ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Πρωτοκόλλου Proof-of-Location για κινητές συσκευές με εφαρμογή στο Ethereum Blockchain

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Δ. Μητρόπουλος

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Ιούλιος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΟΙΚΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Πρωτοκόλλου Proof-of-Location για κινητές συσκευές με εφαρμογή στο Ethereum Blockchain

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Κωνσταντίνος Δ. Μητρόπουλος

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Ιουλίου 2019.

.....	.....	.....
Θεοδώρα Βαρβαρίγου	Εμμανουήλ Βαρβαρίγος	Συμεών Παπαβασιλείου
Καθηγήτρια Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2019

.....

Κωνσταντίνος Δ. Μητρόπουλος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η διείσδυση των έξυπνων κινητών συσκευών στην καθημερινότητα των ανθρώπων, έχει οδηγήσει και σε σημαντικότατη αύξηση των υπηρεσιών που βασίζονται στην τοποθεσία του χρήστη. Εφαρμογές που βασίζονται σε τέτοιες υπηρεσίες ποικίλουν, από αυτές επιχειρήσεων που επιθυμούν να επιβραβεύσουν τακτικούς πελάτες, μέχρι αυτές τραπεζών που επιθυμούν να παρακολουθήσουν την τοποθεσία των κατόχων καρτών, ώστε να τους προστατέψουν από ύποπτες συναλλαγές.

Είναι φανερό λοιπόν ότι υπηρεσίες σαν και αυτές βασίζονται στην ειλικρίνεια των χρηστών, οι οποίοι όταν έχουν όφελος από την παρουσία τους σε κάποια συγκεκριμένη τοποθεσία, έχουν και κίνητρο να αναφέρουν ψευδή τοποθεσία στις υπηρεσίες. Αυτό το φαινόμενο έχει οδηγήσει στη δημιουργία πολλών πρωτοκόλλων απόδειξης τοποθεσίας των χρηστών, με σκοπό την αποφυγή ψευδών δηλώσεων τοποθεσίας και επομένως την πιο ασφαλή λειτουργία των υπηρεσιών που βασίζονται σε αυτή.

Παράλληλα, την εμφάνιση τους έχουν κάνει τεχνολογίες τύπου Blockchain. Με αρχή την εμφάνιση του Bitcoin, ένα κρυπτονόμισμα που βασίζεται σε τέτοιες τεχνολογίες, η τεχνολογία εξελίσσεται συνεχώς και το πεδίο εφαρμογής της γίνεται όλο και πιο ευρύ. Πλέον, φαίνεται να έχει παρουσιαστεί σημαντική άνοδος δημιουργίας και λειτουργίας αποκεντροποιημένων εφαρμογών (DApps - Decentralized Applications), που βασίζονται δηλαδη σε ένα κατανεμημένο δίκτυο ομότιμων κόμβων (P2P Network - peer-to-peer network) και όχι στους εξυπηρετητές κάποιας κεντρικής οντότητας. Το πιο χαρακτηριστικό παράδειγμα Blockchain που προωθεί την ανάπτυξη τέτοιων εφαρμογών, είναι το Ethereum Blockchain.

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη ενός πρωτοκόλλου απόδειξης τοποθεσίας που μπορεί να χρησιμοποιηθεί από το χρήστη οποιουδήποτε κινητού τηλεφώνου. Βασίζεται στην ύπαρξη γειτόνων που μπορούν να επικοινωνήσουν με το χρήστη και να επαληθεύσουν την τοποθεσία του λόγω της μικρής εμβέλειας της τεχνολογίας του Bluetooth και ενός ειδικού μηχανισμού που απαγορεύει σε γειτονικές συσκευές να παρέχουν με απόδειξη τοποθεσίας τον ίδιο χρήστη πολλές φορές.

Το προτεινόμενο πρωτόκολλο υλοποιείται σε εφαρμογή τύπου blockchain στο Ethereum που δίνει τη δυνατότητα μεταφοράς κρυπτονομισμάτων υπό την προϋπόθεση ότι πρώτα έχει αποδειχθεί και επαληθευθεί η τοποθεσία του χρήστη που επιθυμεί την μεταφορά αυτή.

Λέξεις Κλειδιά: Απόδειξη Τοποθεσίας, Blockchain, Ethereum, Αποκεντρωμένες Εφαρμογές, Bluetooth, Κινητές Συσκευές

Abstract

The appearance of smart mobile devices into people's everyday lives has led to a significant increase in location-based services. Applications that depend on such services vary from those of businesses that wish to reward regular customers to those of banks wishing to track the location of cardholders in order to protect them from suspicious transactions.

It is obvious, thus, that services like these depend on the honesty of the users who, when they benefit from their presence in a particular location, have an incentive to report a false location to the services. This phenomenon has led to the creation of many Proof of Location protocols that make it difficult for users to claim false location and therefore can provide safer operation of the services that depend on it.

At the same time, many Blockchain technologies have made their appearance. Beginning with the emergence of Bitcoin, a cryptocurrency based on such technologies, Blockchain technologies are constantly evolving and their scope is becoming wider. Today, there appears to have been a significant rise in the creation and operation of decentralized applications (DApps - Decentralized Applications) that are based on a distributed peer-to-peer network (P2P Network) and not on the servers of a central entity. The most typical Blockchain example that promotes the development of such applications is the Ethereum Blockchain.

The purpose of this diploma thesis is to develop a Proof of Location protocol that can be used by the user of any smartphone. It is based on the existence of neighbours who can communicate with the user and verify his location thanks to the small range of Bluetooth technology and a special decaying mechanism that prohibits neighbouring devices from providing Proofs of Location to the same user multiple times .

The proposed protocol is implemented in a blockchain application in Ethereum that enables the transfer of cryptocurrencies, provided that the location of the user who wants to make the transfer, is first proven and verified.

Keywords: Proof of Location, Blockchain, Ethereum, Decentralized apps, Bluetooth, Smartphones

Ευχαριστίες

Με την ευκαιρία της ολοκλήρωσης της διπλωματικής μου εργασίας, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες προς την κ. Θεοδώρα Βαρβαρίγου, Καθηγήτρια Ε.Μ.Π., για τη δυνατότητα που μου παρείχε να εργαστώ ερευνητικά στον τομέα των Blockchain.

Ακόμη, θα ήθελα να ευχαριστήσω ιδιαίτερα τον Ορφέα Βουτυρά για την επίβλεψη και την πολύτιμη συμβολή του στην εκπόνηση αυτής της διπλωματικής εργασίας. Οι κατευθύνσεις που παρείχε και το διαρκές ενδιαφέρον που έδειχνε υπήρξαν κρίσιμες.

Έπειτα, θα ήθελα να ευχαριστήσω και τα υπόλοιπα μέλη του εργαστηρίου Distributed Knowledge and Media Systems Group του τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ε.Μ.Π.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την αγάπη και στήριξη της καθ' όλη τη διάρκεια των σπουδών μου.

Περιεχόμενα

Κατάλογος Σχημάτων

1. Εισαγωγή

Ο άνθρωπος είχε από την αρχαιότητα την ανάγκη γνώσης της τοποθεσίας του. Με αυτόν τον τρόπο ανέπτυξε την αστρονομία και την τοπογραφία και κατάφερε να χρησιμοποιήσει τις επιστήμες αυτές για να κατακτήσει τις θάλασσες και τον ουρανό. Με την πρόοδο της τεχνολογίας και την ευρεία χρήση των κινητών τηλεφώνων, μπορεί πλέον να αποκτήση την ακριβή γεωγραφική του θέση οποιαδήποτε στιγμή το επιθυμήσει.

