Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Προηγμένα Συστήματα Πληροφορικής – Ανάπτυξη Λογισμικού και Τεχνητής Νοημοσύνης»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	*Η συνεισφορά της τεχνολογίας στην πανδημία του COVID-19 και οι εφαρμογές ιχνηλάτησης για smartphone* The contribution of technology to the COVID-19 pandemic and smartphone tracking applications
Ονοματεπώνυμο Φοιτητή	**Βασίλειος Ζωγράφος**
Πατρώνυμο	**Ιωάννης**
Αριθμός Μητρώου	**ΜΠΣΠ19013**
Επιβλέπων	**Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Ιούνιος 2021**

Τριμελής Εξεταστική Επιτροπή

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

Περίληψη

Τη σημερινή εποχή βιώνουμε καταστάσεις πρωτόγνωρες για το σύνολο της ανθρωπότητας. Η πανδημία της νόσου του κορονοϊού, χωρίς να κάνει καμία διάκριση σε φυλή, φύλο, ηλικία και πλούτο, έχει οδηγήσει στον θάνατο εκατομμυρίων συνανθρώπων μας, έχει αλλάξει δραματικά την καθημερινή ζωή όλων μας, ενώ ταυτόχρονα έχει επιφέρει τεράστιες οικονομικές και υγειονομικές καταστροφές ακόμα και σε όλες τις δυνάμεις. Αυτό έχει ως αποτέλεσμα, να καλούμαστε όλοι μας να βοηθήσουμε σε οποιοδήποτε βαθμό δύναται για την αντιμετώπιση αυτού του φαινομένου.

Επομένως, ο σκοπός της παρούσας εργασίας σχετίζεται με τις εφαρμογές έξυπνων κινητών που πραγματοποιούν ιχνηλάτηση των επαφών των κρουσμάτων του COVID-19.

Αρχικά, θα σας ενημερώσουμε περαιτέρω για την τρέχουσα πανδημία και θα καλύψουμε τα κύρια σημεία αναφορικά με τον κορονοϊό και τους τρόπους μετάδοσής του.

Ύστερα, θα εξηγήσουμε τους τομείς στους οποίους η ψηφιακή τεχνολογία μπορεί να φανεί χρήσιμη στην τρέχουσα κατάσταση, ενώ θα δώσουμε έμφαση στις εφαρμογές ιχνηλάτησης επαφών.

Συγκεκριμένα, θα καλύψουμε τα προβλήματα που αναδύονται σε αυτού του είδους εφαρμογές, τις διάφορες τακτικές και υλοποιήσεις αυτών, καθώς και τους κινδύνους που καραδωκούν τόσο σε επίπεδο ιδιωτικότητας όσο και σε επίπεδο επιθέσεων.

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

Abstract

Today, the situation we are experiencing was something unpredicted, unexpected and is extremely difficult for all of the humanity. The pandemic of coronavirus, without warning and without making any distinction in race, gender, age and wealth, has led to the death of millions, has dramatically changed the daily routine for all of us and for over a year, simultaneously has caused enormous economic and health disasters even on the greatest economies. As a result, we are all called upon to contribute however we can to combat this rare and challenging phenomenon.

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

Firstly, we will review the current situation and cover the main points regarding the virus and its ways of transmission.

Next, we will identify the areas in which digital technology can be useful in the current situation, while giving our attention to the user contact tracking mobile applications.

Specifically, we will inform you about the difficulties that occur from this type of applications, their different tactics and structures, as well as the risks involved both in terms of privacy and in terms of attacks.

Keywords: COVID-19, contact tracking, smartphone applications

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1ο:

Η πανδημία της νόσου του κορονοϊού COVID-19

1.1 Η πανδημία

Η πανδημία της νόσου του κορονοϊού 2019 (COVID-19) είναι μια τρέχουσα πανδημία που προκλήθηκε από τον κορονοϊό SARS-CoV-2 και αναγνωρίστηκε για πρώτη φορά στην πόλη Ουχάν, πρωτεύουσα της επαρχίας Χουπέι της Κίνας, τον Δεκέμβριο του 2019. Ως και τις 22 Νοεμβρίου 2020 είχαν επιβεβαιωθεί πάνω από 58,5 εκατομμύρια κρούσματα σε 215 χώρες και περιοχές, είχαν σημειωθεί περισσότεροι από 1,38 εκατομμύρια θάνατοι που οφείλονται στη νόσο και είχαν ανακάμψει περισσότεροι από 40,2 εκατομμύρια άνθρωποι.

Ο ιός μεταδίδεται μεταξύ των ανθρώπων μέσω των σταγονιδίων που παράγονται όταν οι άνθρωποι φτερνίζονται ή βήχουν. Ο χρόνος μεταξύ της έκθεσης και της εμφάνισης συμπτωμάτων είναι συνήθως από 2 έως 14 ημέρες. Τα συμπτώματα μπορεί να περιλαμβάνουν πυρετό, βήχα και δυσκολίες στην αναπνοή, ενώ επιστημονικές έρευνες υποστηρίζουν ότι πιθανή απώλεια γεύσης και όσφρησης αποτελούν συμπληρωματικές ενδείξεις μόλυνσης από τον ιό. Οι επιπλοκές μπορούν να περιλαμβάνουν πνευμονία και σύνδρομο οξείας αναπνευστικής δυσχέρειας. Μέχρι στιγμής, η υποστήριξη ανθρώπων που έχουν νοσήσει αποσκοπεί στην πρόωρη διαχείριση των συμπτωμάτων και της υποστηρικτικής θεραπείας.

Οι επαγγελματίες δημόσιας υγείας τονίζουν τη σημασία των βασικών πρακτικών υγιεινής στην πρόληψη της μόλυνσης και εξέδωσαν προτεινόμενες κατευθυντήριες οδηγίες συμπεριφοράς για άτομα που είναι ύποπτα ότι έχουν τον ιό. Το πλύσιμο των χεριών, η διατήρηση της απόστασης άνω των 2 μέτρων από ανθρώπους που βήχουν και η αποφυγή επαφής με το πρόσωπο συνιστώνται για την πρόληψη της νόσου. Οποιοσδήποτε είναι ύποπτος για τη μεταφορά του ιού, συνιστάται να παρακολουθεί την υγεία του για δύο εβδομάδες, να φοράει χειρουργική μάσκα και να ζητά ιατρική συμβουλή, καλώντας έναν γιατρό πριν επισκεφτεί μια κλινική.

Η περίοδος επώασης (χρόνος από την έκθεση έως την έναρξη των συμπτωμάτων) κυμαίνεται από 2 έως 14 ημέρες, αλλά μπορεί να είναι μεταδοτική κατά τη διάρκεια

αυτής της περιόδου και μετά την αποκατάσταση. Τα συμπτώματα περιλαμβάνουν πυρετό, βήχα και δυσκολίες στην αναπνοή. Ως τις 7 Απριλίου η εκτίμηση του ποσοστού θνησιμότητας ήταν το 1% των επιβεβαιωμένων περιπτώσεων, υψηλότερο μεταξύ εκείνων που απαιτούν εισαγωγή στο νοσοκομείο. Ως τις αρχές Απριλίου του 2020, δεν υπάρχει εμβόλιο και καμία ειδική θεραπεία. Εξετάζονται αρκετές προσεγγίσεις εμβολίων και αντιικών. Σχολεία και πανεπιστήμια έχουν κλείσει σε εθνικό ή τοπικό επίπεδο σε τουλάχιστον 188 χώρες, επηρεάζοντας πάνω από 1.5 δισεκατομμύρια μαθητές και φοιτητές.

Η πανδημία έχει κηρυχθεί από τον Παγκόσμιο Οργανισμό Υγείας (Π.Ο.Υ.) ως «Έκτακτη Ανάγκη Δημόσιας Υγείας Διεθνούς Ενδιαφέροντος» (PHEIC), με βάση τις πιθανές επιπτώσεις που θα μπορούσε να έχει ο ιός εάν εξαπλωθεί σε χώρες με ασθενέστερα συστήματα υγειονομικής περίθαλψης. Αυτή η κήρυξη ήταν η έκτη φορά μετά από την πανδημία του H1N1 το 2009.

1.2 Επιδημιολογία

Η επιδημιολογική ανάλυση της εκδήλωσης έδειξε ένα πιθανό μοτίβο μιας «μικτής εστίας» - πιθανότατα υπήρχε μια συνεχής κοινή εστία στην Αγορά Θαλασσινών Χουανάν το Δεκέμβριο του 2019, ενδεχομένως από διάφορα ζωονοσογόνα γεγονότα. Μετά από αυτό, οι επιδημιολόγοι διαπίστωσαν ότι το ξέσπασμα πιθανότατα μετατράπηκε σε πηγή (που μεταδίδεται από άτομο σε άτομο), πιθανώς λόγω της ικανότητας μετάλλαξης του ιού. Ως εκ τούτου, καθώς ο αριθμός των υποθέσεων έχει αυξηθεί, η σημασία της αγοράς έχει μειωθεί.

Κατά τα πρώτα στάδια, ο αριθμός των περιστατικών διπλασιαζόταν συνήθως κάθε 7,5 ημέρες. Στις αρχές και στα μέσα Ιανουαρίου του 2020, ο ιός εξαπλώθηκε σε άλλες κινεζικές επαρχίες, με τη βοήθεια των μετακινήσεων της Κινεζικής Πρωτοχρονιάς. Στις 20 Ιανουαρίου, η Κίνα ανέφερε σχεδόν 140 νέους ασθενείς την ημέρα, συμπεριλαμβανομένων δύο ατόμων στο Πεκίνο και ενός στη Σενζέν. Μεταγενέστερα επίσημα στοιχεία δείχνουν ότι 6.174 ασθενείς που έχουν μολυνθεί από SARS-CoV-2 είχαν ήδη αναπτύξει συμπτώματα έως τις 20 Ιανουαρίου 2020.

