα. Θα ήταν επίσης χρήσιμο να διασφαλιστεί ότι ο κώδικας υπόκειται σε περιοδικό έλεγχο και αξιόπιστους ελέγχους τρίτων.

Ενώ η διαφάνεια είναι το κλειδί για την ευρύτερη υιοθέτηση από τους τελικούς χρήστες, είναι σημαντικό να σημειωθεί ότι τελικά απαιτείται και ένας βαθμός εμπιστοσύνης στη χρήση οποιασδήποτε εφαρμογής για κινητά. Αυτό περιλαμβάνει την εμπιστοσύνη στους προγραμματιστές, την ανεξάρτητη ομάδα δοκιμών και επαλήθευσης, τους χειριστές και τους ιδιοκτήτες της υπηρεσίας, και το σημαντικότερο είναι οι να εταιρείες παρέχουν βασικά στοιχεία όπως τα λειτουργικά συστήματα κινητών τηλεφώνων.

Τέλος, όλες οι λειτουργικές εφαρμογές θα πρέπει να συνοδεύονται από αξιολόγηση απορρήτου (PIA). Μεταξύ της πλειονότητας των εφαρμογών εφαρμογών που υπάρχουν αυτήν τη στιγμή, μόνο το CovidSafe (AU) και το DP-3T συνοδεύονται από PIA.

Κεφάλαιο 2° :

Η Android εφαρμογή "CoronaTracker" της μεταπτυχιακής εργασίας

Στα πλαίσια της παρούσας μεταπτυχιακής εργασίας, αναπτύξαμε και εμείς μια δική μας υλοποίηση μιας εφαρμογής Android, γραμμένη σε **Kotlin**, ιχνηλάτησης των επαφών των κρουσμάτων του COVID-19, που ακούει στο όνομα "**CoronaTracker**".

Η εφαρμογή μας ανήκει στην κατηγορία της **κεντρικής αρχιτεκτονικής συστήματος** μιας και δεν διατηρούμε τίποτα στην συσκευή του χρήστη, πέρα από ένα **τυχαίο** αλλά **μοναδικό** για κάθε χρήστη **αναγνωριστικό ID**.

Όλες οι πληροφορίες σχετικά με τις επαφές διατηρούνται πλήρως από την **firebase**, αλλά συνάμα έχουμε σεβαστεί σε πολύ μεγάλο βαθμό το ζήτημα της **ιδιωτικότητας** και της **ασφάλειας** και ως αποτέλεσμα συλλέγουμε το μικρότερο δυνατό ποσοστό των απαραίτητων πληροφοριών.

Ειδικότερα, τα δεδομένα που διατηρούνται στην firebase είναι κυρίως μόνο τα αναγνωριστικά των χρηστών, αν έχουν μολυνθεί ή έρθει σε επαφή με μολυσμένο άτομο και όλες οι επαφές τους με άλλους χρήστες (μέσω των αντίστοιχων αναγνωριστικών αυτών) μαζί με τις χρονικές σημάνσεις αυτών και το γεωγραφικό στίγμα όπου και έλαβαν χώρα.

Αναφορικά με το γεωγραφικό στίγμα, για να εξασφαλίσουμε ότι θα πάρουμε στίγμα σχεδόν σε κάθε περίπτωση, αρχικά ελέγχουμε τους αισθητήρες GPS, εάν δεν πάρουμε σήμα, τότε προσπαθούμε να λάβουμε συντεταγμένες μέσω του δικτύου κινητής τηλεφωνίας. Αν μετά από 20 δευτερόλεπτα δεν λάβουμε στίγμα με κανέναν από τους παραπάνω τρόπους, τότε προσπαθούμε να λάβουμε το τελευταίο στίγμα που είχε καταχωρηθεί από την συσκευή του χρήστη.

2.1. Στόχος της Εφαρμογής

Η εφαρμογή μας έχει έναν πολύ απλό αλλά συνάμα πολύ σημαντικό και χρήσιμο σκοπό, δηλαδή το να μπορεί να καταγράψει με μια ικανοποιητική αξιοπιστία όλες τις επαφές που μπορεί να έχει στη μέρα του ένα άτομο με άλλα άτομα, έτσι ώστε μετέπειτα, να διευκολύνει σε σημαντικό βαθμό την ιχνηλάτηση των πιθανών κρουσμάτων του COVID-19.

Παραδοσιακά, εάν κάποιο άτομο διαγνωστεί θετικό στον COVID-19, οι αρμόδιες αρχές θα εκκινήσουν μια διαδικασία «χειροκίνητης» ιχνηλάτησης των επαφών αυτού. Αυτή η διαδικασία βασίζεται πρωτίστως στο να μπορέσει το μολυσμένο άτομο ή και οι οικείοι αυτού, να παρέχουν χρήσιμες πληροφορίες που θα μπορούσαν να οδηγήσουν και σε άλλα πιθανώς μολυσμένα άτομα. Όπως καταλαβαίνουμε όμως, αυτή αποτελεί μια χρονοβόρα και «επίπονη» διαδικασία η οποία μπορεί και ταυτόχρονα να οδηγήσει σε πολύ αναξιόπιστα δεδομένα. Επιπλέον, πρέπει να ληφθεί υπόψη, ότι στην παρούσα φάση της πανδημίας αυτής, πέρα από το επιπλέον χρήμα που μια «χειροκίνητη» διαδικασία ιχνηλάτησης χρειάζεται, απαιτεί επίσης και χρόνο, ο οποίος αποτελεί τροχοπέδη για τις γρήγορες αντιδράσεις που χρειάζονται.

Αντιθέτως, με την συμβολή της σύγχρονης τεχνολογίας, μια τέτοια εφαρμογή θα μπορούσε να συνδράμει δραματικά στη βελτίωση των αντανακλαστικών των αρμόδιων αρχών, μιας και ο απαιτούμενος χρόνος για την ιχνηλάτηση των κρουσμάτων θα ήταν δραματικά μικρότερος.

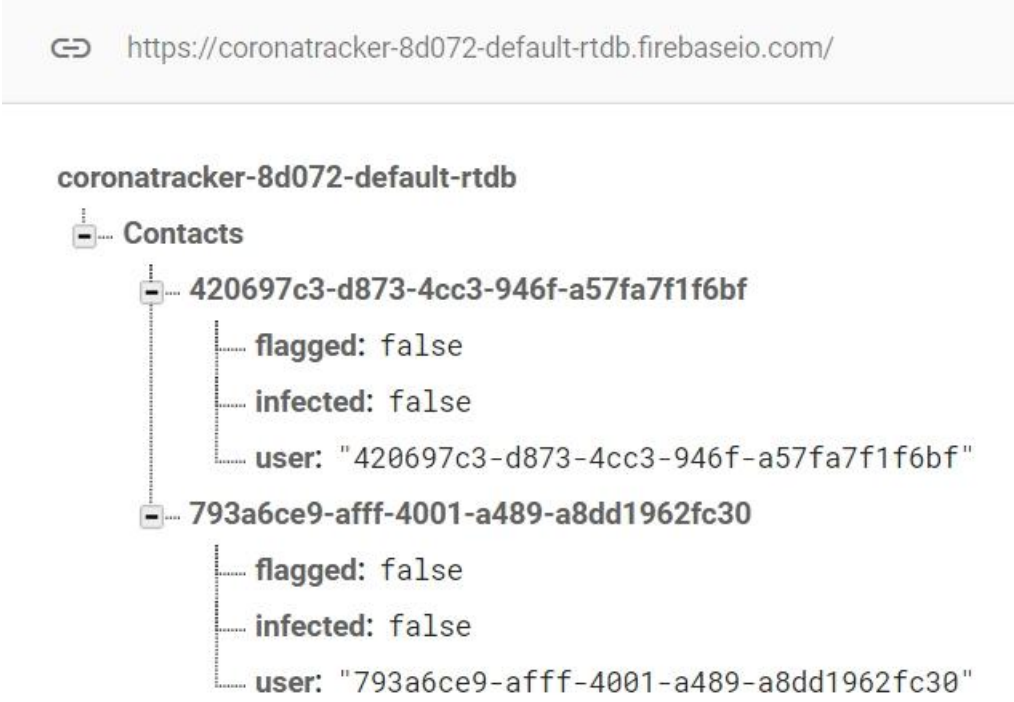
2.2 Τρόπος Λειτουργίας της Εφαρμογής

Ο τρόπος λειτουργίας της εφαρμογής μας είναι αρκετά απλός αλλά ταυτόχρονα και αποτελεσματικός. Κάναμε χρήση της τεχνολογίας **Bluetooth** (και ειδικότερα **BLE, Bluetooth Low Energy** με στόχο την όσο το δυνατόν μικρότερη κατανάλωση ενέργειας) έτσι ώστε να μπορούμε να εντοπίζουμε τις κινητές συσκευές που βρίσκονται κοντά μας.