Η εξαιρετικά εκτενής χρήση των κινητών τηλεφώνων καθιστά πλέον εύκολο τον εντοπισμό των χρηστών τους. Όλες αυτές οι συνθήκες ετοίμασαν το έδαφος για τη δημιουργία υπηρεσιών που απαιτούν τη γνώση της τοποθεσίας του χρήστη. Τέτοιες υπηρεσίες έχουν μεγάλο εύρος εφαρμογών, από την εμφάνιση προτάσεων ανάλογα με την τοποθεσία του χρήστη, στην επιβράβευση τακτικών πελατών από επιχειρήσεις, μέχρι και την πρόσβαση σε απόρρητα έγγραφα μόνο όταν ο χρήστης βρίσκεται σε εξουσιοδοτημένη περιοχή. Σε όλες τις παραπάνω εφαρμογές η τοποθεσία αποκτά καθοριστικό ρόλο στην αλληλεπίδραση του ανθρώπου με τις συσκευές.

Όπως είναι φυσικό τέτοιες υπηρεσίες βασίζονται στην ειλικρίνεια του χρήστη. Επειδή πολλές φορές αυτός έχει κίνητρο να τροποποιήσει την τοποθεσία του, δηλαδή να πει ψέματα στην υπηρεσία, ενδεχομένως για να αποκτήσει κάποια οφέλη από αυτήν, απαιτείται η δημιουργία κάποιου πρωτοκόλλου το οποίο να εξασφαλίζει ότι ο χρήστης λέει την αλήθεια. Έχει ήδη αναπτυχθεί πληθώρα πρωτοκόλλων Απόδειξης Τοποθεσίας (Proof of Location protocols), επιτρέποντας στις διάφορες εφαρμογές την επιβεβαίωση της τοποθεσίας του χρήστη.

1.1 Σκοπός της εργασίας

Η εργασία αυτή έχει ως σκοπό τη μελέτη και τη δημιουργία ενός πρωτοκόλλου απόδειξης της τοποθεσίας (Proof of Location), που θα μπορεί να χρησιμοποιηθεί σε οποιαδήποτε εφαρμογή. Στο πλαίσιο της εργασίας αυτής προσπαθούμε να υλοποιήσουμε την εφαρμογή του πρωτοκόλλου με σκοπό να αποδείξουμε την τοποθεσία του χρήστη κατά τη διάρκεια μιας συναλλαγής σε Ethereum Blockchain.

Συγκεκριμένα, ο χρήστης ο οποίος θέλει να κάνει μια συναλλαγή στο Ethereum χρειάζεται να αποδείξει την τοποθεσία του. Στέλνει τη συντεταγμένες σε άλλους χρήστες που βρίσκονται γύρω του με τη χρήση του Bluetooth του κινητού του. Αυτοί με τη σειρά τους είτε θα αποδεχτούν την τοποθεσία αν είναι σωστή, είτε θα την απορρίψουν στην αντίθετη περίπτωση και θα στείλουν την απάντηση τους στο Blockchain. Το Blockchain με τη σειρά του είτε θα αποδεχτεί τη συναλλαγή στέλνοντας τα χρήματα που είχε δεσμεύσει

στον τελικό αποδέκτη, αν η απόδειξη ήταν θετική, είτε θα την απορρίψει αν ήταν αρνητική επιστρέφοντας τα χρήματα πίσω στον αποστολέα.

1.2 Δομή Περιεχομένου

Η παρούσα εργασία είναι χωρισμένη σε 6 κεφάλαια. Αρχικά στο πρώτο κεφάλαιο γίνεται η εισαγωγή στη διπλωματική εργασία και στη δομή της. Στο δεύτερο κεφάλαιο αναλύεται το απαραίτητο θεωρητικό υπόβαθρο για την κατανόηση της εργασίας, δηλαδή το θεωρητικό υπόβαθρο για τις τεχνολογίες Blockchain και τις Αποδείξεις Ταυτότητας. Στο τρίτο κεφάλαιο θα γίνει αναλυτική επεξήγηση του προτεινόμενου πρωτοκόλλου Proof-of-Location το οποίο καλείται να λύσει ορισμένα υπάρχοντα προβλήματα. Στο τέταρτο κεφάλαιο θα γίνει μια σύντομη περιγραφή της εφαρμογής και θα παρουσιαστούν οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της. Στο πέμπτο κεφάλαιο θα παρουσιαστεί αναλυτικά ο σχεδιασμός και η υλοποίηση της εφαρμογής, διάφορες παρατηρήσεις επί της εφαρμογής, οι οδηγίες εγκατάστασης και χρήσης της, καθώς και τα συμπεράσματα που προκύπτουν. Στο έκτο κεφάλαιο δίνονται κατευθύνσεις για μελλοντική εργασία και βελτιώσεις πάνω στο πρωτόκολλο που προτείνεται. Στο έβδομο κεφάλαιο θα παρουσιαστεί η βιβλιογραφία στην οποία βασίστηκε η διπλωματική εργασία.

2. Θεωρητικό υπόβαθρο

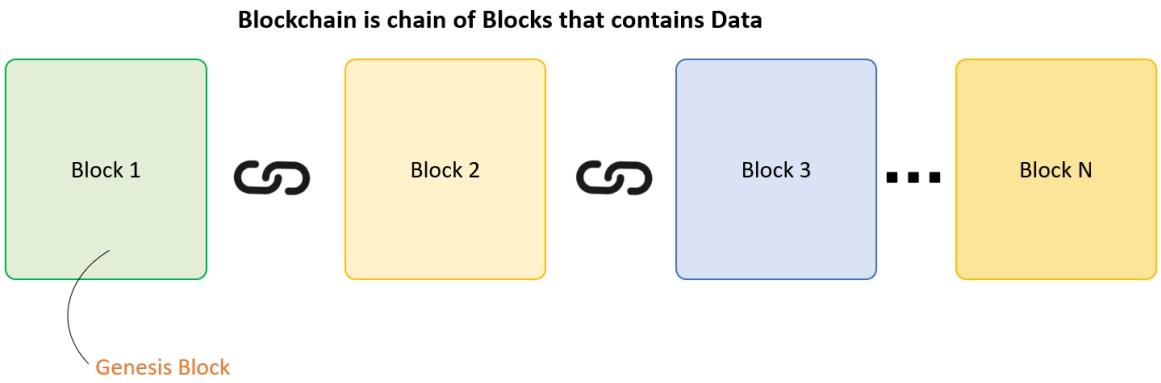
Για την κατανόηση της εργασίας είναι απαραίτητη γνώση και κατανόηση ορισμένων εννοιών και τεχνολογιών. Συγκεκριμένα θα εξηγήσουμε αναλυτικά τι είναι Proof-of-Location και γιατί είναι απαραίτητο, ύστερα θα αναλύσουμε τι είναι Blockchain και πιο συγκεκριμένα το Ethereum και τέλος θα δούμε γιατί αποδείξεις τύπου Proof-of-Location είναι απαραίτητες σε εφαρμογές σε αυτές τις τεχνολογίες.

2.1 Blockchain

Το Blockchain είναι μια τεχνολογία που εμφανίστηκε ως ιδέα για πρώτη φορά το 1991 από τους Stuart Haber και W. Scott Stornetta [1], οι οποίοι ήθελαν να φτιάξουν ένα σύστημα στο οποίο τα στοιχεία δε θα μπορούν να αλλαχθούν.

Ο πρώτος (ή οι πρώτοι) που υλοποίησε αυτήν την ιδέα ήταν ο Satoshi Nakamoto το 2008, όταν δημιούργησε το παγκοσμίως γνωστό Bitcoin. Ο Nakamoto βελτίωσε την ιδέα των Haber και Stornetta και την εφάρμοσε για το κρυπτονόμισμα Bitcoin, όπου το blockchain αποτελεί ένα "βιβλίο" όπου καταγράφονται όλες οι συναλλαγές του δικτύου. Πολλές νέες μορφές Blockchain πλατφορμών εμφανίστηκαν μετά το Bitcoin, με πιο γνωστή το Ethereum.