Στις 26 Φεβρουαρίου 2020, ο ΠΟΥ ανέφερε ότι, καθώς οι νέες περιπτώσεις που αναφέρθηκαν μειώθηκαν στην Κίνα, αλλά ξαφνικά αυξήθηκαν στην Ιταλία, το Ιράν και τη

Νότια Κορέα, ο αριθμός νέων περιπτώσεων εκτός Κίνας ξεπέρασε τον αριθμό νέων περιπτώσεων στην Κίνα για πρώτη φορά στις 25 Φεβρουαρίου 2020.

Επίσημες πηγές στην Γερμανία και το Ηνωμένο Βασίλειο εκτιμούν ότι το 60-70% του πληθυσμού θα μολυνθεί πριν επιτευχθεί ανοσία της αγέλης.

1.3 Μετάδοση

Ο κύριος τρόπος μετάδοσης είναι από άνθρωπο σε άνθρωπο μέσω των σταγονιδίων που αποβάλλουν οι άνθρωποι όταν βήχουν ή φτερνίζονται. Επιπλέον, αυτά τα σταγονίδια μπορούν να πέσουν πάνω σε επιφάνειες και να μεταδώσουν τον ιό όταν ο άνθρωπος αγγίξει την επιφάνεια, και στη συνέχεια αγγίξει τα μάτια, τη μύτη ή το στόμα του.

Ο ΠΟΥ αξιολογεί κατά πόσο η ασυμπτωματική μετάδοση και η μετάδοση των κοπράνων αποτελούν σημαντικούς τρόπους μετάδοσης.

Τα ιογόνα σταγονίδια παραμένουν αιωρούμενα στον αέρα για μικρό χρονικό διάστημα. Οι λεπτομέρειες για τον ιό της COVID-19 δεν είναι διαθέσιμες προς το παρόν αλλά υποτίθεται ότι είναι παρόμοιες με άλλους κορονοϊούς οποίοι μπορεί να παραμείνουν βιώσιμοι και μεταδοτικοί σε μεταλλική, γυάλινη ή πλαστική επιφάνεια για έως και εννέα ημέρες σε θερμοκρασία δωματίου. Απολύμανση επιφανειών είναι δυνατή με ουσίες όπως 62-71% αιθανόλη για ένα λεπτό.

Οι εκτιμήσεις για τον βασικό αριθμό αναπαραγωγής (ο μέσος αριθμός ατόμων που ένα μολυσμένο άτομο είναι πιθανό να μολύνει), κυμαίνονται από 2,13 έως 4,82. Στις 24 Ιανουαρίου 2020 αναφέρθηκε ότι ο ιός μπόρεσε να μεταδοθεί σε μια αλυσίδα έως και τεσσάρων ατόμων. Αυτό είναι παρόμοιο με το σοβαρό οξύ αναπνευστικό σύνδρομο από κορονοϊό (SARS-CoV) .

Ο ΠΟΥ αναφέρει επίσης ότι, αν και η ιστορία με το SARS και το MERS δείχνει ότι η μετάδοση μέσω τροφίμων δεν συμβαίνει, η δυνατότητα παραμένει ανοιχτή και ότι το κρέας και τα ζωικά προϊόντα κατά κανόνα πρέπει να μαγειρεύονται σχολαστικά.

Ο ιός ευρέως θεωρείται ότι έχει ζωονοτική προέλευση και ως εκ τούτου πιθανότατα μεταδίδεται από ζώο σε άνθρωπο, αν και το ζώο και ο τρόπος μετάδοσης δεν έχουν ακόμη ταυτοποιηθεί.

1.4 Διαχείριση της εξάπλωσης

Υπάρχουν δύο βασικές στρατηγικές για τον έλεγχο μιας εστίας: συγκράτηση και μετριασμός. Η συγκράτηση πραγματοποιείται στα αρχικά στάδια της επιδημίας και στοχεύει να εντοπίσει και να απομονώσει τους μολυσμένους για να σταματήσει η ασθένεια από το να εξαπλωθεί στον υπόλοιπο πληθυσμό. Όταν δεν είναι πλέον δυνατό να περιοριστεί η εξάπλωση της νόσου, οι προσπάθειες μεταφέρονται στη φάση μετριασμού, όταν λαμβάνονται μέτρα για να επιβραδυνθεί η εξάπλωση και να μετριαστούν οι επιπτώσεις της στο σύστημα υγείας και στην κοινωνία. Ένας συνδυασμός τόσο των μέτρων περιορισμού όσο και μετριασμού μπορεί να πραγματοποιηθεί ταυτόχρονα.

Μέρος της διαχείρισης μιας εστίας μολυσματικών ασθενειών προσπαθεί να μειώσει την κορύφωση της επιδημίας, γνωστή ως ισοπέδωση της επιδημικής καμπύλης. Αυτό μειώνει τον κίνδυνο κατάρρευσης των υπηρεσιών υγείας και παρέχει περισσότερο χρόνο για την ανάπτυξη εμβολίων και θεραπειών. Μη φαρμακευτικές παρεμβάσεις που μπορεί να διαχειριστούν την εστία περιλαμβάνουν προσωπικά προληπτικά μέτρα, όπως η υγιεινή των χεριών, η χρήση μάσκας προσώπου και η αυτοαπομόνωση · κοινοτικά μέτρα που στοχεύουν στην κοινωνική απομάκρυνση, όπως το κλείσιμο σχολείων και η ματαίωση εκδηλώσεων μαζικής συγκέντρωσης · συμμετοχή της κοινότητας για την ενθάρρυνση της αποδοχής και της συμμετοχής στις παρεμβάσεις αυτές · καθώς και περιβαλλοντικά μέτρα όπως καθαρισμών επιφανειών.

Πιο δραστικές ενέργειες έγιναν στην Κίνα μόλις έγινε εμφανής η σοβαρότητα της έξαρσης, όπως η απομόνωση ολόκληρων πόλεων που επηρέασαν 60 εκατομμύρια άτομα στη Χουπέι και αυστηρές ταξιδιωτικές απαγορεύσεις. Άλλες χώρες υιοθέτησαν ποικίλα μέτρα με στόχο τον περιορισμό της εξάπλωσης του ιού. Για παράδειγμα, η Νότια Κορέα εισήγαγε μαζική εξέταση, τοπικές καραντίνες, και έκανε ειδοποιήσεις σχετικά με τις κινήσεις των προσβεβλημένων ατόμων. Η Σιγκαπούρη παρείχε οικονομική υποστήριξη σε εκείνους τους μολυσμένους οι οποίοι απομόνωσαν τους

εαυτούς τους, και επέβαλε μεγάλα πρόστιμα σε όσους δεν το έκαναν. Η Ταϊβάν αύξησε την παραγωγή μασκών προσώπου και τιμωρούσε την αποθεματοποίηση των ιατρικών προμηθειών. Ορισμένες χώρες απαιτούσαν από τους ανθρώπους να αναφέρουν συμπτώματα που έμοιαζαν με γρίπη στον γιατρό τους, ειδικά εάν είχαν επισκεφθεί την ηπειρωτική Κίνα.

Κεφάλαιο 2ο:

Η συνεισφορά της ψηφιακής τεχνολογίας στην αντιμετώπιση της πανδημίας COVID-19

2.1 Σχεδιασμός και παρακολούθηση

Τα μεγάλα δεδομένα και η τεχνητή νοημοσύνη (AI) βοήθησαν στη διευκόλυνση της ετοιμότητας ενάντια του COVID-19 και της παρακολούθησης των ανθρώπων, και ενάντια στην εξάπλωση της λοίμωξης, σε πολλές χώρες. Εργαλεία όπως χάρτες μετεγκατάστασης, τα οποία χρησιμοποιούν κινητά τηλέφωνα, εφαρμογές πληρωμών μέσω κινητού και κοινωνικά μέσα για τη συλλογή δεδομένων σε πραγματικό χρόνο σχετικά με την τοποθεσία των ατόμων, επέτρεψαν στις κινεζικές αρχές να παρακολουθούν τη μετακίνηση ατόμων που είχαν επισκεφτεί την αγορά Γουχάν, το επίκεντρο της πανδημίας. Με αυτά τα δεδομένα, αναπτύχθηκαν μοντέλα μηχανικής μάθησης για να προβλέψουν τη δυναμική περιφερειακής μετάδοσης του SARS-CoV-2 και να καθοδηγήσουν τους ελέγχους και την επιτήρηση των συνόρων.

Μόλις η Κίνα ανέφερε το ξέσπασμα, η Ταϊβάν ξεκίνησε ελέγχους υγείας για ταξιδιώτες αεροπορικών εταιρειών από τη Γουχάν, ενσωματώνοντας δεδομένα από αρχεία μετανάστευσης με την κεντρική, εθνική βάση δεδομένων ασφάλισης υγείας σε πραγματικό χρόνο. Αυτή η ενσωμάτωση επέτρεψε στις εγκαταστάσεις υγειονομικής

περίθαλψης να έχουν πρόσβαση στο ιστορικό ταξιδιού των ασθενών και να εντοπίσουν άτομα για test και παρακολούθηση για τον SARS-CoV-2. Η γειτνίαση της Ταϊβάν με τη Γουχάν της Κίνας, έκανε την περιοχή ιδιαίτερα ευαίσθητη στο COVID-19, αλλά η αποτελεσματική χρήση μεγάλων δεδομένων πιστώνεται για τον χαμηλό αριθμό περιπτώσεων και θανάτων.