Ειδικότερα, αξιοποιώντας τις δυνατότητες της αξιόπιστης βιβλιοθήκης "Close To Me" , μπορούμε να πραγματοποιήσουμε τον εντοπισμό και την καταγραφή των άλλων συσκευών που ο χρήστης έρχεται σε επαφή σε απόσταση εντός **2 μέτρων.**

Αρχικά, με το που εγκαθιστά για πρώτη φορά ο χρήστης την εφαρμογή "CoronaTracker", λαμβάνει ένα τυχαίο αλλά μοναδικό αναγνωριστικό ID το οποίο και θα τον διακρίνει από τους υπόλοιπους χρήστες. Ταυτόχρονα, γίνεται και η πρώτη εγγραφή αυτού στην Firebase.

Εικόνα 15: Πρώτη εγγραφή ενός χρήστη στην εφαρμογή

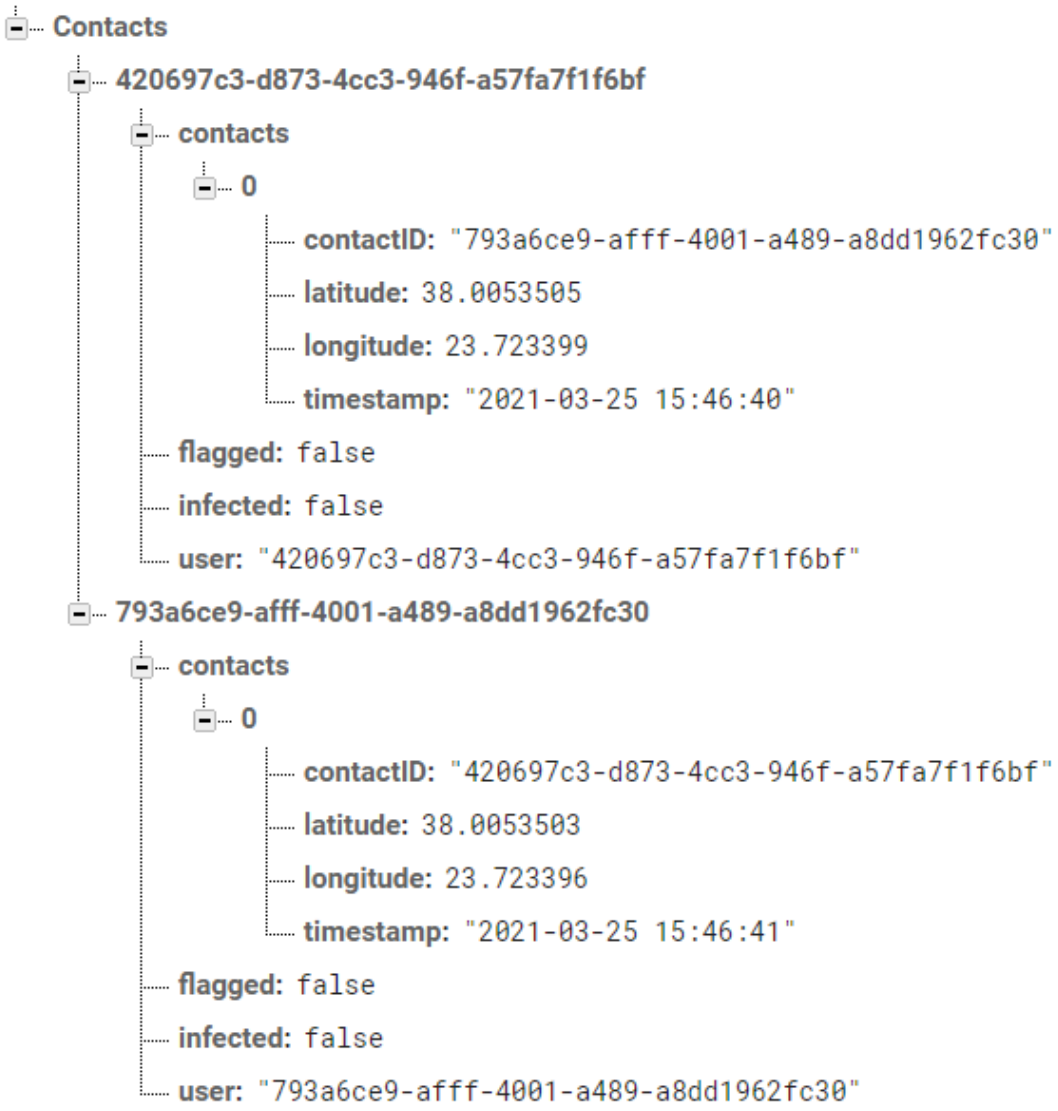


Εικόνα 16: Πρώτη εγγραφή και ενός άλλου χρήστη στην εφαρμογή

Στις παραπάνω εικόνες, παρατηρούμε ακριβώς το τι καταγράφεται σε μια πρώτη εγγραφή ενός χρήστη. Συγκεκριμένα:

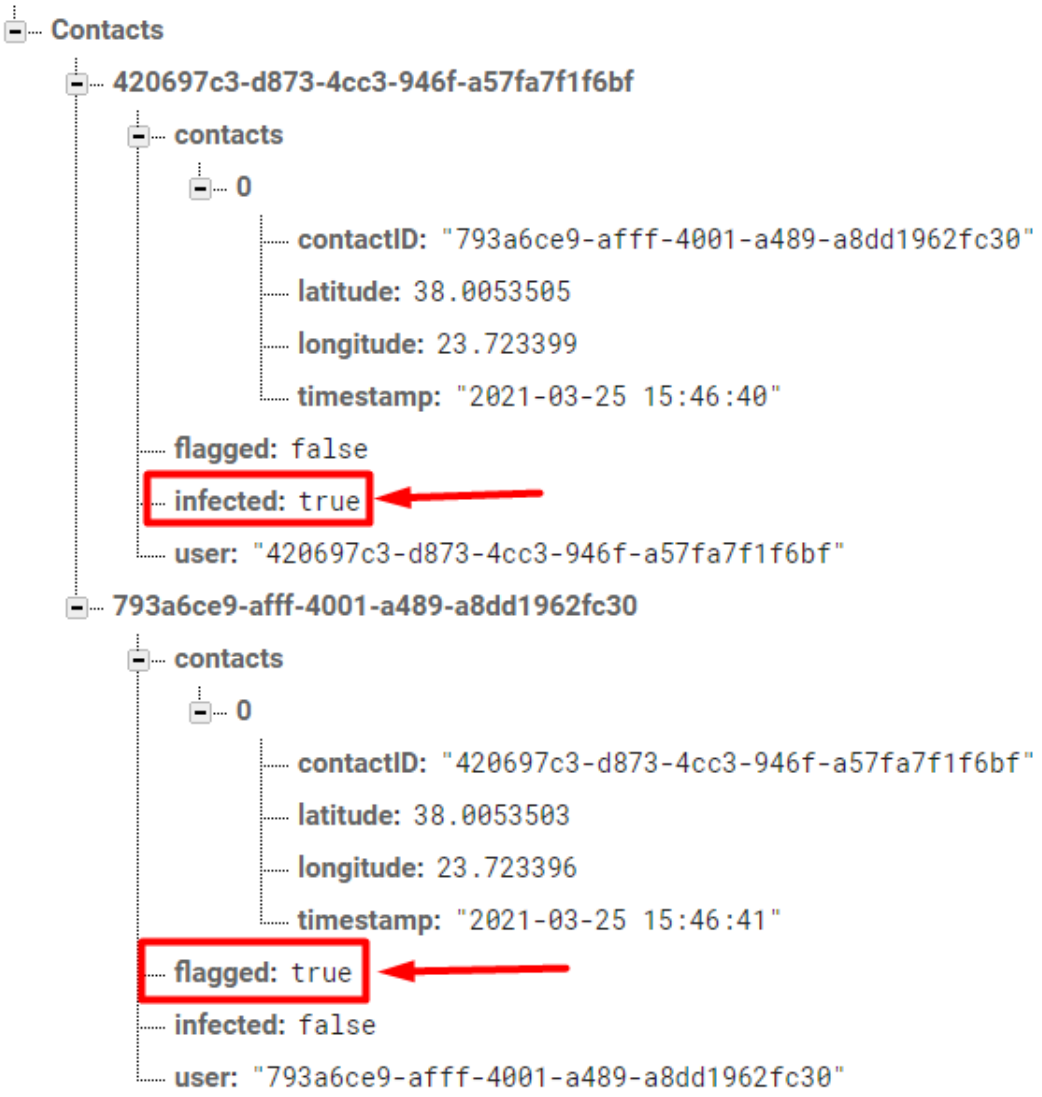
- το πεδίο “**user**” αποτελεί το τυχαίο μοναδικό αναγνωστικό που λαμβάνει ο χρήστης
- το πεδίο “**flagged**” αποτελεί το εάν ο χρήστης έχει έρθει σε επαφή με μολυσμένο άτομο
- το πεδίο “**infected**” αποτελεί το εάν ο χρήστης έχει διαγνωστεί θετικός στον COVID-19

Στη συνέχεια, όταν η εφαρμογή βρίσκεται σε mode εντοπισμού (*περαιτέρω λεπτομέρειες σχετικά με την χρήση της ίδιας της εφαρμογής θα καλυφθούν στην παράγραφο **4.3***), σκανάρει συνεχώς για άλλους χρήστες που και αυτοί διαθέτουν την εφαρμογή μας και βρίσκονται στην ίδια λειτουργία. Έτσι, εάν εντοπιστεί κάποια άλλη τέτοια συσκευή εντός **2 μέτρων**, τότε θα πραγματοποιηθεί και καταγραφή της επαφής αυτής για τους εκάστοτε χρήστες.