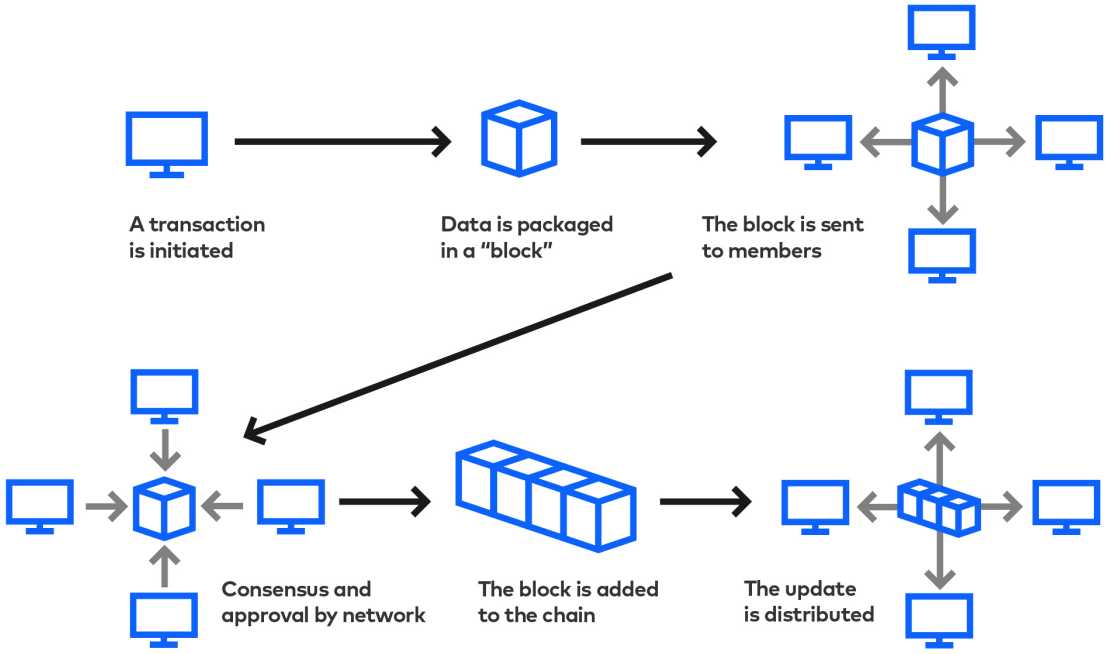
2.1.1 Τι είναι το Blockchain



Σχήμα 2.1: Παράδειγμα αλυσίδας Blockchain

Το Blockchain είναι μια κατανεμημένη δομή δεδομένων σε ένα δίκτυο όπου δεν υπάρχει κεντρική οντότητα που να είναι υπεύθυνη για τη διαχείριση των δεδομένων που καταγράφονται [2]. Αποτελεί μια σειρά από αμετάβλητα δεδομένα, τα οποία τα

διαχειρίζονται πολλές συστάδες υπολογιστών και όχι μια συγκεκριμένη οντότητα. Κάθε μπλοκ πληροφοριών είναι ασφαλές και εξαρτώμενο από άλλα μπλοκ με τη χρήση κρυπτογραφίας, δημιουργώντας έτσι αλυσίδες από μπλοκς, από όπου προκύπτει και το όνομα Blockchain.



Σχήμα 2.2: Παράδειγμα αλυσίδας Blockchain

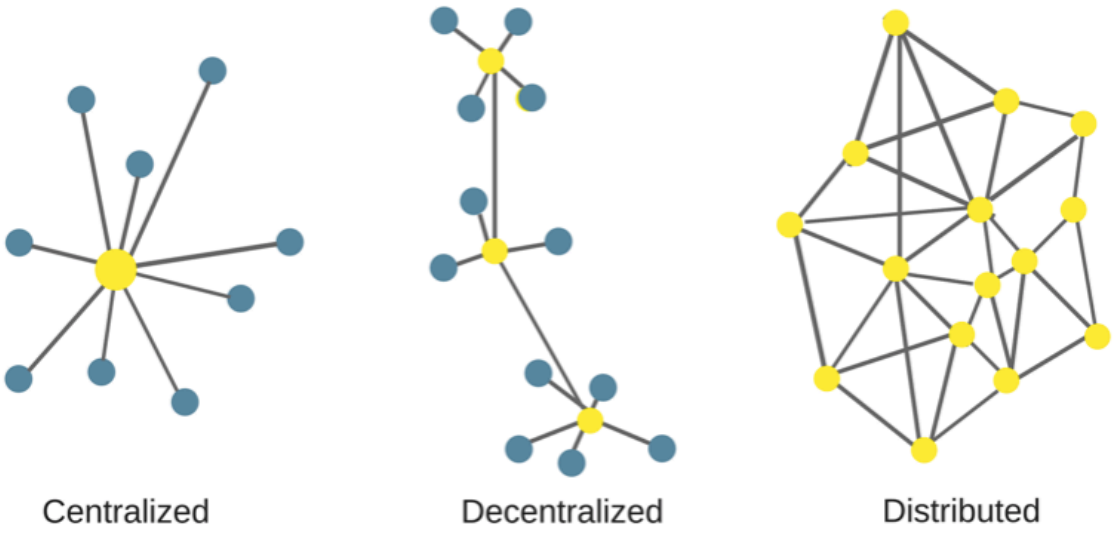
Η ιδέα του Blockchain αποτελεί έναν τρόπο μεταφοράς δεδομένων από τον χρήστη Α στον χρήστη Β με μια απόλυτα αυτοματοποιημένη και ασφαλή διαδικασία. Κατά τη δημιουργία κάθε μπλοκ χιλιάδες ή και εκατομμύρια υπολογιστές σε ολόκληρο το δίκτυο επικυρώνουν αυτό το μπλοκ. Το επικυρωμένο μπλοκ προστίθεται στην αλυσίδα, η οποία είναι μοναδική. Η παραποίηση των δεδομένων μιας καταγραφής, προϋποθέτει την παραποίηση ολόκληρης της αλυσίδας, το οποίο είναι πρακτικά αδύνατο, διότι κάτι τέτοιο θα απαιτούσε πρόσβαση σε ολόκληρο το δίκτυο.

Το Bitcoin χρησιμοποιεί αυτήν την τεχνολογία αποκλειστικά και μόνο για χρηματοοικονομικές συναλλαγές. Υπάρχουν όμως και άλλες εφαρμογές της τεχνολογίας του Blockchain που επιτρέπουν την ανάπτυξη διαφόρων εφαρμογών. Ένα τέτοιο Blockchain είναι το Ethereum.

2.1.2 Αποκεντροποίηση

Ένα πολύ σημαντικό χαρακτηριστικό που καθιστά το Blockchain εξαιρετικά σημαντικό είναι η αποκεντροποίηση [2]. Αποθηκεύοντας πληροφορία σε ένα δίκτυο P2P

(peer-to-peer network), το Blockchain μπορεί να αντιμετωπίσει ένα μεγάλο αριθμό κινδύνων που θα εμφανίζονταν αν αποθηκεύουμε τα πάντα σε ένα κεντρικό κόμβο.



Σχήμα 2.3: Κεντροποιημένο, αποκεντροποιημένο και κατανεμημένο δίκτυο

Αρχικά με αυτόν τον τρόπο εξαλείφεται ο κίνδυνος κάποιας βλάβης του κεντρικού κόμβου. Επίσης είναι δυσκολότερες οι επιθέσεις απο κακοποιά στοιχεία καθώς δεν μπορούν να επιτεθούν σε κάποια κεντρική οντότητα, αλλά αντίθετα σε έναν τεράστιο αριθμό κόμβων-υπολογιστών. Επιπρόσθετα, με τη χρήση αλγορίθμων κρυπτογραφίας δημοσίου κλειδιού, καθιστά στην ουσία τις πληροφορίες που είναι αποθηκευμένες στην αλυσίδα αμετάβλητες.

Κάθε κόμβος στο αποκεντροποιημένο σύστημα κατέχει ένα αντίγραφο του blockchain. Με αυτόν τον τρόπο προστατεύεται η εγκυρότητα των δεδομένων που διατηρείται από την τεράστια αναπαραγωγή αντιγράφων. Κάθε κόμβος και κατ'επέκταση κάθε χρήστης έχει τα ίδια δικαιώματα.