Στη Σουηδία, οι αρχές έχουν αναπτύξει μια πλατφόρμα για τους εργαζόμενους στον τομέα της υγειονομικής περίθαλψης για την αναφορά δεδομένων σε πραγματικό χρόνο σχετικά με τον όγκο των ασθενών με COVID-19, εξοπλισμό ατομικής προστασίας, στελέχωση, χρήση αναπνευστήρα και άλλες πληροφορίες πόρων. Αυτές οι πληροφορίες έχουν κοινοποιηθεί σε εθνικό επίπεδο με τις αρχές υγειονομικής περίθαλψης για την παρακολούθηση της κατάστασης των εγκαταστάσεων, τη διάθεση πόρων υγειονομικής περίθαλψης και την αύξηση της χωρητικότητας νοσοκομειακών κρεβατιών.

Η ανάγκη παρακολούθησης του COVID-19 τροφοδότησε την καινοτομία των ταμπλό δεδομένων που εμφανίζουν οπτικά το βάρος των ασθενειών. Το UpCode χρησιμοποιεί δεδομένα που παρέχονται από το Υπουργείο Υγείας της Σιγκαπούρης για να απεικονίσει τις τάσεις μόλυνσης σε όλη την ηλικία, το φύλο και την τοποθεσία και να σχεδιάσει το χρόνο αποκατάστασης μολυσμένων ατόμων. Το ταμπλό κορονοϊών του Πανεπιστημίου Johns Hopkins (MD, USA) και η διαδικτυακή πλατφόρμα HealthMap παρέχουν ενημερωμένες εικόνες για περιπτώσεις και θανάτους COVID-19 σε όλο τον κόσμο. Οι αλγόριθμοι AI επιτρέπουν την ενσωμάτωση της επίδρασης του κλίματος στις προβολές.

Το AI βέβαια έχει και αυτό περιορισμούς και απαιτεί εκπαίδευση με σύνολα δεδομένων για COVID-19. Τα περισσότερα από τα μοντέλα πρόβλεψης AI έχουν χρησιμοποιήσει κινεζικά δείγματα, τα οποία μπορεί να μην είναι γενικευμένα. Εκτός από την απουσία ιστορικών δεδομένων εκπαίδευσης, τα μέσα κοινωνικής δικτύωσης και η άλλη διαδικτυακή κυκλοφορία έχουν δημιουργήσει θόρυβο σε σύνολα μεγάλων δεδομένων, δημιουργώντας δυνητικά μοντέλα καλής εφαρμογής. Αυτός ο θόρυβος πρέπει να φιλτράρεται προτού διακριθούν οι ακριβείς τάσεις και προβλέψεις. Η ακρίβεια, η εγκυρότητα και η αξιοπιστία κάθε πρόβλεψης AI θα πρέπει να αξιολογούνται κατά την ερμηνεία των προβολών.

2.2 Ανίχνευση επαφών

Η Νότια Κορέα έχει εφαρμόσει εργαλεία για επιθετικό εντοπισμό επαφών, χρησιμοποιώντας πλάνα κάμερας ασφαλείας, τεχνολογία αναγνώρισης προσώπου, αρχεία τραπεζικών καρτών και δεδομένα παγκόσμιου συστήματος εντοπισμού θέσης (GPS) από οχήματα και κινητά τηλέφωνα για να παρέχει δεδομένα σε πραγματικό χρόνο και λεπτομερή χρονοδιαγράμματα των ταξιδιών των ανθρώπων. Οι Νότιοι Κορεάτες λαμβάνουν ειδοποιήσεις κειμένου έκτακτης ανάγκης σχετικά με νέες περιπτώσεις COVID-19 στην περιοχή τους και τα άτομα που θα μπορούσαν να έρθουν σε επαφή με μολυσμένα άτομα ενημερώνονται για αναφορά σε κέντρα δοκιμών και απομόνωση. Με τον εντοπισμό και την απομόνωση των μολύνσεων να έχουν λάβει χώρα νωρίς, η Νότια Κορέα έχει διατηρήσει μια μεταξύ των χαμηλότερων κατά κεφαλήν θνησιμότητα στον κόσμο.

Η Σιγκαπούρη ξεκίνησε μια εφαρμογή για smartphones που ανταλλάσσει σήματα Bluetooth μικρής απόστασης όταν τα άτομα βρίσκονται κοντά στο άλλο. Η εφαρμογή καταγράφει αυτές τις συναντήσεις και τις αποθηκεύει στα αντίστοιχα κινητά τηλέφωνα για 21 ημέρες. Εάν ένα άτομο διαγνωστεί με COVID-19, το Υπουργείο Υγείας της Σιγκαπούρης αποκτά πρόσβαση στα δεδομένα για να εντοπίσει τις επαφές του μολυσμένου ατόμου. Όπως και η Νότια Κορέα, η Σιγκαπούρη έχει διατηρήσει ένα από τα χαμηλότερα ποσοστά θνησιμότητας COVID-19 κατά κεφαλήν στον κόσμο.

Η Γερμανία έχει ξεκινήσει μια εφαρμογή smartwatch που συλλέγει δεδομένα παλμών, θερμοκρασίας και ύπνου για να ελέγξει για σημάδια ιογενούς ασθένειας. Τα δεδομένα από την εφαρμογή παρουσιάζονται σε έναν διαδικτυακό, διαδραστικό χάρτη στον οποίο οι αρχές μπορούν να εκτιμήσουν την πιθανότητα εμφάνισης COVID-19 σε ολόκληρη τη χώρα. Με εκτεταμένες δοκιμές και ψηφιακές παρεμβάσεις υγείας, η Γερμανία έχει διατηρήσει χαμηλό κατά κεφαλήν ποσοστό θνησιμότητας, σε σχέση με άλλες χώρες, παρά τον υψηλό επιπολασμό περιπτώσεων.

Οι εφαρμογές ανίχνευσης επαφών δεν απαλλαγμένες από παγίδες. Δεν απαιτείται καραντίνα για την παραμικρή έκθεση, όπως όταν τα εκτεθειμένα άτομα φορούν ατομικό προστατευτικό εξοπλισμό ή χωρίζονται από λεπτά τοιχώματα που διαπερνούνται από σήματα κινητού τηλεφώνου. Από την άλλη πλευρά, η σχετική έκθεση θα μπορούσε να χαθεί όταν τα άτομα δεν μεταφέρουν τα κινητά τους τηλέφωνα ή όταν δεν υπάρχει διαθέσιμη κινητή τηλεφωνία. Επιπλέον, ερευνητές στο Πανεπιστήμιο της Οξφόρδης (HB) έχουν προτείνει ότι το 60% του πληθυσμού μιας χώρας θα χρειαστεί να χρησιμοποιήσει μια εφαρμογή ανίχνευσης επαφών για να είναι αποτελεσματική μια στρατηγική μετριασμού.

2.3 Έλεγχος λοίμωξης

Η Κίνα χρησιμοποιεί δωρεάν εργαλεία που βασίζονται στον ιστό και βασίζονται στο cloud για να ελέγχουν και να κατευθύνουν τα άτομα σε κατάλληλους πόρους. Οι υπέρυθρες θερμικές κάμερες υψηλής απόδοσης που έχουν εγκατασταθεί στα αεροδρόμια της Ταϊβάν χρησιμοποιούνται για τη λήψη θερμικών εικόνων ανθρώπων σε πραγματικό χρόνο, εντοπίζοντας γρήγορα άτομα με πυρετό. Στη Σιγκαπούρη, οι άνθρωποι μετρούν τη θερμοκρασία τους στις εισόδους των χώρων εργασίας, των σχολείων και των μέσων μαζικής μεταφοράς. Τα δεδομένα από τα θερμόμετρα παρακολουθούνται και χρησιμοποιούνται για τον εντοπισμό αναδυόμενων καυτών σημείων και συστάδων λοίμωξης όπου θα μπορούσε να ξεκινήσει η δοκιμή.

Σε αντίθεση με τις περισσότερες άλλες χώρες, η Ισλανδία έχει ξεκινήσει εκτεταμένα test ασυμπτωματικών ατόμων. Χρησιμοποιώντας τεχνολογία κινητής τηλεφωνίας, η Ισλανδία συλλέγει δεδομένα σχετικά με τα συμπτώματα που αναφέρθηκαν από τον ασθενή και συνδυάζει αυτά τα δεδομένα με άλλα σύνολα δεδομένων όπως κλινικά και γονιδιωματικά δεδομένα αλληλουχίας για να αποκαλύψει πληροφορίες σχετικά με την παθολογία και την εξάπλωση του ιού. Αυτή η προσέγγιση έχει προσθέσει στη βάση γνώσεων σχετικά με τον επιπολασμό και τη μετάδοση του ασυμπτωματικού COVID-19. Μέχρι σήμερα, η Ισλανδία είχε το υψηλότερο ποσοστό test κατά κεφαλήν και ένα από τα χαμηλότερα ποσοστά θνησιμότητας COVID-19 κατά κεφαλήν. Άλλες χώρες που προσφέρουν εκτεταμένες δοκιμές περιλαμβάνουν τη Γερμανία και τη Νότια Κορέα.

Στις ΗΠΑ, μια ιδιωτική εταιρεία έχει χρησιμοποιήσει ψηφιακά θερμόμετρα για να συλλέξει δεδομένα σε πραγματικό χρόνο σχετικά με συστάδες εμπύρετων ασθενειών και μια εθνική μελέτη καταγράφει τον καρδιακό ρυθμό ηρεμίας με μια εφαρμογή smartwatch, η οποία θα μπορούσε να εντοπίσει τα αναδυόμενα κρούσματα COVID-19. Αυτές οι πρωτοβουλίες είναι είτε καθοδηγούμενες από επιχειρήσεις είτε διερευνητικές και δεν ενσωματώνονται στην πολιτική και την πρακτική.