Εικόνα 17: Καταγραφή της επαφής δύο χρηστών

Ταυτόχρονα, η εφαρμογή μας πραγματοποιεί στο **background** συνεχώς και **ανά 1 ώρα ελέγχους** στην Firebase για το αν ο χρήστης μας έχει έρθει σε επαφή με άλλο μολυσμένο άτομο (infected) ή με άτομο το οποίο με τη σειρά του έχει έρθει επαφή με άλλο μολυσμένο άτομο (flagged). (Οι έλεγχοι πραγματοποιούνται εσκεμμένα κατά 1 ώρα, ώστε να μην καταναλώνεται μεγάλο ποσοστό μπαταρίας, αλλά εκτός από κάθε μία ώρα, πραγματοποιείται και άλλος ένας έλεγχος, **5 δευτερόλεπτα** μετά από κάθε άνοιγμα της εφαμοργής.) Εάν πραγματοποιηθεί κάτι τέτοιο, τότε ο χρήστης **ενημερώνεται** σχετικά και η βάση ενημερώνεται επίσης, αλλάζοντας το αντίστοιχο **state** τόσο του δικού μας χρήστη όσο και των άλλων που ήρθαν σε επαφή με αυτόν. Με αυτόν τον τρόπο θα ενημερωθούν και οι άλλοι χρήστες καταλλήλως μιας και ο έλεγχος των 5 δευτερολέπτων εννοείται πως πραγματοποιείται και στις δικές τους συσκευές.



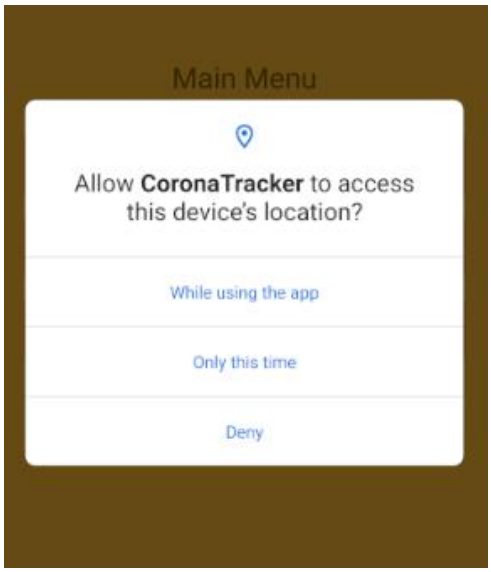
Εικόνα 18: Ο άνω χρήστης διαγνώστηκε θετικός και άλλαξε το state και του κάτω

Στην παραπάνω εικόνα, παρατηρούμε το τι ακριβώς συμβαίνει και στην Firebase, εάν ένας χρήστης διαγνωστεί θετικός και ως συνέπεια αλλάξει και το state ενός άλλου χρήστη με τον οποίο και ήρθε σε επαφή. Να σημειωθεί πως για να γίνει flagged ένας άλλος χρήστης και να ενημερωθεί, **απαραίτητη προϋπόθεση** είναι η επαφή με κάποιον μολυσμένο χρήστη να έχει πραγματοποιηθεί εντός **14 ημερών**.

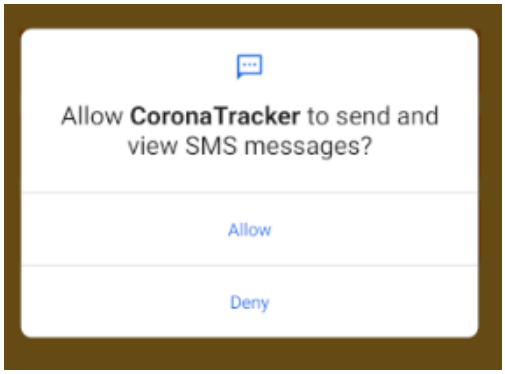
Αναφορικά, με το πως μπορεί να τεθεί ένας χρήστης ο θετικός στον COVID-19 στην συγκεκριμένη εφαρμογή, πραγματοποιείται συνεχώς έλεγχος στα **SMS** του χρήστη και εάν ληφθεί κάποιο μήνυμα από τον **ΕΟΔΥ** που περιλαμβάνει την λέξη **"POSITIVE"**, τότε αυτόματα ο χρήστης ορίζεται ως θετικός και πραγματοποιούνται οι αντίστοιχες προαναφερθείσες ενέργειες. Ο λόγος που έγινε χρήση της συγκεκριμένης μεθοδολογίας, είναι διότι δεν απαιτείται η ανάπτυξη κάποιου αντίστοιχου back-end server από την πλευρά του ΕΟΔΥ και επιπλέον με αυτόν τον τρόπο ο ΕΟΔΥ ενημερώνει τους πολίτες για τα αποτελέσματα των test τους για COVID-19.

2.3 Παρουσίαση της Εφαρμογής

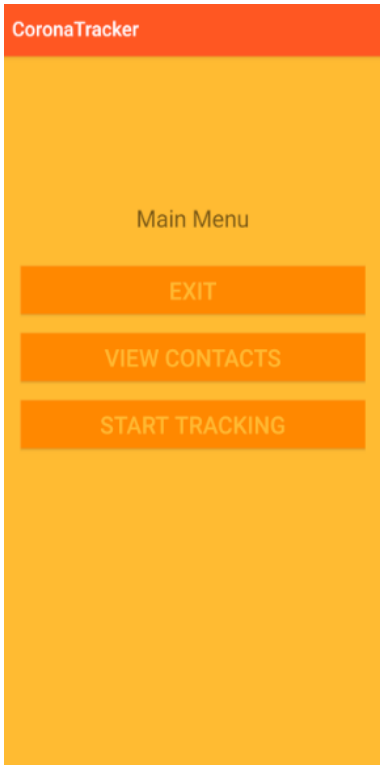
Ανοίγοντας την εφαρμογή “CoronaTracker” αντικρίζουμε το κεντρικό μενού της. Εάν πραγματοποιούμε πρώτο άνοιγμα, τότε θα μας ζητηθούν να παραχωρήσουμε πρώτα και τα κατάλληλα permissions που απαιτούνται για την ορθή λειτουργία αυτής (το GPS απαιτείται από το android για να μπορούμε να σκανάρουμε για Bluetooth συσκευές).



Εικόνα 19: Permission Dialog για τοποθεσία



Εικόνα 20: Permission Dialog για SMS

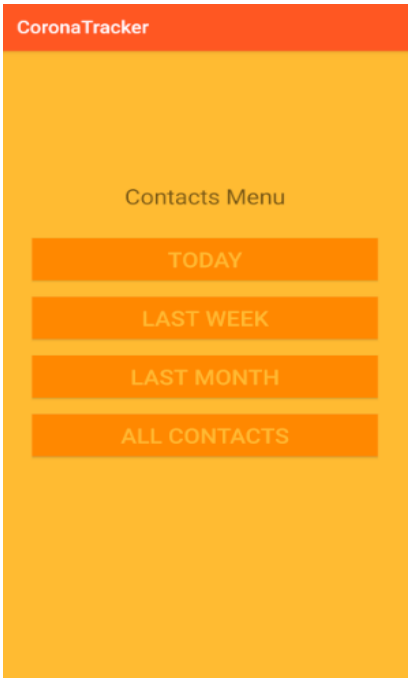


Εικόνα 21: Κεντρικό μενού εφαρμογής

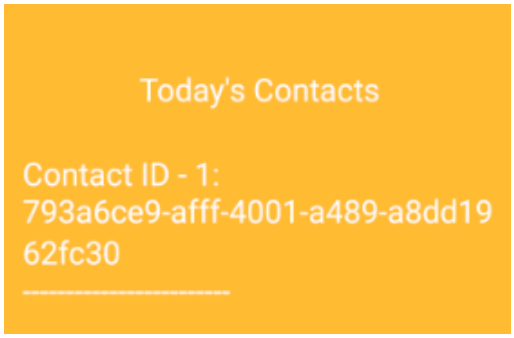
Εν συνεχεία, έχουμε 3 βασικές επιλογές:

- **EXIT:** Έξοδος από την εφαρμογή
- **VIEW CONTACTS:** Προβολή των επαφών μας με άλλους χρήστες
- **START TRACKING:** Μετάβαση στο mode ιχνηλάτησης

Η λειτουργία "**VIEW CONTACTS**" μας δίνει την δυνατότητα να δούμε τις επαφές τις οποίες και έχουμε πραγματοποιήσει και ειδικότερα μπορούμε να τις δούμε και **φιλτραρισμένες**. Ειδικότερα, μπορούμε να δούμε είτε μόνο τις σημερινές, είτε τις προηγούμενης εβδομάδας, είτε του προηγούμενου μήνα αλλά και όλες τις επαφές που είχαμε καταγράψει.