2.1.3 Κρυπτογραφία ελλειπτικής καμπύλης

Ένας άλλος παράγοντας που κάνει τις αλυσίδες τύπου Blockchain εξαιρετικά ελκυστικές είναι η ασφάλεια που προσφέρουν στην οποιαδήποτε μεταφορά δεδομένων κατά τις συναλλαγές. Το Bitcoin και το Ethereum χρησιμοποιούν κρυπτογραφία ελλειπτικής καμπύλης (Elliptic curve cryptography - ECC) για την υπογραφή συναλλαγών [3].

Η κρυπτογραφία ελλειπτικής καμπύλης είχε προταθεί από τους μαθηματικούς Neal Koblitz και Victor S. Miller, ανεξαρτήτως, το 1985. Παρόλα αυτά δεν χρησιμοποιήθηκε

μέχρι τις αρχές του 2000 όπου με την εμφάνιση και την εδραίωση του Internet πολλές κυβερνήσεις και πάροχοι Internet αρχισαν να τη χρησιμοποιούν ως μέθοδο κρυπτογραφίας.

Το ECC, συγκριτικά με το εξαιρετικά δημοφιλές RSA, προσφέρει ένα πολύ σημαντικό πλεονέκτημα. Το μέγεθος του κλειδιου για το ECC είναι εξαιρετικά μικρότερο από αυτό που χρειάζεται για κρυπτογράφηση RSA, παρέχοντας όμως τον ίδιο βαθμό ασφάλειας. Το ECC δεν χρησιμοποιείται ακόμα όσο το RSA, αλλά αποτελεί ουσιαστικά μια πιο αποτελεσματική μορφή του RSA και για αυτό χρησιμοποιείται εκτενώς σήμερα σε διάφορα κρυπτονομίσματα.

2.1.4 Εφαρμογές Blockchain

Το Blockchain είναι μια τεράστια βάση δεδομένων, η οποία είναι αποκεντροποιημένη, εξαιρετικά ασφαλής και όλα τα δεδομένα της είναι αμετάβλητα. Αυτό το καθιστά μια τεχνολογία που βρίσκει πάρα πολλές εφαρμογές που λύνουν διαφορετικά προβλήματα.

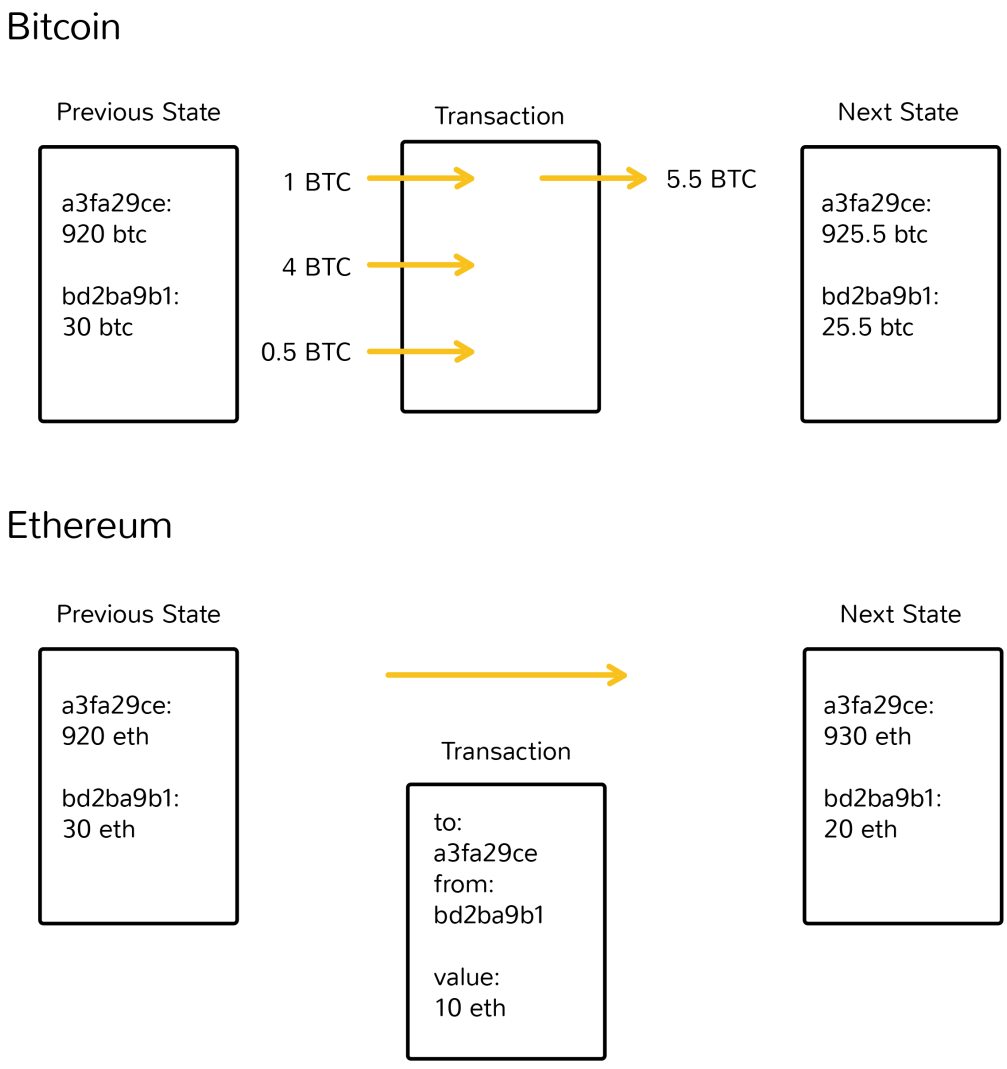
Η πρώτη εφαρμογή που έχει ήδη αναφερθεί είναι τα κρυπτονομίσματα. Το Bitcoin, το Ethereum και το Litecoin είναι μόνο μερικές από τις υπάρχουσες εναλλακτικές χρήματος που προσφέρουν το πλεονέκτημα της αποκεντροποίησης και δεν εξαρτώνται από μια κεντρική τράπεζα. Στις 8 Μαΐου του 2018 μάλιστα, το γνωστό μέσο κοινωνικής δικτύωσης Facebook ανακοίνωσε ότι ετοιμάζει το λανσάρισμα του δικού κρυπτονομίσματος, με το όνομα Libra, για χρήση μέσα στην πλατφόρμα

Μια δεύτερη πολύ σημαντική εφαρμογή είναι η δημιουργία συμβολαίων γνωστά και ως έξυπνα συμβόλαια (smart contracts), τα οποία είναι συμβόλαια που εκτελούνται χωρίς την ανάγκη αλληλεπίδρασης με ανθρώπινο δυναμικό. Επίσης, τεχνολογίες Blockchain έχουν αρχίσει να χρησιμοποιούνται και από τράπεζες για την καταγραφή συναλλαγών με σκοπό την αύξηση της ταχύτητας, της αποτελεσματικότητας των συναλλαγών, καθώς και τη μείωση του κόστους τους.

Παρόλα αυτά, οι τεχνολογίες τύπου blockchain βρίσκουν εφαρμογές και εκτός του οικονομικού τομέα. Εταιρείες λογιστικής και ασφαλιστικές τις χρησιμοποιούν για την καταγραφή και την οργάνωση των δεδομένων τους, ενώ ταυτόχρονα βρίσκουν εφαρμογές σε συστήματα IoT (Internet of Things), συστήματα ψηφοφορίας, μέσα κοινωνικής δικτύωσης και πολλά άλλα. Νέες εφαρμογές του Blockchain εμφανίζονται καθημερινά, γεγονός που το καθιστά μια ιδιαίτερα ελκυστική τεχνολογία για το μέλλον.

2.1.5 Ethereum Blockchain

Το πλέον διαδεδομένο Bitcoin παρέχει κυρίως τη δυνατότητα οικονομικών συναλλαγών μέσω του κρυπτονομίσματός του. Η καινοτομία του Ethereum συγκριτικά με το Bitcoin, είναι πως αποτελεί μια ιδιαίτερα πιο ευέλικτη και προσαρμόσιμη πλατφόρμα, πάνω στην οποία μπορούν να δημιουργήσουν και να λειτουργήσουν αποκεντρωμένες εφαρμογές (Decentralized Apps - Dapps) με ιδιαίτερη ασφάλεια.