Οι συστηματικές τεχνολογίες ελέγχου είναι δαπανηρές και απαιτούν εκπαιδευμένο προσωπικό, περιορίζοντας την πρόσληψή τους σε πολλές χώρες. Η περίοδος επώασης και ο σχετικά υψηλός επιπολασμός της ασυμπτωματικής λοίμωξης σε σύγκριση με άλλες μολυσματικές ασθένειες περιορίζουν την αποτελεσματικότητα των ψηφιακών συστημάτων που ελέγχουν ζωτικά σημεία ή αυτοαναφορά συμπτωμάτων. Οι ερευνητές στο Ευρωπαϊκό Κέντρο Πρόληψης και Ελέγχου Νόσων εκτιμούν ότι η πλειονότητα των επιβατών από κινεζικές πόλεις δεν θα ανιχνευόταν σε έλεγχο λόγω αυτών των παραγόντων.

2.4 Καραντίνα και αυτοαπομόνωση

Το «αδιάκριτο» lockdown για τον έλεγχο των λοιμώξεων σε πολλές χώρες έχει σοβαρές κοινωνικοοικονομικές συνέπειες. Με την ψηφιακή τεχνολογία, η καραντίνα μπορεί να εφαρμοστεί σε άτομα που έχουν εκτεθεί ή μολυνθεί από τον ιό, με λιγότερο αυστηρούς περιορισμούς που επιβάλλονται σε άλλους πολίτες. Το σύστημα κώδικα γρήγορης απόκρισης (QR) της Κίνας, στο οποίο τα άτομα υποχρεούνται να συμπληρώσουν μια έρευνα για τα συμπτώματα και να καταγράψουν τη θερμοκρασία τους, επιτρέπει στις αρχές να παρακολουθούν την υγεία και να ελέγχουν την κίνηση. Ο κωδικός QR χρησιμεύει ως πιστοποιητικό υγειονομικής κατάστασης COVID-19 και ως ταξιδιωτικό πάσο, με χρωματικούς κωδικούς που αντιπροσωπεύουν χαμηλό, μεσαίο και υψηλό κίνδυνο. Τα άτομα με πράσινους κωδικούς επιτρέπεται να ταξιδεύουν χωρίς περιορισμούς, ενώ τα άτομα με κόκκινους κωδικούς υποχρεούνται να αυτοαπομονώνονται για 14 ημέρες. Η Κίνα χρησιμοποιεί επίσης κάμερες παρακολούθησης με τεχνητή νοημοσύνη, κάμερες που φέρουν drone και φορητές ψηφιακές συσκευές εγγραφής για την παρακολούθηση και τον περιορισμό της συγκέντρωσης ανθρώπων στο κοινό.

Στην Αυστραλία, διεθνείς ταξιδιώτες σε καραντίνα σε ξενοδοχεία κατά την άφιξη, με ταξιδιώτες από Γουχάν σε καραντίνα έξω από την ηπειρωτική Αυστραλία. Στη νέα νομοθεσία, τα άτομα που παραβιάζουν την καραντίνα θα υποχρεωθούν να φορούν συσκευές παρακολούθησης, ενώ επιβάλλονται πρόστιμα για περαιτέρω περιπτώσεις παραβίασης των περιορισμών. Στην Ταϊβάν, η ηλεκτρονική παρακολούθηση ατόμων που βρίσκονται σε καραντίνα στο σπίτι διευκολύνεται μέσω κινητών τηλεφώνων που εκδίδονται από την κυβέρνηση και παρακολουθούνται από GPS. Σε περίπτωση παραβίασης της καραντίνας, αυτός ο λεγόμενος ψηφιακός φράκτης στέλνει μηνύματα στο εκάστοτε άτομο και επιβάλλει πρόστιμα. Στη Νότια Κορέα, τα άτομα σε αυτο-απομόνωση λαμβάνουν οδηγίες να κατεβάσουν μια εφαρμογή κινητού τηλεφώνου που ειδοποιεί τις αρχές εάν εγκαταλείψουν τον τόπο απομόνωσής τους. Στο Χονγκ Κονγκ, τα άτομα που βρίσκονται σε αυτο-απομόνωση υποχρεούνται να φορούν βραχιολάκι που συνδέεται μέσω της τεχνολογίας cloud σε μια βάση δεδομένων που ειδοποιεί τις αρχές εάν παραβιαστεί η καραντίνα. Η Ισλανδία κυκλοφόρησε μια λύση κινητού τηλεφώνου για την παρακολούθηση ατόμων με COVID-19 και για να διασφαλίσει ότι θα παραμείνουν σε απομόνωση.

Οι λύσεις κινητών τηλεφώνων για την επιβολή καραντίνας μπορούν να παρακαμφθούν εάν τα άτομα εγκαταλείψουν την τοποθεσία καραντίνας χωρίς τις συσκευές τους. Οι αυτοαναφερόμενες έρευνες όπως αυτές που χρησιμοποιούνται σε συστήματα κώδικα QR λειτουργούν μόνο όταν τα άτομα είναι συμπτωματικά και αναφέρουν τα συμπτώματά τους με ακρίβεια. Ωστόσο, τέτοιες τεχνολογικές καινοτομίες

θα μπορούσαν να προσφέρουν οφέλη όταν χρησιμοποιούνται σε συνδυασμό με άλλες στρατηγικές.

2.5 Κλινική διαχείριση

Το AI μπορεί να διευκολύνει την ταχεία διάγνωση και την πρόβλεψη κινδύνου του COVID-19. Μια υπηρεσία CT (αξονικής τομογραφίας) που υποστηρίζεται από το cloud με βάση το AI χρησιμοποιείται για τον εντοπισμό περιπτώσεων πνευμονίας COVID-19 στην Κίνα. Αυτή η τεχνολογία επεξεργάζεται εικόνες CT σε δευτερόλεπτα, διαφοροποιώντας τον COVID-19 από άλλες ασθένειες των πνευμόνων και επιταχύνοντας ουσιαστικά τη διαγνωστική διαδικασία. Το COVID-Net, είναι ένας βαθύς συνελικτικός σχεδιασμός νευρικού δικτύου ανοιχτού κώδικα που διατίθεται σε κλινικούς ιατρούς σε ολόκληρο τον κόσμο και μπορεί γρήγορα να ανιχνεύσει περιπτώσεις COVID-19 από άλλες πνευμονικές παθήσεις σε ακτινογραφίες στο στήθος. Οι αλγόριθμοι μηχανικής μάθησης που αναπτύχθηκαν στην Κίνα μπορούν να προβλέψουν την πιθανότητα εμφάνισης συνδρόμου οξείας αναπνευστικής δυσχέρειας και κρίσιμης ασθένειας μεταξύ μολυσμένων ασθενών. Αυτά τα μοντέλα πρόβλεψης μπορούν να καθοδηγήσουν τη λήψη κλινικών αποφάσεων και την κατανομή πόρων, εντοπίζοντας περιοχές και νοσοκομεία που χρειάζονται πόρους κρίσιμης φροντίδας και ιατρικά εφόδια.

Οι πλατφόρμες εικονικής φροντίδας, χρησιμοποιώντας τηλεδιάσκεψη και ψηφιακή παρακολούθηση, έχουν χρησιμοποιηθεί παγκοσμίως για την παροχή απομακρυσμένης υγειονομικής περίθαλψης στους ασθενείς ως μέσο μείωσης της έκθεσής τους στον SARS-CoV-2 σε ιδρύματα υγειονομικής περίθαλψης. Στον Καναδά, οι βιντεο-επισκέψεις μεταξύ ιατρών και ασθενών, αυξήθηκαν από περίπου 1.000 επισκέψεις την ημέρα το Φεβρουάριο του 2020, σε 14.000 την ημέρα μέχρι τα μέσα Μαΐου. Χώρες όπως οι ΗΠΑ και η Αυστραλία έχουν επίσης αξιοποιήσει την ψηφιακή τεχνολογία για να παρέχουν απομακρυσμένη φροντίδα σε ασθενείς με χρόνιες παθήσεις ή με ήπια ή μέτρια ασθένεια COVID-19 στα σπίτια τους. Εάν εφαρμοστεί και υλοποιηθεί κατάλληλα, η εικονική φροντίδα μπορεί να αυξήσει την πρόσβαση στην υγειονομική περίθαλψη κατά τη διάρκεια της πανδημίας και μετά, αλλά οι πιθανοί κίνδυνοι θα μπορούσαν να περιλαμβάνουν εσφαλμένη διάγνωση, δυσλειτουργία εξοπλισμού, παραβιάσεις της ιδιωτικής ζωής και κόστος στο σύστημα υγειονομικής περίθαλψης.

Κεφάλαιο 3ο :

Ανίχνευση επαφών COVID-19 μέσω εφαρμογών smartphone

3.1 Ιχνηλάτηση Επαφών μέσω smartphone

Η ανίχνευση επαφών COVID-19 πραγματοποιείται συνήθως μέσω μιας χειροκίνητης συνέντευξης των μολυσμένων ατόμων, που διεξάγονται από τις υγειονομικές αρχές. Ο στόχος της συνέντευξης είναι να συλλέξει επαφές που είχε το μολυσμένο άτομο με άλλα άτομα τις τελευταίες 14-21 ημέρες (εφόσον προσδιορίστηκε ως η περίοδος επώασης για τον COVID-19). Οι υπεύθυνοι υγείας μπορούν στη συνέχεια να χρησιμοποιήσουν αυτές τις πληροφορίες για να υπολογίσουν μια βαθμολογία κινδύνου για κάθε μία από τις επαφές, με βάση το περιβάλλον (π.χ., σε εσωτερικούς χώρους / εξωτερικούς χώρους), τη διάρκεια και την εγγύτητα (απόσταση μεταξύ των επαφών). Ωστόσο, είναι δύσκολο για τους ανθρώπους να θυμούνται με ακρίβεια κάθε άτομο που μπορεί να έχουν συναντήσει τις τελευταίες τρεις εβδομάδες. Εκτός αυτού, ένα μολυσμένο άτομο μπορεί να έχει μολύνει πολλά άτομα τα οποία δεν μπορούν να αναγνωρίσουν, για παράδειγμα, επαφή με άγνωστα άτομα που στέκονται σε ουρά ενός σούπερ μάρκετ. Επιπλέον, πολλές επακόλουθες συνεντεύξεις απαιτούν σημαντικό εργατικό δυναμικό υπαλλήλων υγείας εκπαιδευμένο στην τέχνη της χειροκίνητης ανίχνευσης επαφών.