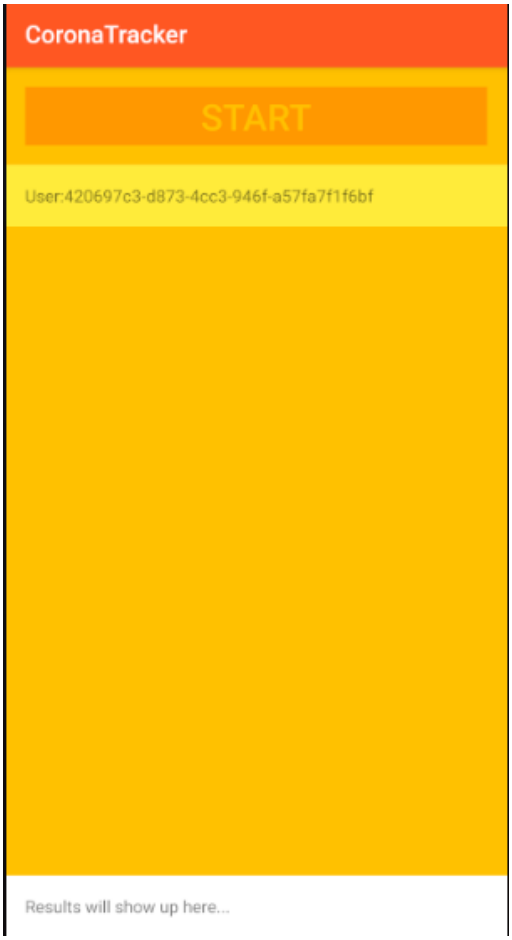


Εικόνα 22: Μενού προβολής επαφών



Εικόνα 23: Προβολή σημερινών επαφών

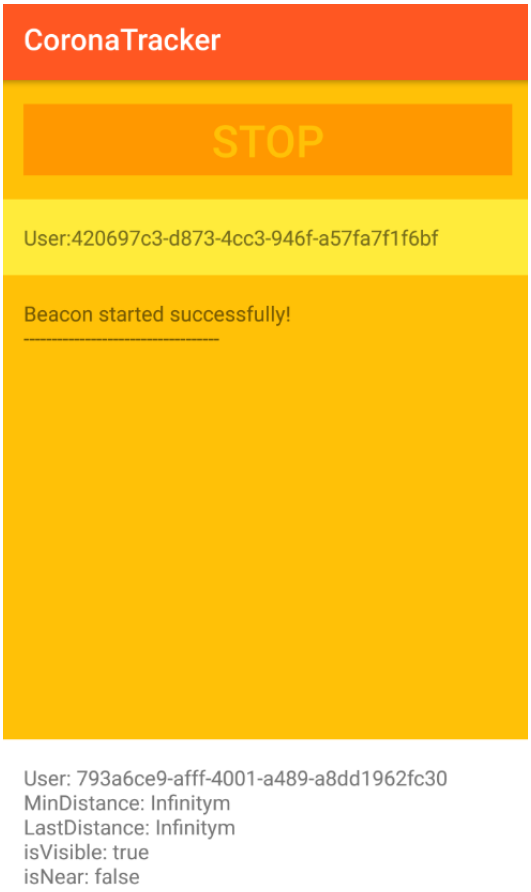
Η λειτουργία “**START TRACKING**” μας οδηγεί στην οθόνη του mode ιχνηλάτησης επαφών.



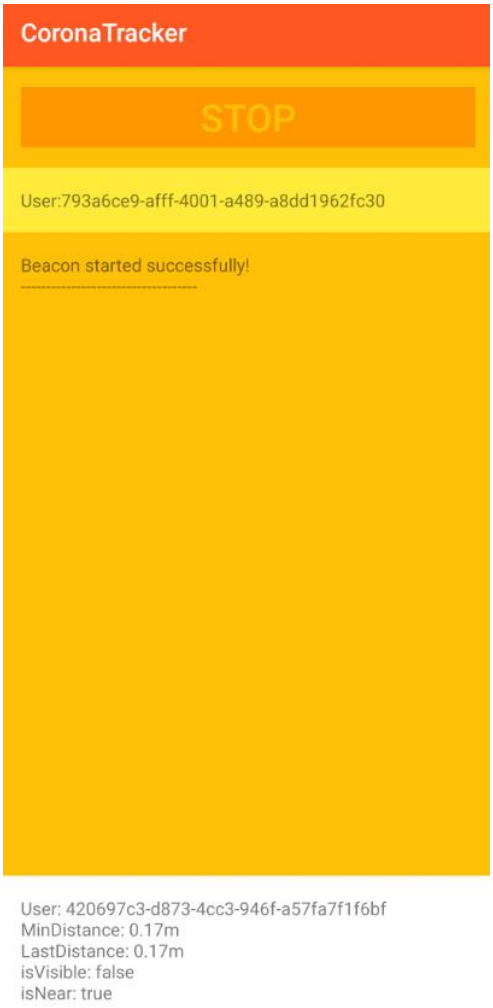
Εικόνα 24: Λειτουργία ιχνηλάτησης επαφών

Στο επάνω μέρος έχουμε το κουμπί “START” με το οποίο και μπορούμε να εκκινήσουμε την λειτουργία ιχνηλάτησης. Επίσης μπορούμε να δούμε και το μοναδικό ID χρήστη μας.

Στο κάτω μέρος της οθόνης ενημερωνόμαστε ότι εκεί θα μπορούμε να βλέπουμε **live** τα **στοιχεία** της εκάστοτε ανίχνευσης. Ειδικότερα, στις επόμενες δύο εικόνες, μπορούμε να δούμε δύο κινητές συσκευές που είναι σε mode ανίχνευσης και σε πολύ κοντινή απόσταση και το τι ακριβώς βλέπει ο καθένας στην οθόνη του.



Εικόνα 25: Ιχνηλάτηση επαφής χρήστη (Χρήστης Α)



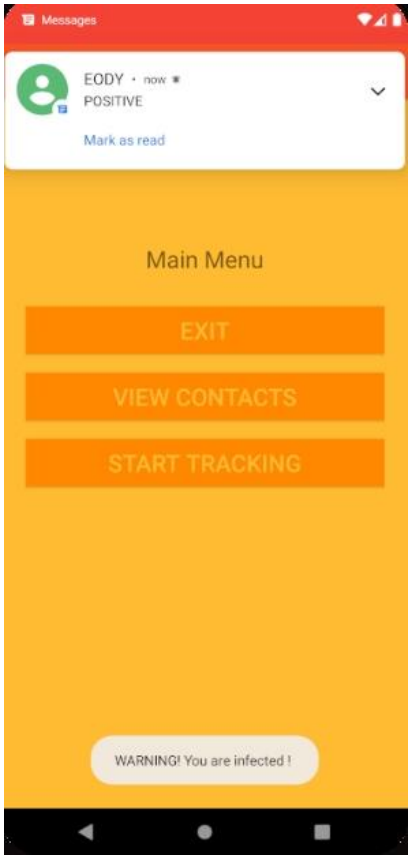
Εικόνα 26: Ιχνηλάτηση επαφής χρήστη (Χρήστης Β)

Όπως παρατηρούμε, την στιγμή που δύο χρήστες ήρθαν σε επαφή οι συσκευές των χρηστών μας εντόπισαν η μία την άλλη και στην περιοχή των στοιχείων ανίχνευσης βλέπουμε διάφορες πληροφορίες όπως:

- **User:** Αποτελεί το μοναδικό ID του άλλου χρήστη
- **MinDistance:** Αποτελεί την ελάχιστη απόσταση που έχουμε έρθει με τον άλλο χρήστη.
- **LastDistance:** Αποτελεί την τελευταία απόσταση που έχουμε έρθει με τον άλλο χρήστη.
- **isVisible:** Αποτελεί το εάν η συσκευή του άλλου χρήστη είναι ανιχνεύσιμη.
- **isNear:** Αποτελεί το εάν η συσκευή του άλλου χρήστη βρίσκεται κοντά.

Τέλος, όπως αναφέρθηκε και εκτενέστερα στην προηγούμενη παράγραφο *(4.2),* στο background πραγματοποιούνται συνεχώς έλεγχοι, τόσο στα SMS του χρήστη για το εάν έχει λάβει κάποιο μήνυμα

από τον ΕΟΔΥ που υποδεικνύει ότι διαγνώστηκε θετικός, τόσο και για το αν έχει διαγνωστεί θετικός κάποιος από τους χρήστες που έχει έρθει σε επαφή, εντός 14 ημερών (το ίδιο ισχύει και για το εάν κάποιος από τους χρήστες που έχει έρθει σε επαφή εντός 14 ημερών έρθει και αυτός με τη σειρά του σε επαφή με κάποιο μολυσμένο άτομο). Σε όλες αυτές τις περιπτώσεις ο χρήστης ενημερώνεται με κατάλληλο μήνυμα.