Σχήμα 2.4: Διαφορά συναλλαγής Bitcoin με συναλλαγή Ethereum

Στην περίπτωση του Bitcoin η κατανεμημένη βάση δεδομένων γίνεται αντιληπτή ως ένας πίνακας από λογαριασμούς με bitcoin και οι συναλλαγές είναι μεταφορές των bitcoin token μεταξύ λογαριασμών. Όσο γινόταν πιο δημοφιλές, πολύς κόσμος άρχισε να χρησιμοποιεί το δίκτυο του Bitcoin για σκοπούς που δεν αφορούσαν την μεταφορά

κρυπτονομισμάτων. Το 2013 ο Vitalik Buterin πρότεινε την ιδέα ενός blockchain με τη δυνατότητα να είναι επαναπρογραμματιζόμενο ώστε να μπορεί να εκτελεί πολλές και περίπλοκες εργασίες. Το 2014 αυτή η ιδέα έγινε πράξη με την δημιουργία της πλατφόρμας Ethereum με τα έξυπνα συμβόλαια (smart contracts) που δεν δίνουν στους χρήστες έτοιμες λειτουργίες [4], αλλά του επιτρέπουν να φτιάξει τις δικές του με όση πολυπλοκότητα αυτός επιθυμεί. Στην ουσία, είναι στην ευχέρεια των επιχειρηματιών και των προγραμματιστών να αποφασίσουν την εφαρμογή και πώς θα τη σχεδιάσουν. Συνήθως η πλατφόρμα Ethereum χρησιμοποιείται για το σχεδιασμό εφαρμογών που αυτοματοποιούν την άμεση αλληλεπίδραση μεταξύ κόμβων ή τη συλλογική δράση μιας ομάδας κόμβων στο δίκτυο.

Το Ethereum στην ουσία είναι μια σουίτα από πρωτόκολλα που ορίζουν μια πλατφόρμα για αποκεντροποιημένες εφαρμογές (Dapps). Στο κέντρο του βρίσκεται το Ethereum Virtual Machine (EVM) που μπορεί να εκτελεί κώδικα αυθαίρετης αλγοριθμικής πολυπλοκότητας. Το Ethereum είναι “Turing Complete”.

Όπως όλα τα blockchain, το Ethereum λειτουργεί σε ένα P2P δίκτυο. Η Ethereum βάση δεδομένων διατηρείται και συντονίζεται από πολλούς κόμβους που είναι συνδεδεμένοι στο δίκτυο. Κάθε κόμβος του δικτύου τρέχει το EVM και εκτελεί τις ίδιες εντολές. Για αυτό κάποιες φορές το Ethereum χαρακτηρίζεται ως ένας “παγκόσμιος υπολογιστής”.

Η τεράστια παραλληλοποίηση των υπολογισμών στο Ethereum δίκτυο δεν γίνεται για μεγαλύτερη αποτελεσματικότητα, καθώς οι υπολογισμοί στο Ethereum είναι αρκετά πιο αργοί από έναν παραδοσιακό υπολογιστή. Αντίθετα ο παραλληλισμος επιτρέπει την ύπαρξη ομοφωνίας (blockchain consensus). Αυτό επιτρέπει στο Ethereum να είναι ιδιαίτερα ανεκτικό σε σφάλματα, να λειτουργεί ασταμάτητα και τα δεδομένα που είναι αποθηκευμένα να είναι αμετάβλητα.

2.1.6 Πως λειτουργεί το Ethereum

Η βασική μονάδα στο Ethereum είναι ο λογαριασμός, αντίθετα με το Bitcoin όπου βασική μονάδα είναι οι συναλλαγές [5]. Στο Ethereum το Blockchain καταγράφει την πορεία των καταστάσεων κάθε λογαριασμού, και κάθε μεταβολή κατάστασης στο Ethereum αποτελεί μεταφορά δεδομένων και πληροφορίας μεταξύ λογαριασμών. Υπάρχουν δύο τύποι λογαριασμών:

- Οι Externally Owned Accounts (EOAs), που ελέγχονται από ιδιωτικά κλειδιά
- Οι Contract Accounts (CAs), που ελέγχονται από τον κώδικα του συμβολαίου και μπορούν μόνο να ενεργοποιηθούν από EOAs

Στην ουσία οι λογαριασμοί EOAs είναι αυτοί που ελέγχονται από τους κλασικούς χρήστες με τη χρήση των ιδιωτικών κλειδιών τους, ενώ οι λογαριασμοί CA ελέγχονται από τον κώδικα τους. Οι CA μπορούν να ελεγχθούν από ανθρώπινους χρήστες μέσω EOAs μόνο αν έχουν προγραμματιστεί με αυτόν τον τρόπο. Ο περίφημος όρος "Smart Contracts" αφορά τον κώδικα των CA λογαριασμών, δηλαδή τα προγράμματα που εκτελούνται όταν γίνεται μια συναλλαγή με τον CA λογαριασμό.

Για να εκτελέσουν συναλλαγές οι λογαριασμοί EOA στο δίκτυο του Ethereum, πρέπει πρώτα να υπογράψουν τα δεδομένα της συναλλαγής με το ιδιωτικό τους κλειδί (εδώ βρίσκεται η κρυπτογραφία ελλειπτικών καμπυλών - ECDSA - Elliptic Curve Digital Signature Algorithm). Έτσι το δίκτυο μπορεί να είναι σίγουρο ότι ο αποστολέας της συναλλαγής είναι αυτός που ισχυρίζεται ότι είναι και όχι κάποιος κακόβουλος χρήστης.

Όπως και στο Bitcoin, οι χρήστες πρέπει να πληρώσουν ένα μικρό τέλος συναλλαγής στο δίκτυο. Πρωταρχικός λόγος που γίνεται αυτό είναι για να προστατευτεί το δίκτυο Ethereum από κακόβουλες επιθέσεις ή από προγραμματιστικά λάθη, όπως DDoS επιθέσεις ή άπειρους βρόχους. Ο αποστολέας μιας συναλλαγής πρέπει να πληρώσει για κάθε βήμα του προγράμματος που ενεργοποιεί, συμπεριλαμβανομένων όλων των υπολογισμών και της αποθηκευτικής μνήμης. Αυτά τα τέλη πληρώνονται σε Ether, το οποίο είναι το κρυπτονόμισμα του Ethereum.

Τα τέλη από κάθε συναλλαγή μαζεύονται από τους κόμβους που επικυρώνουν το δίκτυο. Αυτοί οι κόμβοι ονομάζονται "miners" και ο ρόλος τους είναι να αποδέχονται, να μεταδίδουν, να επικυρώνουν και να εκτελούν τις συναλλαγές. Ύστερα ομαδοποιούν τις συναλλαγές - οι οποίες περιλαμβάνουν πολλές μεταβολές των καταστάσεων των λογαριασμών του Ethereum - σε αυτό που αποκαλούμε "blocks" και οι miners μετά ανταγωνίζονται ο ένας τον άλλον ώστε το δικό τους block να είναι αυτό που θα προστεθεί ως επόμενο στην αλυσίδα του blockchain. Όποτε το καταφέρνουν ανταμείβονται με Ether.

2.2 Proof of Location

Η ανάπτυξη της τεχνολογίας των τελευταίων χρόνων έχει δώσει τη δυνατότητα για ακριβή εντοπισμό της τοποθεσίας μιας οντότητας. Αυτή η δυνατότητα πλέον είναι διαθέσιμη σε όλους. Παράλληλα, τα "έξυπνα" κινητά τηλέφωνα διαθέτουν τη δυνατότητα γεωεντοπισμού μέσω GPS, WiFi και του δικτύου κινητής τηλεφωνίας, να εκτελούν εφαρμογές που να μπορούν να εκμεταλλευτούν την τοποθεσία της συσκευής και να μεταδίδουν την τοπεθεσία τους στις εφαρμογές, όπου και αν βρίσκονται.