Σε αυτό το πλαίσιο, οι ερευνητές επικεντρώθηκαν σε τεχνολογικές λύσεις για την αυτοματοποίηση της διαδικασίας ανίχνευσης επαφών με στόχο τον γρήγορο και αξιόπιστο εντοπισμό επαφών που ενδέχεται να διατρέχουν σημαντικό κίνδυνο μόλυνσης. Η πανταχού παρουσία των smartphone και η ικανότητά τους να παρακολουθούν την τοποθεσία τους (π.χ. μέσω GPS και WiFi), μαζί με την ενσωματωμένη διεπαφή Bluetooth που επιτρέπει την επικοινωνία και τον εντοπισμό εγγύτητας με κοντινά smartphone, τα καθιστούν ιδανικές συσκευές για αυτοματοποιημένη και αξιόπιστη ανίχνευση επαφών. Ως αποτέλεσμα, έχουν προταθεί πολλές εφαρμογές παρακολούθησης επαφών smartphone, με κάποιες ήδη να έχουν αναπτυχθεί. Χρησιμοποιώντας τη διεπαφή Bluetooth, αυτές οι εφαρμογές ανίχνευσης συλλέγουν αυτόματα τα δεδομένα επαφής των χρηστών τους - δεδομένα που θα χρησιμοποιηθούν στη συνέχεια στο μέλλον σε περίπτωση που ένας χρήστης αναγνωριστεί ως μολυσμένος με COVID-19.

Η εισαγωγή των εφαρμογών παρακολούθησης επαφών οδήγησε σε μια συζήτηση σχετικά με την αρχιτεκτονική, τη διαχείριση δεδομένων, την αποτελεσματικότητα, το απόρρητο και την ασφάλειά τους. Οι περισσότερες από αυτές τις εφαρμογές ισχυρίζονται ότι διατηρούν το απόρρητο - που σημαίνει ότι δεν αποκαλύπτουν πληροφορίες προσωπικής ταυτοποίησης (PII), ταυτότητα ή πληροφορίες τοποθεσίας των επαφών χωρίς την ρητή άδεια του χρήστη. Πράγματι, οι ανησυχίες περί απορρήτου που σχετίζονται με εφαρμογές ανίχνευσης επαφών είναι ένας από τους παράγοντες που επηρεάζουν την υιοθέτησή τους. Ένα άλλο πρωταρχικό μέλημα των υπερασπιστών απορρήτου είναι ο βαθμός στον οποίο οι εφαρμογές μπορούν να αναπροσανατολιστούν για να παρακολουθούν τους χρήστες τους και πώς μπορούν να χρησιμοποιηθούν τα συλλεγόμενα δεδομένα όταν λήξει η τρέχουσα πανδημία.

3.2 Ζητήματα και προκλήσεις στην ιχνηλάτηση επαφών μέσω smartphone

Η ψηφιακή ανίχνευση επαφών επιταχύνει τη διαδικασία αναγνώρισης των ατόμων που ενδέχεται να είχαν έρθει σε στενή επαφή με τα μεταδοτικά άτομα. Ωστόσο, πριν από τη συγχώνευση της παραδοσιακής μεθοδολογίας εντοπισμού επαφών με την τρέχουσα τεχνολογία αιχμής, προκύπτουν πιθανοί κίνδυνοι και ζητήματα. Μερικές από τις πρωταρχικές ανησυχίες των οποίων είναι η διασφάλιση της ταυτότητας ενός μολυσμένου ατόμου από άλλους, η διακοπή της εξάπλωσης παραπληροφόρησης και η διακοπή της παρακολούθησης, προκαλούν πανικό μεταξύ των μαζών με τον φόβο για την δημιουργία ενός κράτους επιτήρησης. Μερικά παραδείγματα δραστηριοτήτων μαζικής παρακολούθησης μεταξύ λύσεων ανίχνευσης επαφών που ενδέχεται να έχουν σοβαρές συνέπειες είναι:

Το Ισραήλ εξέδωσε νομοθεσία που επιτρέπει στην κυβέρνηση να παρακολουθεί τα δεδομένα κινητής τηλεφωνίας ατόμων που είναι ύποπτα ότι έχουν μολυνθεί.

Από την άλλη πλευρά, η κυβέρνηση της Νότιας Κορέας έχει διατηρήσει μια δημόσια βάση δεδομένων γνωστών ασθενών, η οποία περιέχει πληροφορίες σχετικά με το επάγγελμα, την ηλικία, το φύλο και τις διαδρομές τους.

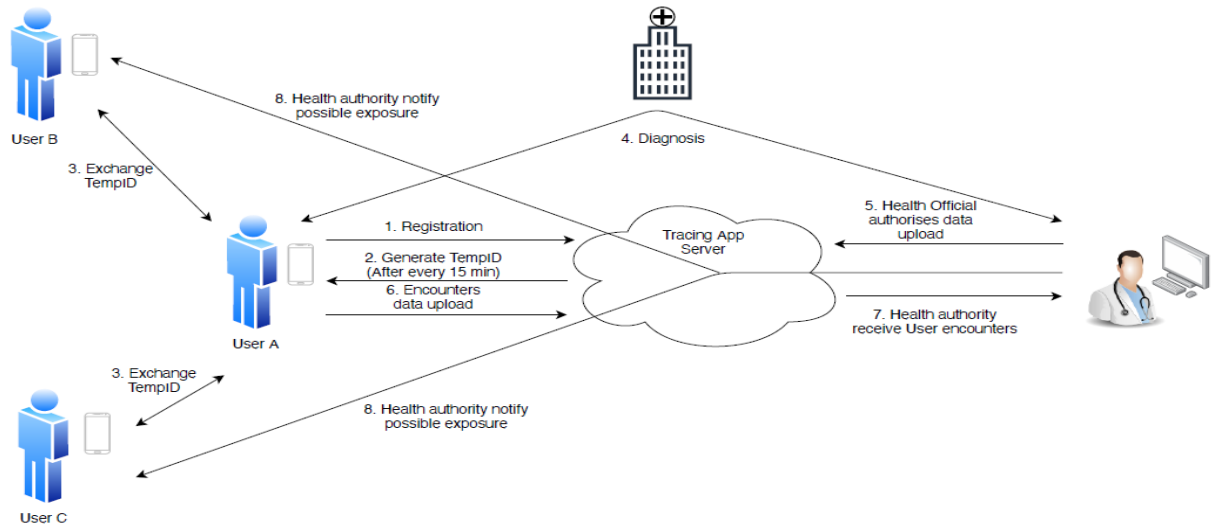
Υπάρχουν αρκετές λύσεις βασισμένες στην τεχνολογία που προτείνονται να υιοθετηθούν ως λύση / μηχανισμός παρακολούθησης ψηφιακών επαφών, μερικές από τις οποίες βασίζονται στην παρακολούθηση GPS και άλλες, που βασίζονται στο Bluetooth. Παρά τις προσπάθειες, υπάρχουν ζητήματα με τις υποκείμενες τεχνολογίες που πρέπει να κατανοηθούν, να επισημανθούν και να μετριαστούν με τους καλύτερους δυνατούς τρόπους για την ενίσχυση των διαθέσιμων λύσεων. Αυτά τα ζητήματα θα πρέπει να αντιμετωπιστούν για να περιορίσουν κάθε στρατηγικό παράθυρο ευκαιριών για κακόβουλους παράγοντες, κατασκοπευτείς και για επιτήρηση του κράτους / κυβέρνησης. Τα συστήματα αυτόματης παρακολούθησης επαφών που βασίζονται σε επικοινωνίες Bluetooth προτάθηκαν για πρώτη φορά από τους Altuwaiyan κ.α. το 2018. Τα συστήματα ανίχνευσης επαφών που βασίζονται σε Bluetooth μπορούν να ανιχνεύσουν άμεσα εάν οι χρήστες ήρθαν κοντά. Η εγγύτητα μπορεί να προσεγγιστεί από την ισχύ του σήματος, το οποίο όμως μειώνεται και από εμπόδια όπως οι τοίχοι. Επομένως, σε περιβάλλον υψηλού κινδύνου για στενή επαφή, όπως κτίρια ή δημόσιες συγκοινωνίες, μπορεί να αντικατοπτρίζει πιο αποτελεσματικά και με ακρίβεια τη λειτουργική εγγύτητα. Ωστόσο, με εφαρμογές που αξιολογούν τον κίνδυνο έκθεσης με βάση το Bluetooth, η ανταλλαγή εγγύτητας στην ουσία δεν επαρκεί λόγω του γεγονότος ότι εκτός από την αλληλεπίδραση ανθρώπου με άνθρωπο, ο κορονοϊός (COVID-19) μπορεί επίσης να μεταδωθεί μέσω κοινών περιβαλλόντων ή κοινώς αγγιζόμενων επιφανειών. Ένα άλλο σημαντικό μειονέκτημα των συστημάτων που βασίζονται στο Bluetooth είναι το πρόβλημα του αργού ή χαμηλού ρυθμού υιοθέτησης που με τη σειρά του περιορίζει τη βάση χρηστών επηρεάζοντας έτσι την αποτελεσματικότητα του συστήματος. Οι Zeadally κ.α. έχουν πραγματοποιήσει μια λεπτομερή τοποθέτηση για ζητήματα και πιθανές επιθέσεις στην τεχνολογία Bluetooth. Το GPS, από την άλλη πλευρά, δεν είναι ασφαλές από την εγγενή φύση του. Επίσης, υπάρχουν ορισμένες λειτουργίες που δεν μπορούν να παρέχουν συστήματα με βάση το GPS. Μία από τις κύριες ανησυχίες είναι οι spoofing επιθέσεις, όπου ένα spoofer δημιουργεί ένα λανθασμένο σήμα GPS με εσφαλμένο χρόνο και τοποθεσία σε έναν συγκεκριμένο δέκτη. Ο Warner κ.α. έκαναν μια απλή επίδειξη για να δείξουν ότι το GPS είναι ευάλωτο σε πλαστογράφηση.