Εικόνα 27: Λήψη SMS από ΕΟΔΥ για μόλυνση χρήστη

Βιβλιογραφικές Αναφορές

1. *"A Survey of COVID-19 Contact Tracing Apps"*, Nadeem Ahmed, Regio A. Michelin, Wanli Xue, Sushmita Ruj, Robert Malaney, Salil S. Kanhere, Aruna Seneviratne, Wen Hu, Helge Janicke, And Sanjay Jha, 2020

2. *"Applicability of mobile contact tracing in fighting pandemic (COVID-19): Issues, challenges and solutions"*, Aaqib Bashir Dar, Auqib Hamid Lone, Saniya Zahoor, Afshan Amin Khan, Roohie Naaz, 2020

3. *"Applications of digital technology in COVID-19 pandemic planning and response"*, Sera Whitelaw, Mamas A Mamas, Eric Topol, Harriette G C Van Spall, 2020

Παράρτημα Α':

Κώδικας Android Εφαρμογής Μεταπτυχιακής Εργασίας

Package "corona-tracker"

Σημειώνεται πως το package "close-to-me" αποτελεί την βιβλιοθήκη και την οποία χρησιμοποιήσαμε για τον εντοπισμό Bluetooth (BLE).

- **Αρχείο "Contact.kt":**

```
package com.example.coronatracker

import java.text.ParseException
import java.text.SimpleDateFormat
import java.util.*

class Contact {
    var user: String? = null
```

```
    private var contacts: MutableList<ContactDetail>? = null
    public var isFlagged: Boolean = false
    public var isInfected: Boolean = false

    fun getContacts(): List<ContactDetail>? {
        return contacts
    }

    fun setContacts(contacts: MutableList<ContactDetail>?) {
        this.contacts = contacts
    }

    fun addContact(newContact: ContactDetail) {
        contacts!!.add(newContact)
    }

    fun recentContacts(): List<ContactDetail> {
        val currentTimestamp = currentTimestamp()
        val results: MutableList<ContactDetail> = ArrayList()
        if (contacts !== null) {
            for (i in contacts!!.indices) {
                if (isValidTimestamp(contacts!![i].timestamp, currentTimestamp)) {
                    results.add(contacts!![i])
                }
            }
        }
        return results
    }

    private fun isValidTimestamp(timestamp: String, currentTimestamp: String):
Boolean {
        // initialize date objects
        var currentTime: Date? = null
        var newTime: Date? = null
        try {
            currentTime = SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").parse(currentTimestamp)
```

```
            newTime = SimpleDateFormat("yyyy-MM-dd HH:mm:ss").parse(timestamp)
        } catch (e: ParseException) {
            e.printStackTrace()
        }

        // prepare date objects for comparison
        val c = Calendar.getInstance()
        c.time = currentTime
        c.add(Calendar.DAY_OF_MONTH, -14)
        val oldDate = c.time
        c.time = newTime
        val newDate = c.time

        // compare month and day
        val oldCalendar = Calendar.getInstance()
        val newCalendar = Calendar.getInstance()
        oldCalendar.time = oldDate
        newCalendar.time = newDate
        val yearDiff = oldCalendar[Calendar.YEAR] - newCalendar[Calendar.YEAR]
        val monthDiff = oldCalendar[Calendar.MONTH] - newCalendar[Calendar.MONTH]
        val dayDiff = oldCalendar[Calendar.DAY_OF_MONTH] -
newCalendar[Calendar.DAY_OF_MONTH]
        return if (yearDiff == 0 && monthDiff == 0 && (dayDiff <= 14 || dayDiff >=
0)) {
            true
        } else {
            false
        }
    }
    companion object {
        fun currentTimestamp(): String {
            val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
            val date = Date()
            return dateFormat.format(date)
        }
    }
}
```

```
class ContactDetail {
    var contactID:String = ""
    var timestamp:String = Contact.currentTimestamp()
    var latitude: Double = 0.0
    var longitude: Double = 0.0
}
```

- **Αρχείο "ContactListActivity.kt":**

```
package com.example.coronatracker

import android.content.Context
import android.graphics.Color
import android.os.Bundle
import android.view.View
import android.widget.LinearLayout
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.example.coronatracker.sample.R

class ContactListActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_contact_list)
        val prefs = getSharedPreferences(MyPREFS, Context.MODE_PRIVATE)
        val listName = prefs.getString("list-name", "empty")
        val listNameTxt = findViewById<TextView>(R.id.contactListName)
        listNameTxt.text = listName
        listNameTxt.textAlignment = View.TEXT_ALIGNMENT_CENTER
        listNameTxt.setTextColor(Color.WHITE)
        listNameTxt.textSize = 24.0f
        val contactsViews = findViewById<LinearLayout>(R.id.contactList)

        val userName = prefs!!.getString("user-id", "empty")
        ContactUtil.fetchUser(userName!!) { self ->

            if (!self.user.equals("empty")) {
```

```
                var contactList = mutableListOf<ContactDetail>()
                if (listName.equals("Today's Contacts")) {
                    contactList = ContactUtil.recentContacts(self!!,
0).toMutableList()
                } else if (listName.equals("Last Week's Contacts")) {
                    contactList = ContactUtil.recentContacts(self!!,
7).toMutableList()
                } else if (listName.equals("Last Month's Contacts")) {
                    contactList = ContactUtil.recentContacts(self!!,
30).toMutableList()
                } else {
                    contactList = self.getContacts() as MutableList<ContactDetail>?
?: mutableListOf()
                }

                if (contactList.isNotEmpty()) {
                    for (i in 0..contactList.size) {
                        // generate random name
                        var name = ""
                        if (contactList.getOrNull(i) != null) {
                            name = contactList.get(i).contactID
                        } else {
                            break
                        }

                        // add random name to contacts list
                        val contact = TextView(this)
                        contact.text = """
                Contact ID - ${i + 1}:
                $name
                -----------------------

                """.trimIndent()
                        // contact.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);
                        contact.setTextColor(Color.WHITE)
                        contact.textSize = 24.0f
                        contactsViews.addView(contact)
                    }
```

```
                    val space1 = TextView(this)
                    space1.setTextColor(Color.WHITE)
                    space1.textSize = 24.0f
                    contactsViews.addView(space1)
                    val space2 = TextView(this)
                    space2.setTextColor(Color.WHITE)
                    space2.textSize = 24.0f
                    contactsViews.addView(space2)
                } else {
                    var result = TextView(this)
                    result.text = "No results!"
                    result.setTextColor(Color.WHITE)
                    result.textSize = 24.0f
                    contactsViews.addView(result)
                }
            }
        }
    }

    companion object {
        const val MyPREFS = "MyPrefs"
    }
}
```

- **Αρχείο “ContactsMenuActivity.kt”:**

```
package com.example.coronatracker

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.view.View
import android.widget.Button
```

```
import androidx.appcompat.app.AppCompatActivity
import com.example.coronatracker.sample.R

class ContactsMenuActivity : AppCompatActivity() {
    var sharedpreferences: SharedPreferences? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_contacts)
        sharedpreferences = getSharedPreferences(MyPREFS, Context.MODE_PRIVATE)
        val btn1 = findViewById<Button>(R.id.today)
        btn1.setOnClickListener { v: View? ->
            val editor = sharedpreferences!!.edit()
            editor.putString("list-name", "Today's Contacts")
            editor.commit()

            // Explicit Intent by specifying its class name
            val i = Intent(applicationContext, ContactListActivity::class.java)
            // Starts TargetActivity
            startActivity(i)
        }
        val btn2 = findViewById<Button>(R.id.lastweek)
        btn2.setOnClickListener { v: View? ->
            val editor = sharedpreferences!!.edit()
            editor.putString("list-name", "Last Week's Contacts")
            editor.commit()

            // Explicit Intent by specifying its class name
            val i = Intent(applicationContext, ContactListActivity::class.java)
            // Starts TargetActivity
            startActivity(i)
        }
        val btn3 = findViewById<Button>(R.id.lastmonth)
        btn3.setOnClickListener { v: View? ->
            val editor = sharedpreferences!!.edit()
            editor.putString("list-name", "Last Month's Contacts")
            editor.commit()
```

```
            // Explicit Intent by specifying its class name
            val i = Intent(applicationContext, ContactListActivity::class.java)
            // Starts TargetActivity
            startActivity(i)
        }
        val btn4 = findViewById<Button>(R.id.allcontacts)
        btn4.setOnClickListener { v: View? ->
            val editor = sharedpreferences!!.edit()
            editor.putString("list-name", "All Contacts")
            editor.commit()

            // Explicit Intent by specifying its class name
            val i = Intent(applicationContext, ContactListActivity::class.java)
            // Starts TargetActivity
            startActivity(i)
        }
    }


    companion object {
        const val MyPREFS = "MyPrefs"
    }
}
```