Οι παραπάνω συνθήκες οδήγησαν τα τελευταία χρόνια στην εμφάνιση μιας πολύ μεγάλης αύξησης στις υπηρεσίες τοποθεσίας. Αυτό προσφέρει νέες δυνατότητες στις

εφαρμογές. Εφαρμογές όπως το Foursquare που προτείνει εστιατόρια με βάση την τοποθεσία του χρήστη και εφαρμογές τραπεζών που χρησιμοποιούν την γεωτοποθεσία του κινητού τηλεφώνου του χρήστη για να επιβεβαιώσουν οτι η πιστωτική κάρτα του ή ο αριθμός της δεν έχει κλαπεί, εμφανίζονται συνεχώς. Κάπως έτσι η ακρίβεια, αλλά και η πιστότητα της τοποθεσίας ενός ατόμου αποκτά πολύ σημαντικό ρόλο στην αλληλεπίδραση του ανθρώπου με τις συσκευές.

Αυτή η ακρίβεια και η πιστότητα βασίζονται στην ειλικρίνεια των χρηστών ώστε να είναι έμπιστη η τοποθεσία τους. Αν ο χρήστης επωφελείται από την εμφάνιση διαφορετικής τοποθεσίας, έχει μεγάλο κίνητρο είτε να δηλώσει ψεύτικη τοποθεσία, είτε να ζητήσει από άλλες συσκευές να αποδείξουν ότι η τοποθεσία του πρώτου είναι διαφορετική στην αντίστοιχη εφαρμογή. Αυτό θα μπορούσε να συμβεί από κάποιον ληστή μιας πιστωτικής κάρτας ο οποίος θα υποστήριζε ψευδώς ότι κατά τη συναλλαγή του βρίσκεται στο σπίτι του θύματος, όταν στην πραγματικότητα βρίσκεται αλλού.

Η ανάγκη για ακριβή εντοπισμό τοποθεσίας ενός χρήστη, αλλά και η επιβεβαίωση αυτής έχει οδηγήσει στην ανάπτυξη μεγάλου αριθμού πρωτοκόλλων απόδειξης τοποθεσίας (Proof-of-Location protocols) με σκοπό την επιβεβαίωση της τοποθεσίας του χρήστη μιας εφαρμογής, ώστε να μην μπορεί ή να του είναι εξαιρετικά δύσκολο να παραποιήσει την τοποθεσία του.

2.2.1 Εφαρμογές Proof of Location

Τα πρωτόκολλα απόδειξης τοποθεσίας μπορούν να βρουν εφαρμογή σε μεγάλη πληθώρα καταστάσεων, εφόσον απαιτείται γνώση της τοποθεσίας του χρήστη. Μερικές φαίνονται παρακάτω.

Πολλές εφαρμογές παρέχουν υπηρεσίες με βάση την τοποθεσία του χρήστη. Για παράδειγμα το Netflix έχει διαφορετική ταινιοθήκη στις ΗΠΑ και διαφορετική στην Ελλάδα. Αυτό οδηγεί πολλούς χρήστες στο να χρησιμοποιούν εικονικά δίκτυα (VPNs) ή smart proxies για να εμφανίζουν διαφορετική διεύθυνση IP, με αποτέλεσμα να έχουν πρόσβαση σε ταινίες που δεν θα έπρεπε να έχουν, προκαλώντας νομικά προβλήματα για το Netflix που ενδεχομένως να μην έχει τα δικαιώματα αυτών των ταινιών στις χώρες αυτές. Αν ο πάροχος ταινιών χρησιμοποιούσε κάποιο πρωτόκολλο απόδειξης τοποθεσίας, κάτι τέτοιο θα μπορούσε να αποφευχθεί.

Όλο και περισσότερο διάφορα μέσα κοινωνικής δικτύωσης προσθέτουν υπηρεσίες που απαιτούν τη γνώση της τοποθεσίας του χρήστη της συσκευής. Συχνά οι χρήστες μοιράζονται την τοποθεσία τους (πχ. Check In στο Facebook) ή κατά την επικοινωνία τους με άλλους χρήστες μπορούν να διαμοιραστούν την τοποθεσία τους, ενδεχομένως για να

συναντηθούν (πχ. Share Location στο Messenger). Η χρήση κάποιου πρωτοκόλλου Απόδειξης Τοποθεσίας θα οδηγούσε στην ειλικρίνεια των χρηστών.

Επιπρόσθετα, πολλές επιχειρήσεις επιθυμούν την επιβράβευση των τακτικών πελατών ώστε να δώσουν κίνητρα σε αυτούς για συγκεκριμένες ενέργειες. Πιθανά κάποια επιχείρηση να προτιμά την πώληση των προϊόντων της από κάποιο συγκεκριμένο φυσικό μαγαζί και όχι από το ψηφιακό eshop της, με αποτέλεσμα να επιβραβεύει με εκπτώσεις και loyalty πόντους τους πελάτες που επισκέπτονται το φυσικό μαγαζί. Αυτό όμως δίνει σημαντικό κίνητρο στους πελάτες να προσπαθήσουν να δηλώσουν ψευδή τοποθεσία, ώστε να εκμεταλλευτούν τις επιβραβεύσεις. Κάποιο πρωτόκολλο Απόδειξης Τοποθεσία θα έλυνε το πρόβλημα αυτό.

Τέλος, συχνά είναι απαραίτητη η παρακολούθηση της τοποθεσίας ατόμων ή αντικειμένων. Κάτι τέτοιο θα μπορούσε να είναι απαραίτητο από την αστυνομία στην πρώτη περίπτωσή ή από οποιονδήποτε χρήστη κινητής συσκευής που επιθυμεί να βρει το χαμένο του κινητό. Έτσι και εδώ τα πρωτόκολλα Απόδειξης Τοποθεσίας θα βοηθούσαν στην εγκυρότητα της τοποθεσίας των φυσικών προσώπων ή των αντικειμένων.

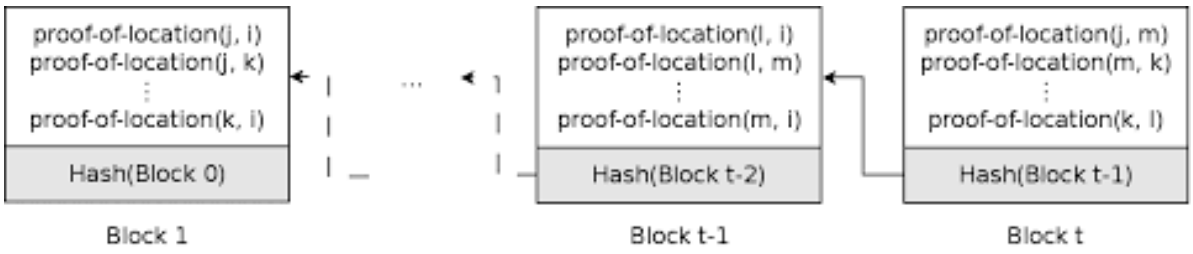
2.2.2 Υπάρχουσες Προτάσεις PoL

Υπάρχει πληθώρα διαφορετικών πρωτοκόλλων Proof-of-Location τα οποία χρησιμοποιούνται εκτενώς. Άλλα βασίζονται στην χρήση εξειδικευμένου hardware, όπως το FOAM το οποίο χρησιμοποιεί beacons που ονομάζονται Zone Anchors για την επιβεβαίωση της τοποθεσίας του χρήστη που ζήτησε την Απόδειξη τοποθεσίας. Άλλα πρωτόκολλα χρησιμοποιούν πρωτότυπο software και χρησιμοποιούν ευρέως διαδεδομένες τεχνολογίες για hardware, όπως εύρεση τοποθεσίας μέσω GPS, WiFi κ.α και επικοινωνία μέσω WiFi, Bluetooth κ.α. Διάφορες από τις προτάσεις βρίσκονται στην βιβλιογραφία από το [7] μέχρι το [23].

Στο σημείο αυτό θα αναφερθούμε στα πρωτόκολλα που σχετίζονται περισσότερο με την εφαρμογή που αναπτύχθηκε στο πλαίσιο αυτής της εργασίας.