3.3 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.

3.3.1 Κεντρική Αρχιτεκτονική

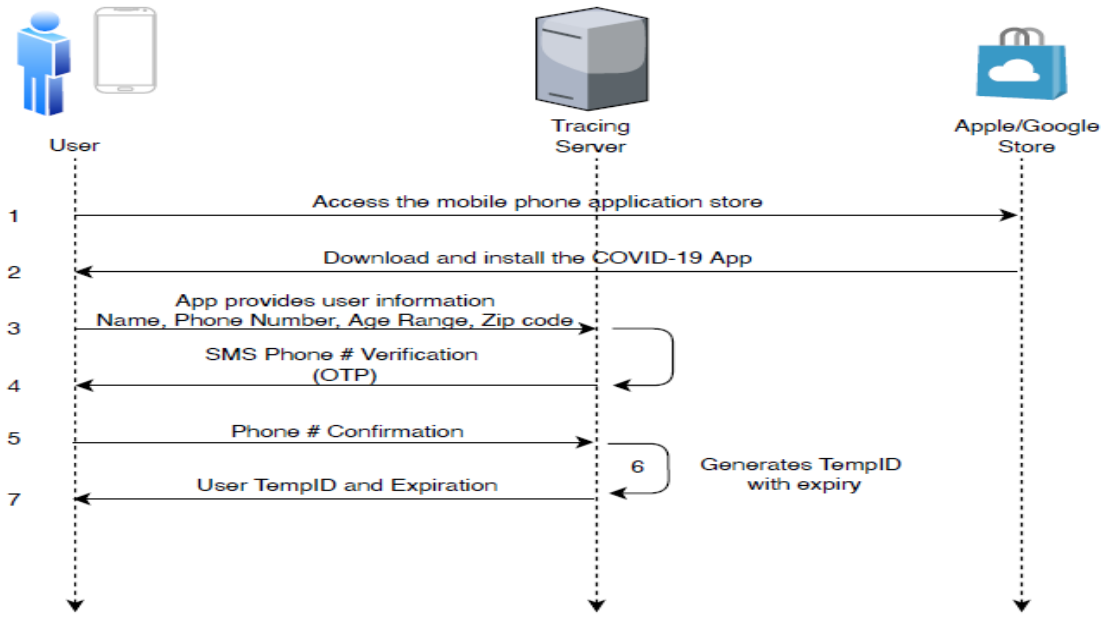
Η εικόνα 1 δείχνει τις κύριες οντότητες και αλληλεπιδράσεις μιας κεντρικής αρχιτεκτονικής. Σημειώνουμε ότι η κεντρική αρχιτεκτονική που περιγράφουμε βασίζεται στο πρωτόκολλο Bluetrace. Η αρχική απαίτηση για την εφαρμογή είναι ότι ένας χρήστης πρέπει να προ-εγγραφεί στον κεντρικό διακομιστή. Ο διακομιστής 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*

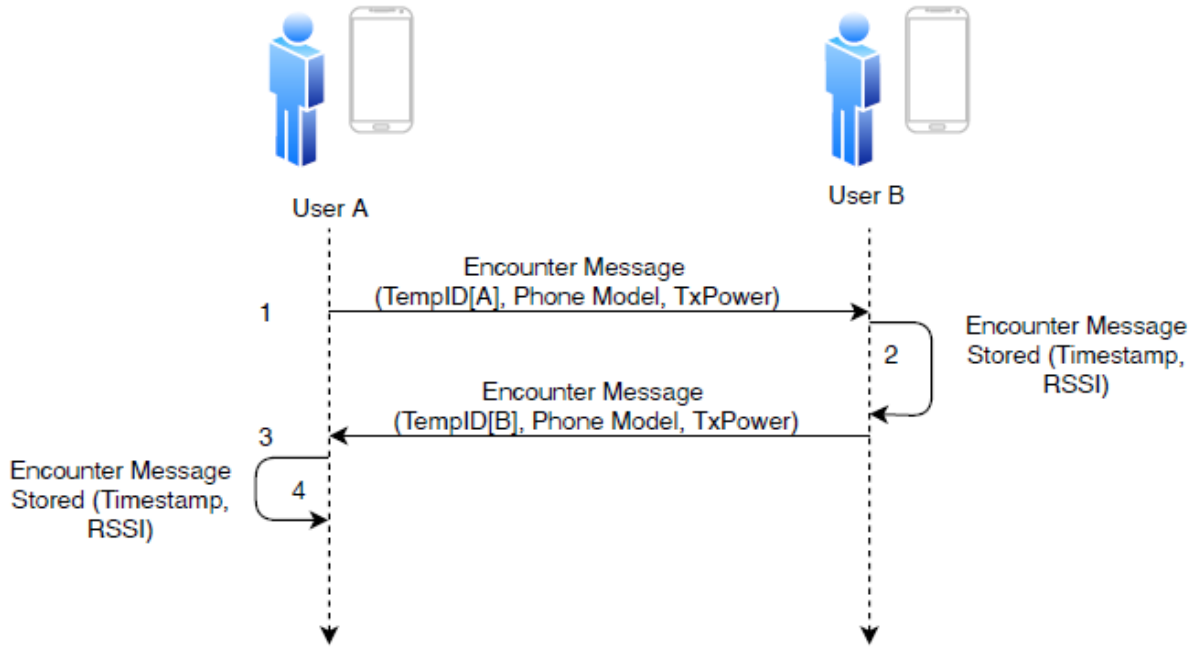
Η εικόνα 2 δείχνει τα βήματα που απαιτούνται για την εγγραφή ενός χρήστη σε μια κεντρική αρχιτεκτονική. Ένας χρήστης κάνει λήψη της εφαρμογής (βήματα 1 και 2) και καταχωρεί λεπτομέρειες όπως όνομα, αριθμό κινητού τηλεφώνου, ηλικιακή ομάδα και ταχυδρομικό κώδικα στον διακομιστή (βήμα 3). Ο διακομιστής 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

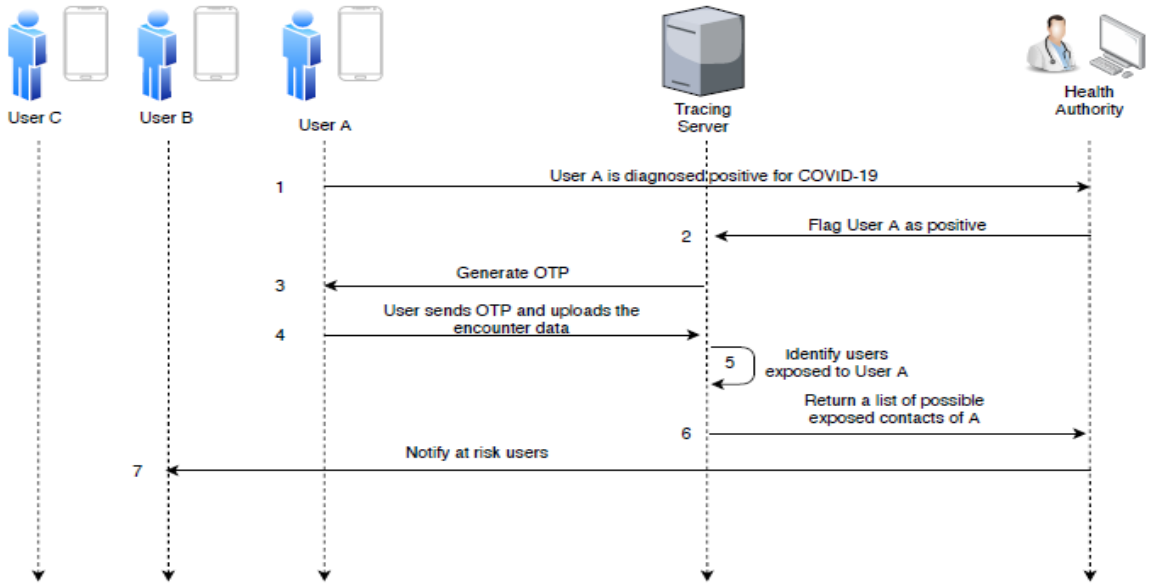
3) *Μεταφόρτωση δεδομένων συναντήσεων*

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

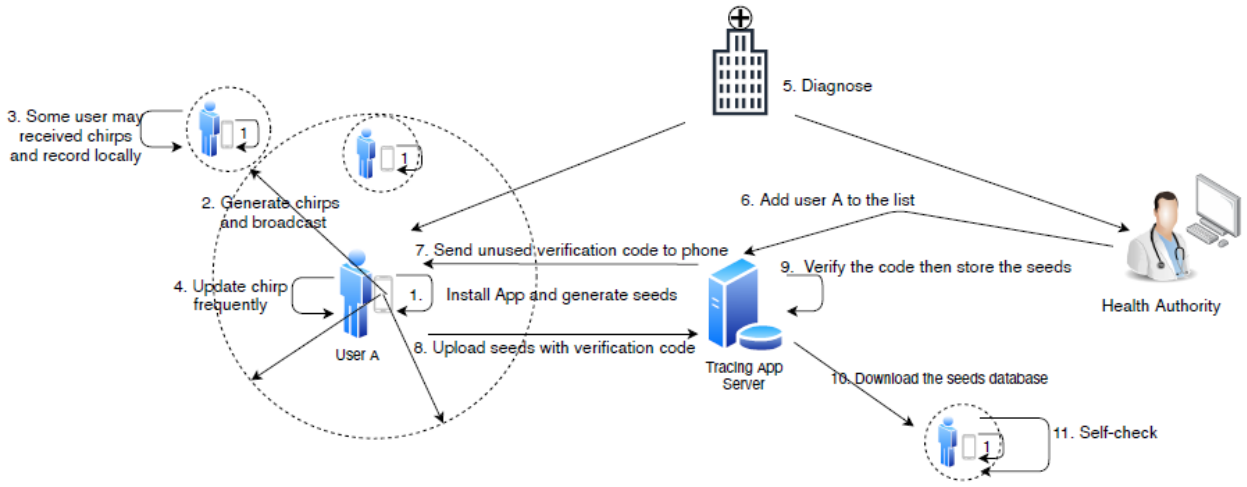
Συνοψίζοντας, στην κεντρική αρχιτεκτονική, ο κεντρικός διακομιστής παίζει βασικό ρόλο στην εκτέλεση βασικών λειτουργιών, όπως η αποθήκευση κρυπτογραφημένων πληροφοριών PII, η δημιουργία ανώνυμων TempIDs, η ανάλυση κινδύνου και οι ειδοποιήσεις για στενές επαφές. Αυτή η συσσώρευση ευθυνών εγείρει ανησυχίες 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.