- Αρχείο “ContactUtil.kt”:

```
package com.example.coronatracker

import android.annotation.SuppressLint
import android.os.Build
import androidx.annotation.RequiresApi
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
import java.text.ParseException
import java.text.SimpleDateFormat
import java.time.LocalDate
```

```
import java.time.format.DateTimeFormatter
import java.time.temporal.ChronoUnit
import java.util.*

internal class ContactUtil {
    companion object {
        @RequiresApi(Build.VERSION_CODES.O)
        fun isValidTimestamp(timestamp: String, currentTimestamp: String, difference:
Int): Boolean {
            val formatter: DateTimeFormatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd HH:mm:ss")
            val start: LocalDate = LocalDate.parse(timestamp, formatter)
            val end: LocalDate = LocalDate.parse(currentTimestamp, formatter)

            val dayDiff = ChronoUnit.DAYS.between(start, end)
            return if (dayDiff <= difference) {
                true
            } else {
                false
            }
        }

        // fetch all contacts
        fun getContacts(contact: Contact): List<String> {
            val users: MutableList<String> = ArrayList()
            for (i in contact.getContacts()!!.indices) {
                val user = contact.getContacts()!![i]
                users.add("Contact: " + user.contactID + " Timestamp: " +
user.timestamp)
            }
            return users
        }

        @RequiresApi(Build.VERSION_CODES.O)
        fun recentContacts(contact: Contact, difference: Int): List<ContactDetail> {
            val currentTimestamp = currentTimestamp()
            val results: MutableList<ContactDetail> = ArrayList()
            if (contact.getContacts() != null) {
```

```
                for (i in contact.getContacts()!!.indices) {
                    if (isValidTimestamp(contact.getContacts()!![i].timestamp,
currentTimestamp, difference)) {
                        results.add(contact.getContacts()!![i])
                    }
                }
            }

            return results
        }

        // update user's contacts
        fun update(contact: Contact, newContact: ContactDetail?) {
            val ref = FirebaseDatabase.getInstance().getReference("Contacts")
            val updatedContact = Contact()
            val newContacts: MutableList<ContactDetail>? =
mutableListOf<ContactDetail>()
            newContacts?.addAll(contact.getContacts()!!)
            updatedContact.setContacts(newContacts)
            updatedContact.addContact(newContact!!)
            ref.child(contact.user!!).setValue(updatedContact)
        }

        fun infected(contact: Contact) {
            val ref = FirebaseDatabase.getInstance().getReference("Contacts")
            contact.isInfected = true
            var contactsDetails = contact.recentContacts()

            if (contactsDetails != null) {
                contactsDetails.forEach { contactDetail ->

ref.child(contactDetail.contactID).child("flagged").setValue(true)
                }
                ref.child(contact.user!!).setValue(contact)
            }
        }

        // fetch user's contacts
```

```
        @SuppressLint("NewApi")
        fun fetchUser(user: String,
                      callback: (Contact) -> Unit) {
            // get all database data and order them by timestamp and user id
            val ref = FirebaseDatabase.getInstance().getReference("Contacts")
            val query = ref.orderByChild("timestamp")

            // prepare result object
            var result : Contact = Contact()
            result.user = "empty"
            query.addListenerForSingleValueEvent(object : ValueEventListener {
                override fun onDataChange(dataSnapshot: DataSnapshot) {
                    // check if there are any contacts
                    if (dataSnapshot.childrenCount > 0) {
                        // filter contacts by user id
                        for (cursor in dataSnapshot.children) {
                            val currentContact =
cursor.getValue(Contact::class.java)!!
                            if (currentContact.user.equals(user)) {
                                callback(currentContact)
                            }
                        }
                    }
                }

                override fun onCancelled(databaseError: DatabaseError) {
                    throw databaseError.toException()
                }
            })

        }

        fun currentTimestamp(): String {
            val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
            val date = Date()
            return dateFormat.format(date)
```

```
        }

        fun getCountOfDays(timestamp: String, currentTimestamp: String): Int {
            val dateFormat = SimpleDateFormat("yyyy-MM-dd HH:mm:ss",
Locale.getDefault())
            var createdConvertedDate: Date? = null
            var expireCovertedDate: Date? = null
            var todayWithZeroTime: Date? = null

            try {
                createdConvertedDate = dateFormat.parse(timestamp)
                expireCovertedDate = dateFormat.parse(currentTimestamp)
                val today = Date()
                todayWithZeroTime = dateFormat.parse(dateFormat.format(today))
            } catch (e: ParseException) {
                e.printStackTrace()
            }

            var cYear = 0
            var cMonth = 0
            var cDay = 0
            if (createdConvertedDate!!.after(todayWithZeroTime)) {
                val cCal = Calendar.getInstance()
                cCal.time = createdConvertedDate
                cYear = cCal[Calendar.YEAR]
                cMonth = cCal[Calendar.MONTH]
                cDay = cCal[Calendar.DAY_OF_MONTH]
            } else {
                val cCal = Calendar.getInstance()
                cCal.time = todayWithZeroTime
                cYear = cCal[Calendar.YEAR]
                cMonth = cCal[Calendar.MONTH]
                cDay = cCal[Calendar.DAY_OF_MONTH]
            }

            val eCal = Calendar.getInstance()
            eCal.time = expireCovertedDate
```

```
            val eYear = eCal[Calendar.YEAR]
            val eMonth = eCal[Calendar.MONTH]
            val eDay = eCal[Calendar.DAY_OF_MONTH]
            val date1 = Calendar.getInstance()
            val date2 = Calendar.getInstance()
            date1.clear()
            date1[cYear, cMonth] = cDay
            date2.clear()
            date2[eYear, eMonth] = eDay
            val diff = date2.timeInMillis - date1.timeInMillis
            val dayCount = diff.toFloat() / (24 * 60 * 60 * 1000)
            return dayCount.toInt()
        }
    }
}
```

- **Αρχείο “MainActivity.kt”:**

```
package com.example.coronatracker

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.os.Handler
import android.os.Process
import android.util.Log
import android.view.View
import android.view.WindowManager
import android.widget.Button
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.coronatracker.sample.R
import com.google.firebase.database.*
import java.util.*
```

```
class MainActivity : AppCompatActivity() {

    var prefs: SharedPreferences? = null
    var firebaseRef: DatabaseReference? = null
    var contact: Contact? = null
    var maxid: Long = 0

    private val userUuid = UUID.randomUUID()
    private lateinit var randID : String

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //keeping screen on
        window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON)

        PermissionManager.setPermissions(this)

        prefs = getSharedPreferences(MyPREFS, Context.MODE_PRIVATE)
        val listName = prefs!!.getString("user-id", "empty")
        var checked = prefs!!.getBoolean("checked", false)

        val ha = Handler()
        ha.postDelayed(object : Runnable {
            override fun run() {
                checked = prefs!!.getBoolean("checked", false)
                if (checked == false) {
                    val ref = FirebaseDatabase.getInstance().getReference("Contacts")

                    ContactUtil.fetchUser(listName!!) { me ->
                        var contacts = me.recentContacts()
                        var checkNext = true

                        for (c in contacts) {
```

```
                            if (checkNext == false) break

                            ContactUtil.fetchUser(c.contactID!!) { result ->
                                if (result.isInfected || result.isFlagged) {
                                    // update database
                                    if (!me.isInfected)
ref.child(me.user!!).child("flagged").setValue(true)

                                    // change shared pref
                                    val editor = prefs!!.edit()
                                    editor.putBoolean("checked", true)
                                    editor.commit()

                                    // inform user
                                    Toast.makeText(applicationContext, "WARNING! Some
of your recent contacts are infected or flagged !", Toast.LENGTH_SHORT).show()
                                    checkNext = false
                                }
                            }
                        }

                        if (me.isInfected || me.isFlagged) {
                            val editor = prefs!!.edit()
                            editor.putBoolean("checked", true)
                            editor.commit()
                            if (me.isInfected) {
                                Toast.makeText(applicationContext, "WARNING! You are
infected ! ", Toast.LENGTH_SHORT).show()
                            } else {
                                Toast.makeText(applicationContext, "WARNING! You are
flagged ! ", Toast.LENGTH_SHORT).show()
                            }

                            ContactUtil.fetchUser(listName!!) { me ->
                                var contacts = me.recentContacts()

                                contacts?.forEach { c ->
                                    ContactUtil.fetchUser(c.contactID!!) { result ->
```

```
                                        if (!result.isFlagged && !result.isInfected)
{
                                            // update database

ref.child(result.user!!).child("flagged").setValue(true)
                                        }
                                    }
                                }
                            }
                        }

                        ha.postDelayed(this, 3600000)  //Repeats hourly
                    } else {
                        Toast.makeText(applicationContext, "Check Completed !",
Toast.LENGTH_SHORT).show()
                    }
                }
            }, 5000)  //First run in 5 seconds

            SmsReceiver.bindListener(object : SmsListener {
                override fun messageReceived(messageText: String?) {
                    if (messageText!!.contains("POSITIVE")) {
                        // read user's rand id
                        val prefs = applicationContext.getSharedPreferences(MyPREFS,
MODE_PRIVATE)
                        val userName = prefs!!.getString("user-id", "empty")

                        ContactUtil.fetchUser(userName!!) {
                            ContactUtil.infected(it)
                        }
                    }
                }
            })

            contact = Contact()
            firebaseRef = FirebaseDatabase.getInstance().reference.child("Contacts")
            firebaseRef!!.addValueEventListener(object : ValueEventListener {
```

```
        override fun onDataChange(snapshot: DataSnapshot) {
            if (snapshot.exists()) {
                maxid = snapshot.childrenCount
            }
        }

        override fun onCancelled(error: DatabaseError) {}
    })

    if (listName == "empty") {

        // generate random name
        randID = userUuid.toString()

        // store it
        val editor = prefs!!.edit()
        editor.putString("user-id", randID)
        editor.commit()

        contact!!.user = randID
        contact!!.setContacts(ArrayList())
        firebaseRef!!.child((randID)).setValue(contact)
    }

    val btn = findViewById<Button>(R.id.contacts_btn)
    btn.setOnClickListener { v: View? ->
        val i = Intent(applicationContext, ContactsMenuActivity::class.java)
        // Starts TargetActivity
        startActivity(i)
    }

    val btn2 = findViewById<Button>(R.id.tracking_btn)
    btn2.setOnClickListener { v: View? ->
        val i = Intent(applicationContext, TrackingActivity::class.java)
        startActivity(i)
    }
```

```
        val exit = findViewById<Button>(R.id.exit_btn)
        exit.setOnClickListener { view: View? ->
            moveTaskToBack(true)
            Process.killProcess(Process.myPid())
            System.exit(1)
        }
    }

    companion object {
        private const val PERMISSION_REQUEST_CODE = 100
        private var stop = false
        val MyPREFS = "MyPrefs"
    }
}
```