Blockchain PoL

Στο Blockchain PoL [24] η αποκεντροποιημένη φύση των peer-to-peer δικτύων εξασφαλίζει μεγάλα επίπεδα ιδιωτικότητας, αφού δεν απαιτείται κάποια κεντρική αρχή για να αποδειχθεί η τοποθεσία κάποιου χρήστη. Αντίθετα με τη χρήση κάποιας τεχνολογίας σχετικά μικρής εμβέλειας και με την κρυπτογραφημένη επικοινωνία μεταξύ κοντινών κόμβων μπορεί να επιτευχθεί η Απόδειξη Τοποθεσίας κάποιου κόμβου.



Σχήμα 2.5: Δομή Blockchain ως βάση δεδομένων

Οι συγγραφείς υποστηρίζουν ότι η δήλωση ψευδούς τοποθεσίας είναι αδύνατη αφού χρησιμοποιούνται τεχνολογίες κοντινής εμβέλειας και αφού οι αποδείξεις τοποθεσίας αποθηκεύονται στο Blockchain, το οποίο καθιστά την αναμετάδοση κάποιας απόδειξης πολλές φορές αδύνατη. Η χρήση τεχνολογίας κοντινής εμβέλειας, καθώς και η χρήση Blockchain ως βάσης δεδομένων για τις παλιές αποδείξεις τοποθεσίας, είναι υπαρκτή και στην υλοποίηση αυτής της εργασίας, όπου χρησιμοποιούμε Bluetooth για την επικοινωνία μεταξύ κοντινών κόμβων, καθώς και το Ethereum Network για την αποθήκευση παλαιών αποδείξεων τοποθεσίας.

Alice in Wonderland

Το πρωτόκολλο Alice in Wonderland [25] δεν έχει ως σκοπό να προσδιορίσει την ακριβή θέση του χρήστη, αλλα να δείξει ότι ο χρήστης βρίσκεται σε μια γενικότερη περιοχή, δίνοντας όχι τις συντεταγμένες αλλά την γειτονιά στην οποία βρίσκεται ο χρήστης.

Το στοιχείο του πρωτοκόλλου Alice in Wonderland αφορά τη χρήση χρόνου απόκρισης για την αντιμετώπιση MITM attacks, ώστε να εντοπίζει την ύπαρξη ενδιάμεσου κόμβου σε περίπτωση που ο χρόνος αυτός είναι μεγάλος.

Witness Based Location Proof

Ένα άλλο, όχι τόσο διαδεδομένο πρωτόκολλο Απόδειξης Τοποθεσίας είναι το Witness Based Location Proof [26]. Και αυτο χρησιμοποιεί Bluetooth για την επικοινωνία των κόμβων.

Ο λόγος που αναφέρεται είναι γιατί χρησιμοποιεί έναν Decay Mechanism για την αντιμετώπιση Sybil Attacks. Με τον όρο Sybil Attack εννοούμε την επίθεση που αφορά την συγκέντρωση πολλών κακόβουλων κόμβων ώστε να επιβληθούν σε ένα δίκτυο. Στο πλαίσιο του Proof of Location θα μπορούσε κάποιος χρήστης που θέλει να δηλώσει ψευδή τοποθεσία να συνεργαστεί με αρκετούς επίσης κακόβουλους χρήστες, ώστε να δηλώσουν και αυτοί την ίδια τοποθεσία και να περάσει η απόδειξη τους ως ψευδής.

Για αυτό οι συγγραφείς προτείνουν τη χρήση ενός Decaying Mechanism, με τον οποίο όταν ζευγάρια χρηστών έχουν συμμετάσχει σε παρελθοντικές Αποδείξεις Ταυτότητας πολλές φορές. να θεωρούνται ύποπτες και να απορρίπτονται. Αυτό δουλεύει υπό την υπόθεση ότι ο χρήστης δεν βρίσκεται τυχαία πολλές φορές στο ίδιο περιβάλλον με τους ίδιους κόμβους κοντά του που να του επιβεβαιώνουν την τοποθεσία. Αυτή η υπόθεση είναι αρκετά έγκυρη, καθώς η πιθανότητα να βρίσκεται ένας χρήστης πολλές φορές κοντά στους ίδιους χρήστες είναι εξαιρετικά μικρή.

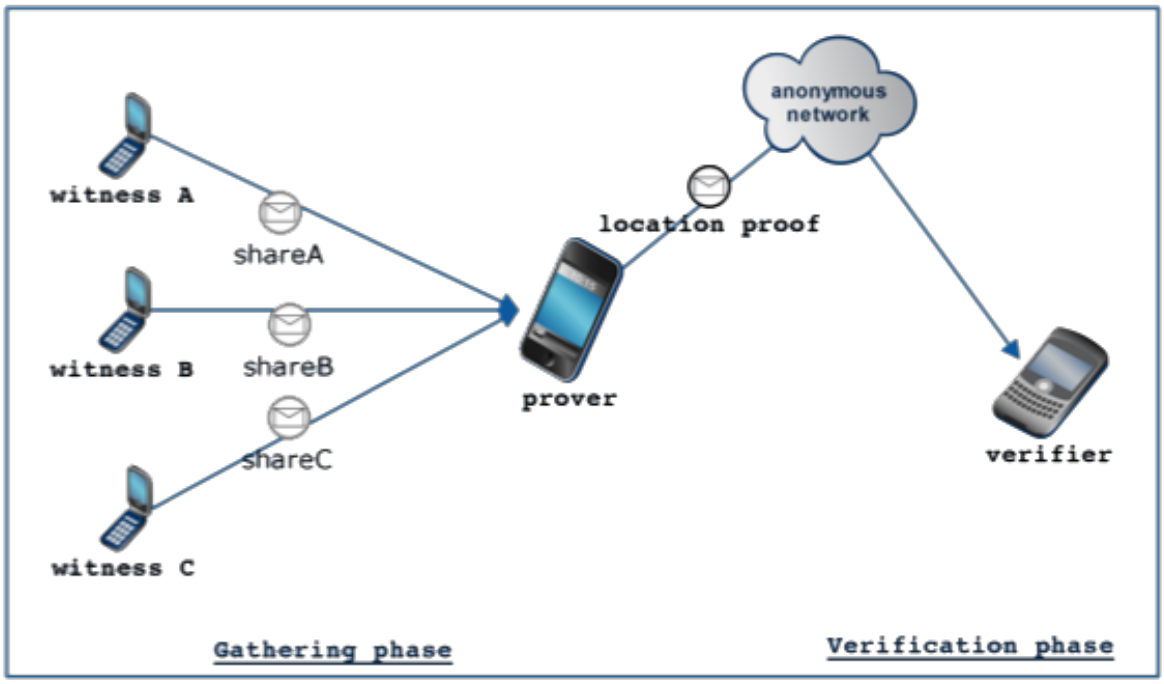
3. Πρωτόκολλο Proof of Location

Στο σημείο αυτό θα περιγράψουμε το πρωτόκολλο μας αναλυτικά. Όπως και πριν θα χρησιμοποιήσουμε κάποιους όρους για να περιγράψουμε τους χρήστες κ.α

Prover: Ο Prover είναι ο κάτοχος κινητής συσκευής που επιθυμεί να αποδείξει την τοποθεσία του.

Witness: Ο Witness είναι ο κάτοχος κινητής συσκευής που προσφέρεται να προμηθεύσει τον prover με Απόδειξη Τοποθεσίας. Μπορεί είτε να αποδείξει την τοποθεσία του prover, είτε να την απορρίψει.

Verifier: Ο Verifier είναι η οντότητα που καλείται να επαληθεύσει την απόδειξη τοποθεσίας που προμηθεύεται από τους prover. Στην περίπτωση μας Verifier είναι το smart contract που έχει γίνει deploy στο Ethereum Blockchain.