3.3.2 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 1η 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.



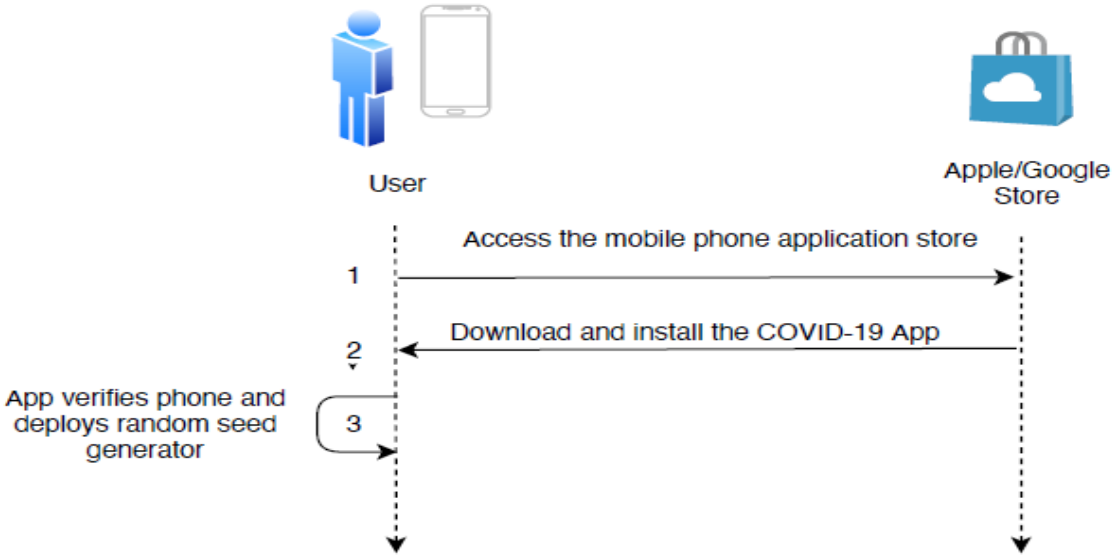
Εικόνα 5: Αποκεντρωμένη Αρχιτεκτονική Εφαρμογών Ιχνηλάτησης

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).

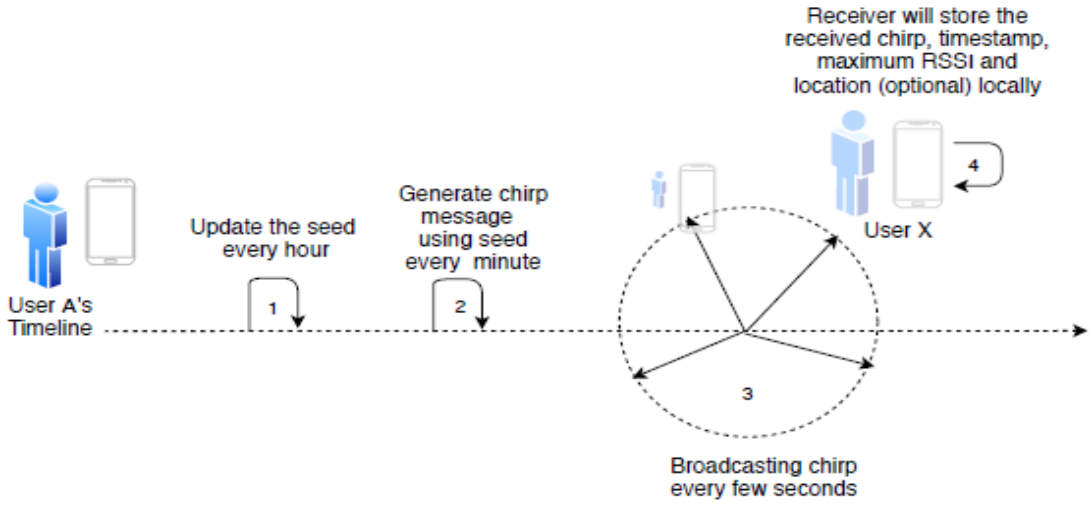


Εικόνα 6: Διαδικασία Εγκατάστασης Εφαρμογής Ιχνηλάτησης Αποκεντρωμένης Αρχιτεκτονικής

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

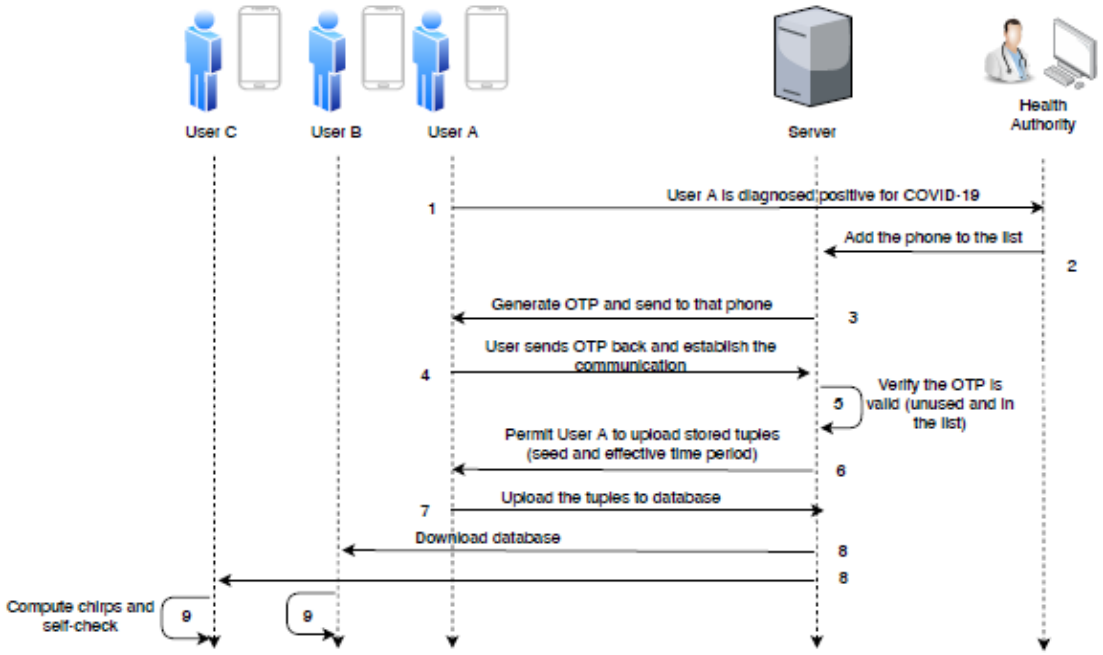
εξ' ορισμού, είναι ανώνυμα. Έπειτα, η εφαρμογή δημιουργεί νέα "chirps" με χρονικό βαθμό λεπτομέρειας το 1 λεπτό. Αυτά μεταδίδονται κάθε λίγα δευτερόλεπτα μέσω του σήματος Bluetooth. Στο τηλέφωνο του «ακροατή», η εφαρμογή αποθηκεύει αυτόματα όλα τα "chirps" που λαμβάνονται (βήμα 4 στην εικόνα 7). Οι πληροφορίες 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

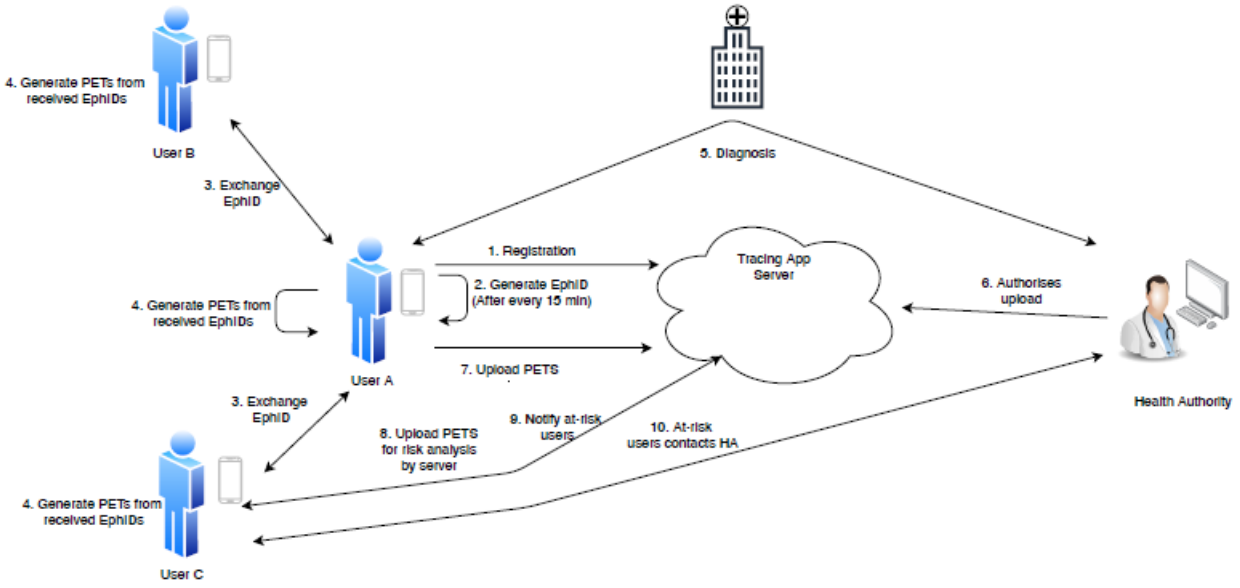
4) *Η διαδικασία ανίχνευσης επαφών*

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.