- **Αρχείο “PermissionManager.kt”:**

```
package com.example.coronatracker

import android.Manifest
import android.app.Activity
import android.content.Context
import com.gun0912.tedpermission.PermissionListener
import com.gun0912.tedpermission.TedPermission

object PermissionManager {
    private lateinit var permissionListener: PermissionListener

    fun setPermissions(context: Context){
        setPermissionListener(context)
        TedPermission.with(context)
            .setPermissionListener(permissionListener)
            .setDeniedMessage("Please grant the required permissions for the app to
function correctly.")
            .setDeniedCloseButtonText("Exit")
            .setGotoSettingButtonText("Settings")
            .setPermissions(
                Manifest.permission.RECEIVE_SMS,
```

```
                Manifest.permission.ACCESS_FINE_LOCATION,
                Manifest.permission.READ_SMS)
            .check()
    }

    private fun setPermissionListener(context: Context){
        permissionListener = object : PermissionListener {
            override fun onPermissionGranted() {
            }

            override fun onPermissionDenied(deniedPermissions: List<String>) {
                //Closing app if permissions not granted
                (context as Activity).finishAndRemoveTask()
            }
        }
    }
}
```

- **Αρχείο "SmsListener.kt":**

```
package com.example.coronatracker

interface SmsListener {
    fun messageReceived(messageText: String?)
}
```

- **Αρχείο "SmsReceiver.kt":**

```
package com.example.coronatracker

import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.telephony.SmsMessage
```

```
class SmsReceiver : BroadcastReceiver() {
    var b: Boolean? = null

    override fun onReceive(context: Context, intent: Intent) {
        val data = intent.extras
        val pdus = data!!["pdus"] as Array<Any>?

        for (i in pdus!!.indices) {
            val smsMessage = SmsMessage.createFromPdu(pdus[i] as ByteArray)
            val sender = smsMessage.displayOriginatingAddress

            b = sender.equals("EODY")
            if (b == true) {
                mListener!!.messageReceived(smsMessage.messageBody) // attach value
to interface object
            }
        }
    }

    companion object {
        private var mListener: SmsListener? = null
        fun bindListener(listener: SmsListener?) {
            mListener = listener
        }
    }
}
```

- **Αρχείο “TrackingActivity.kt”:**

```
package com.example.coronatracker

import android.Manifest
import android.annotation.SuppressLint
```

```
import android.content.Context
import android.content.SharedPreferences
import android.content.pm.PackageManager
import android.location.Location
import android.location.LocationListener
import android.location.LocationManager
import android.os.Bundle
import android.text.method.ScrollingMovementMethod
import android.view.WindowManager
import android.widget.Toast
import android.widget.Toast.LENGTH_LONG
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import androidx.core.view.isVisible
import androidx.databinding.DataBindingUtil
import androidx.lifecycle.Observer
import com.example.coronatracker.ContactListActivity.Companion.MyPREFS
import com.example.coronatracker.sample.R
import com.example.coronatracker.sample.databinding.ActivityTrackingBinding
import com.google.firebase.database.FirebaseDatabase
import java.util.*

@ExperimentalUnsignedTypes
class TrackingActivity : AppCompatActivity() {

    private lateinit var binding: ActivityTrackingBinding

    private val manufacturerUuid = UUID.fromString("01234567-89AB-CD01-2345-
67890ABCD012")

    var prefs: SharedPreferences? = null
    lateinit var listName: UUID
    private var latitude = 0.0
    private var longitude = 0.0
    private var gps_enabled: Boolean = false
    private var network_enabled: Boolean = false
```

```
    private lateinit var timer: Timer
    private lateinit var locationManager: LocationManager

    private var closeToMe: CloseToMe? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //keeping screen on
        window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON)
        startLocationListeners()
        prefs = getSharedPreferences(MyPREFS, Context.MODE_PRIVATE)
        listName = UUID.fromString(prefs!!.getString("user-id", "empty").toString())
        binding = DataBindingUtil.setContentView(this, R.layout.activity_tracking)

        initUi()

        val anyPermissionsNotGranted =
            permissions.any { permission -> ContextCompat.checkSelfPermission(this,
permission) != PackageManager.PERMISSION_GRANTED }

        if (anyPermissionsNotGranted) {
            ActivityCompat.requestPermissions(this, permissions, PERMISSIONS_REQUEST)
        } else {
            initCloseToMe()
        }
    }

    private fun initUi() {
        binding.user.text = "User:$listName"
        binding.log.movementMethod = ScrollingMovementMethod()
        binding.start.setOnClickListener { onStartClick() }
        binding.stop.setOnClickListener { onStopClick() }
    }

    override fun onRequestPermissionsResult(requestCode: Int, permissions:
Array<String>, grantResults: IntArray) {
        when (requestCode) {
```

```
            PERMISSIONS_REQUEST -> {
                if (grantResults.isEmpty()) {
                    onPermissionNotGranted()
                }

                val isNotGranted = grantResults.any {
                    it != PackageManager.PERMISSION_GRANTED
                }

                if (isNotGranted) {
                    onPermissionNotGranted()
                } else {
                    initCloseToMe()
                }
            }
        }
    }

    private fun onPermissionNotGranted() {
        Toast.makeText(this, R.string.permission_is_required,
Toast.LENGTH_SHORT).show()
        finish()
    }

    private fun initCloseToMe() {
        closeToMe = CloseToMe.Builder(this, manufacturerUuid)
            .setUserUuid(listName)
            .setMajor(1U)
            .setMinor(1U)
            .setVisibilityDistanceMeter(2.0)
            .setVisibilityTimeoutMs(5_000)
            .build().also {

                if (!it.hasBleFeature()) {
                    Toast.makeText(this, R.string.ble_not_supported,
Toast.LENGTH_LONG).show()
                    finish()
```

```
                }

                it.state.observe(this, Observer { state ->
                    Toast.makeText(applicationContext, "Beacon state: $state",
Toast.LENGTH_SHORT).show()

                    when (state) {
                        CloseToMeState.STARTED -> {
                            binding.start.isVisible = false
                            binding.stop.isVisible = true
                        }
                        else -> {
                            binding.start.isVisible = true
                            binding.stop.isVisible = false
                        }
                    }
                })

                it.results.observe(this, Observer { beacons ->
                    binding.result.text = beacons.values.joinToString("\n------------
--------------------\n") { beacon ->

                        // save new contact
                        if (!beacon.userUuid.isNullOrBlank()) {
                            ContactUtil.fetchUser(listName.toString()) { self ->
                                if (self.getContacts() != null) {
                                    // concat exist?
                                    var res = self.getContacts()!!.filter {
it.contactID == beacon.userUuid }

                                    if (res.size == 1) {
                                        // update old contact
                                        var newContactDetail = ContactDetail()
                                        self.getContacts()!!.find { it.contactID ==
beacon.userUuid }?.timestamp = newContactDetail.timestamp
                                        self.getContacts()!!.find { it.contactID ==
beacon.userUuid }?.longitude = longitude
```

```
                                        self.getContacts()!!.find { it.contactID ==
beacon.userUuid }?.latitude = latitude

                                        val ref =
FirebaseDatabase.getInstance().getReference("Contacts")

ref.child(listName.toString()).child("contacts").setValue(self.getContacts()!!)
                                    } else {
                                        // add new contact
                                        var newContactDetail = ContactDetail()
                                        newContactDetail.contactID =
beacon.userUuid.toString()
                                        newContactDetail.latitude = latitude
                                        newContactDetail.longitude = longitude
                                        ContactUtil.update(self, newContactDetail)
                                    }
                                } else {
                                    var newContactDetail = ContactDetail()
                                    newContactDetail.contactID =
beacon.userUuid.toString()

                                    val newContacts: MutableList<ContactDetail> =
mutableListOf<ContactDetail>()
                                    newContacts.add(newContactDetail)

                                    val ref =
FirebaseDatabase.getInstance().getReference("Contacts")

ref.child(listName.toString()).child("contacts").setValue(newContacts)
                                }
                            }
                        }

                        // print new contact
                        "User: ${beacon.userUuid}\n" +
                                "MinDistance:
${"%.2f".format(beacon.minDistanceInMeter)}m\n" +
                                "LastDistance:
${"%.2f".format(beacon.distanceInMeter)}m\n" +
```

```
                                "isVisible: ${beacon.isVisible}\n" +
                                "isNear: ${beacon.isNear}"
                    }

                    //log(this, "Result: $beacons", Toast.LENGTH_SHORT).show()
                })
            }
    }

    private fun onStartClick() {
        if (closeToMe?.isBluetoothEnabled?.value != true) {
            log("Enabling bluetooth...")

            closeToMe?.enableBluetooth(object : CloseToMeCallback {
                override fun onSuccess() {
                    Toast.makeText(applicationContext, "Bluetooth is on now",
Toast.LENGTH_SHORT).show()
                    log("Bluetooth is on now")
                }

                override fun onError(throwable: Throwable) {
                    log(throwable.message ?: throwable.toString())
                }
            })
        } else {
            closeToMe?.start(object : CloseToMeCallback {
                override fun onSuccess() {
                    Toast.makeText(applicationContext, "Beacon started
successfully!", Toast.LENGTH_SHORT).show()
                    log("Beacon started successfully!")
                }

                override fun onError(throwable: Throwable) {
                    log(throwable.message ?: throwable.toString())
                }
            })
        }
```

```
    }

    private fun onStopClick() {
        closeToMe?.stop(object : CloseToMeCallback {
            override fun onSuccess() {
                Toast.makeText(applicationContext, "Beacon stopped successfully!",
Toast.LENGTH_SHORT).show()
                log("Beacon stopped successfully!")
            }

            override fun onError(throwable: Throwable) {
                log(throwable.message ?: throwable.toString())
            }
        })
    }

    private fun log(message: String) {
        runOnUiThread {
            binding.log.text = "$message\n" +
                "-----------------------------------\n" +
                "${binding.log.text}"
        }
    }

    override fun onPause() {
        closeToMe?.stop()
        locationManager.removeUpdates(locationListenerGps)
        super.onPause()
    }

    override fun onResume() {
        super.onResume()
        startLocationListeners()
    }

    @SuppressLint("MissingPermission")
    private fun startLocationListeners(){
```

```
        //location listeners initialization
        locationManager = this.getSystemService(LOCATION_SERVICE) as LocationManager
        gps_enabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
        network_enabled =
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 2000,
0f,
                locationListenerGps)
        if (network_enabled)
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 2000, 0f,
                locationListenerNetwork)
        timer = Timer()
        timer.schedule(GetLastLocation(), 20000)
    }

    //GPS Location Listener
    var locationListenerGps: LocationListener = object : LocationListener {
        override fun onLocationChanged(location: Location) {
            timer.cancel()
            latitude = location.latitude
            longitude = location.longitude
            locationManager.removeUpdates(locationListenerNetwork)
        }

        override fun onProviderDisabled(provider: String) {
            Toast.makeText(applicationContext,
                    "Please enable Location Services in order for the SmartAlert app
to function properly.",
                    LENGTH_LONG).show()
        }

        override fun onProviderEnabled(provider: String) {}
        override fun onStatusChanged(provider: String, status: Int, extras: Bundle)
{}
    }

    //Network Location Listener
    var locationListenerNetwork: LocationListener = object : LocationListener {
```

```
        override fun onLocationChanged(location: Location) {
            timer.cancel()
            latitude = location.latitude
            longitude = location.longitude
            locationManager.removeUpdates(this)
        }

        override fun onProviderDisabled(provider: String) {}
        override fun onProviderEnabled(provider: String) {}
        override fun onStatusChanged(provider: String, status: Int, extras: Bundle)
{}
    }

    companion object {
        const val PERMISSIONS_REQUEST = 1010

        val permissions: Array<String> = arrayOf(
                Manifest.permission.BLUETOOTH,
                Manifest.permission.BLUETOOTH_ADMIN,
                Manifest.permission.ACCESS_FINE_LOCATION
        )
    }

    inner class GetLastLocation : TimerTask() {
        //Getting last known location after 20 seconds if both GPS and Network
locations failed
        @SuppressLint("MissingPermission")
        override fun run() {
            //locationManager.removeUpdates(locationListenerGps);
            locationManager.removeUpdates(locationListenerNetwork)
            var net_loc: Location? = null
            var gps_loc: Location? = null
            if (gps_enabled) gps_loc =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER)
            if (network_enabled) net_loc =
locationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER)

            //if there are both values use the latest one
```

```
            if (gps_loc != null && net_loc != null) {
                if (gps_loc.time > net_loc.time) {
                    latitude = gps_loc.latitude
                    longitude = gps_loc.longitude
                } else {
                    latitude = net_loc.latitude
                    longitude = net_loc.longitude
                }
            }
            if (gps_loc != null) {
                run {
                    latitude = gps_loc.latitude
                    longitude = gps_loc.longitude
                }
            }
            if (net_loc != null) {
                run {
                    latitude = net_loc.latitude
                    longitude = net_loc.longitude
                }
            }
        }
    }
}
```