3.3.3 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.

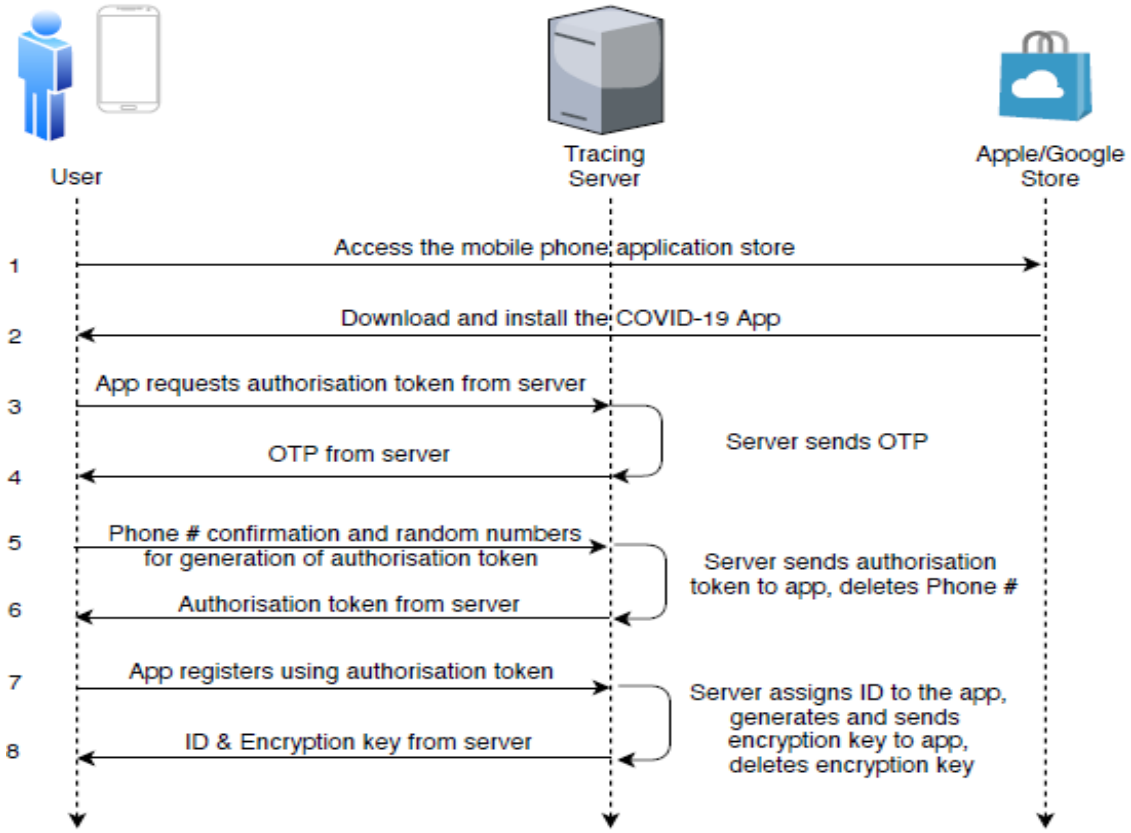
Η εικόνα 9 δείχνει την αλληλουχία αλληλεπιδράσεων στην υβριδική αρχιτεκτονική με βάση το πρωτόκολλο Desire. Αυτό το πρωτόκολλο απαιτεί από τη διαδικασία εγγραφής της εφαρμογής του χρήστη να εκχωρήσει ένα μοναδικό αναγνωριστικό συσκευής χωρίς την καταγραφή PII. Στη συνέχεια, οι συσκευές δημιουργούν κρυπτογραφικά 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*

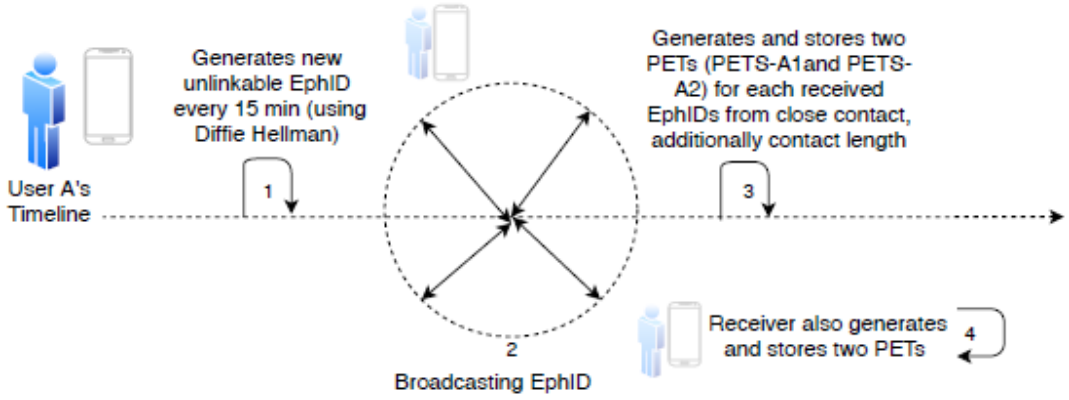
Η διαδικασία εγγραφής στην υβριδική αρχιτεκτονική απαιτεί μια διαδικασία ελέγχου ταυτότητας δύο βημάτων, όπου ο αριθμός τηλεφώνου επαληθεύεται από το OTP και η εφαρμογή επαληθεύεται μέσω ενός διακριτικού εξουσιοδότησης που εκδίδεται από τον διακομιστή. Η εικόνα 10 παρουσιάζει την λεγόμενη διαδικασία. 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*

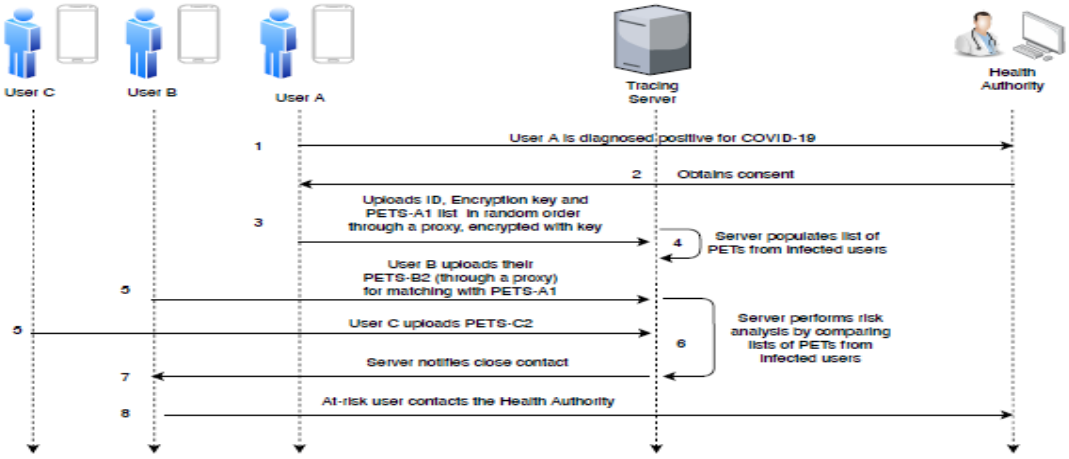
Στη φάση λειτουργίας, η συσκευή δημιουργεί ένα νέο Ephemeral ID (EphID) χρησιμοποιώντας τον μηχανισμό ανταλλαγής κλειδιών Diffie Hellman, ο οποίος διαρκεί για συνήθως 15 λεπτά και συγχρονίζεται με το διάστημα περιστροφής της διεύθυνσης Bluetooth MAC. Η συσκευή ξεκινά τη μετάδοση αυτού του EphID μέσω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

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

4. *“Πανδημία κορονοϊού 2019–20”, (άρθρο wiki),* Wikipedia*,* 2020*.* Διαθέσιμο από: εδώ.

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()
}
```

- ***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.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, 5000)
            } else {
                Toast.makeText(applicationContext, "Check Completed !",
Toast.LENGTH_SHORT).show()
            }
        }
    }, 5000)

    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.content.Context
import android.content.SharedPreferences
import android.content.pm.PackageManager
import android.os.Bundle
import android.text.method.ScrollingMovementMethod
import android.view.WindowManager
import android.widget.Toast
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.google.firebase.database.FirebaseDatabase
import com.example.coronatracker.ContactListActivity.Companion.MyPREFS
import com.example.coronatracker.sample.R
import com.example.coronatracker.sample.databinding.ActivityTrackingBinding
import java.util.UUID


@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 closeToMe: CloseToMe? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        //keeping screen on
        window.addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON)
        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


                                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()
                                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()
        super.onPause()
    }

    companion object {
        const val PERMISSIONS_REQUEST = 1010

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

- ***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>
```