- **Αρχείο “activity_contact_list.xml”:**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/holo_orange_light"
    tools:context="com.example.coronatracker.ContactListActivity">

    <ScrollView
```

```
android:layout_width="360dp"
android:layout_height="671dp"
android:layout_marginStart="8dp"
android:layout_marginTop="8dp"
android:layout_marginEnd="8dp"
android:layout_marginBottom="8dp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent">

<LinearLayout
    android:id="@+id/contactList"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView9"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView8"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView10"
```

```
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textSize="24sp" />

            <TextView
                android:id="@+id/contactListName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textSize="24sp" />

            <TextView
                android:id="@+id/textView13"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textSize="24sp" />
        </LinearLayout>
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

- **Αρχείο “activity_contacts.xml”:**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/holo_orange_light"
    tools:context="com.example.coronatracker.ContactsMenuActivity">

    <LinearLayout
        android:layout_width="349dp"
        android:layout_height="673dp"
        android:layout_marginStart="8dp"
```

```
android:layout_marginTop="8dp"
android:layout_marginEnd="8dp"
android:layout_marginBottom="8dp"
android:orientation="vertical"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent">

<TextView
    android:id="@+id/textView14"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="24sp" />

<TextView
    android:id="@+id/textView15"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="24sp" />

<TextView
    android:id="@+id/textView16"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="24sp" />

<TextView
    android:id="@+id/textView17"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="24sp" />

<TextView
    android:id="@+id/textView18"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```
        android:text="@string/contacts_menu"
        android:textAlignment="center"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView19"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <Button
        android:id="@+id/today"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_orange_dark"
        android:text="@string/today"
        android:textColor="@android:color/holo_orange_light"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView20"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/lastweek"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_orange_dark"
        android:text="@string/last_week"
        android:textColor="@android:color/holo_orange_light"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView21"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
```

```
        <Button
            android:id="@+id/lastmonth"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@android:color/holo_orange_dark"
            android:text="@string/last_month"
            android:textColor="@android:color/holo_orange_light"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/textView22"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <Button
            android:id="@+id/allcontacts"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@android:color/holo_orange_dark"
            android:text="@string/all_contacts"
            android:textColor="@android:color/holo_orange_light"
            android:textSize="24sp" />

    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

- **Αρχείο “activity_main.xml”:**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:background="@android:color/holo_orange_light"
tools:context="com.example.coronatracker.MainActivity" >

<LinearLayout
    android:layout_width="362dp"
    android:layout_height="682dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView5"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView11"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/main_menu"
        android:textAlignment="center"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView7"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="24sp" />

    <Button
        android:id="@+id/exit_btn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_orange_dark"
        android:text="@string/exit"
        android:textColor="@android:color/holo_orange_light"
        android:textSize="24sp" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/contacts_btn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_orange_dark"
```

```
            android:text="@string/view_contacts"
            android:textColor="@android:color/holo_orange_light"
            android:textSize="24sp" />

        <TextView
            android:id="@+id/textView12"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

        <Button
            android:id="@+id/tracking_btn"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@android:color/holo_orange_dark"
            android:text="Start Tracking"
            android:textColor="@android:color/holo_orange_light"
            android:textSize="24sp" />

    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

- **Αρχείο “activity_tracking.xml”:**

```
<?xml version="1.0" encoding="utf-8"?>

<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.coronatracker.TrackingActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FFC107"
        android:orientation="vertical">

        <LinearLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="16dp">

        <Button
            android:id="@+id/start"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#FF9800"
            android:text="Start"
            android:textColor="#FFC107"
            android:textSize="30sp" />

        <Button
            android:id="@+id/stop"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="#FF9800"
            android:text="Stop"
            android:textColor="#FFC107"
            android:textSize="30sp" />
    </LinearLayout>

    <TextView
        android:id="@+id/user"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#FFEB3B"
        android:padding="16dp" />

    <TextView
        android:id="@+id/log"
        android:layout_width="match_parent"
        android:layout_height="0dp"
```

```
            android:layout_weight="1"
            android:padding="16dp" />

        <TextView
            android:id="@+id/result"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/ic_launcher_background"
            android:padding="16dp"
            android:text="Results will show up here..." />
    </LinearLayout>
</layout>
```