Σχήμα 3.1: Δίκτυο Prover-Witness-Verifier και διαδικασία PoL

3.1 Τεχνολογίες

Ένα πρωτόκολλο που απαιτεί πολλούς witnesses χρησιμεύει περισσότερο για εφαρμογή σε κινητά τηλέφωνα, τα οποία συχνά βρίσκονται σε κίνηση και συνήθως κοντά στο χρήστη. Για την επικοινωνία των συσκευών χρησιμοποιείται Bluetooth, ενώ η τοποθεσία αποκτάται από το Network Provider. Όταν δεν είναι διαθέσιμος χρησιμοποιείται το GPS.

3.1.1 Smartphones

Το πρωτόκολλο αναπτύχθηκε για εφαρμογή σε κινητά τηλέφωνα τύπου smartphone, λόγω της εξαιρετικά εκτενούς χρήσης τους και της μεγαλύτερης ανάγκης για απόδειξη τοποθεσίας που αυτά παρουσιάζουν. Συγκεκριμένα έγινε σε κινητά με λειτουργικό Android το οποίο αποτελεί το λειτουργικό που συναντάται συχνότερα σε κινητές συσκευές. Παρόλα αυτά η ίδια εφαρμογή θα μπορούσε να αναπτυχθεί και σε άλλα λειτουργικά όπως πχ iOS.

3.1.2 Bluetooth

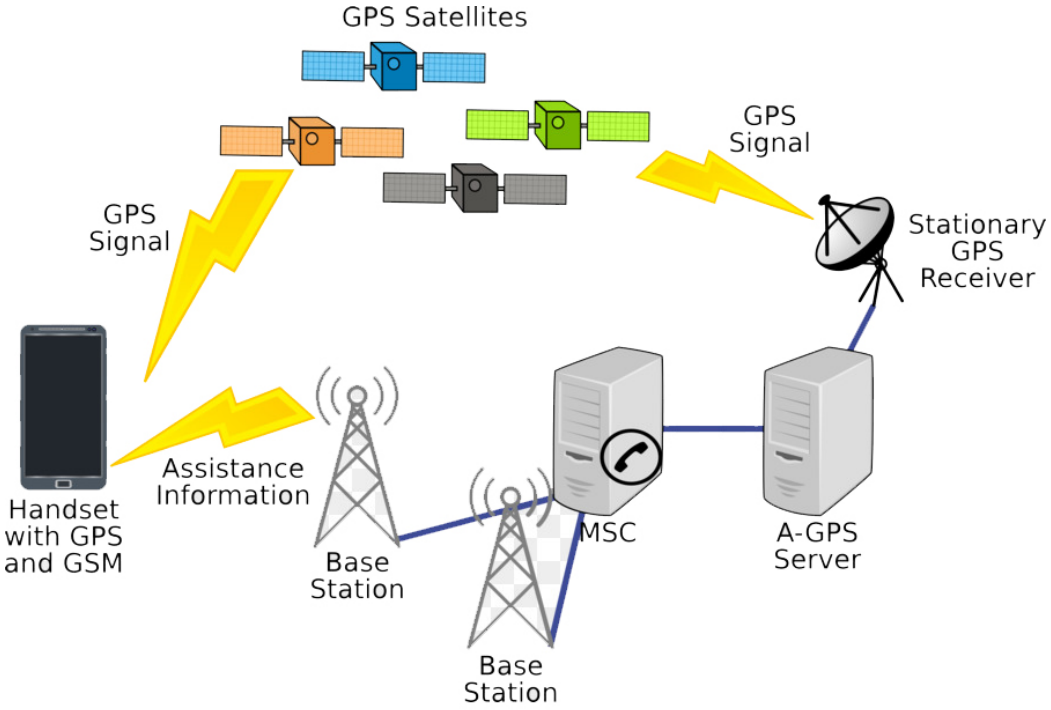
Η χρήση τεχνολογίας επικοινωνίας μικρής εμβέλειας μεταξύ prover και witness είναι εξαιρετικά ελκυστική, καθώς από την φύση της επιβεβαιώνει ότι οι δύο χρήστες βρίσκονται σίγουρα κοντά. Στην ουσία ο prover από μόνος του δεν θα μπορούσε να κοροϊδέψει το σύστημα καθώς οι witnesses (που δέχτηκαν την τοποθεσία του prover με bluetooth και συνεπώς βρίσκονται σίγουρα κοντά του) θα τον πρόδιδαν στην Απόδειξη Τοποθεσίας που θα του έδιναν. Στην περίπτωση που επικοινωνούσαν οι δύο χρήστες με http request, τότε ο witness δεν θα είχε καμία πληροφορία για τον prover και θα ήταν πιο δύσκολη η παραγωγή της απόδειξης τοποθεσίας. Για αυτόν το λόγο επιλέγεται η χρήση Bluetooth η οποία είναι μια ευρέως διαδεδομένη τεχνολογία επικοινωνίας μικρής εμβέλειας, αλλά με ικανοποιητική εμβέλεια ώστε η εφαρμογή να είναι εύκολα χρησιμοποιήσιμη. Η απόσταση που απαιτείται για να δουλέψει το Bluetooth εξαρτάται από τις συσκευές που έχουμε διαθέσιμες. Οι περισσότερες κινητές συσκευές έχουν εύρος μεγαλύτερο από 10 μέτρα, ενώ πιο μοντέρνα κινητά έχουν αρκετά παραπάνω, με μερικά να φτάνουν και τα 150 μέτρα με το Bluetooth 5.0. Υπάρχουν μάλιστα αναφορές για κινητές συσκευές που έχουν καταφέρει σύνδεση Bluetooth ακόμα και στα 350 μέτρα.

Αξίζει να αναφερθεί ότι η επικοινωνία με Bluetooth δεν είναι πάντοτε η πιο ασφαλής. Για αυτό απαιτείται πάντα από όλα τα λειτουργικά σε κινητά τηλέφωνα, να απαιτείται επιβεβαίωση από το χρήστη κατά την έναρξη οποιασδήποτε επικοινωνίας. Πρέπει δηλαδή ο χρήστης να είναι σίγουρος ότι η συσκευή με την οποία συνδέεται είναι έμπιστη και μετά να κάνει pair μαζί της. Αυτή η ανάγκη για pairing με έμπιστες συσκευές μόνο είναι η μεγαλύτερη άμυνα του Bluetooth απέναντι σε Man-in-the-Middle επιθέσεις. Η τεχνολογία Bluetooth είναι συνεχώς αναπτυσσόμενη.

3.1.3 GPS - AGPS

Η συντομογραφία GPS προέρχεται από το Global Positioning System [27]. Χρησιμοποιεί ραδιοκύματα μεταξύ των δορυφόρων και του δέκτη μέσα στο κινητό τηλέφωνο για να προμηθεύσει με την τοποθεσία και τη χρονική στιγμή οποιοδήποτε

λογισμικό χρειάζεται αυτές τις πληροφορίες. Το κινητό τηλέφωνο στην ουσία μπορεί να δέχεται τις πληροφορίες αυτές από 4 ή περισσότερους από τους 28 δορυφόρους που βρίσκονται σε τροχιά και χρησιμοποιούνται σχεδόν αποκλειστικά για αυτόν το σκοπό. Το GPS είναι αρκετά ακριβές, αλλα χρησιμοποιεί ενέργεια και συχνά είναι αρκετά αργό, ενώ δεν λειτουργεί σε κλειστούς χώρους.



Σχήμα 3.2: Παράδειγμα γεωγραφικού προσδιορισμού χρήστη με Assisted GPS

Για τους παραπάνω λόγους τα κινητά χρησιμοποιούν AGPS (Assisted Global Positioning System). Αυτό που προσθέτουν στην παραπάνω διαδικασία είναι ότι χρησιμοποιούν τα δεδομένα του κινητού για να προσδιορίσουν τη γεωγραφική θέση. Αυτό γίνεται εφικτό επειδή το κινητό στέλνει μηνύματα ping στους τηλεπικοινωνιακούς πύργους. Επομένως, τρεις διαφορετικοί πύργοι μπορούν να εντοπίσουν τη συσκευή. Η ακρίβεια του AGPS εξαρτάται από το πόσο καλή είναι η σύνδεση στο διαδίκτυο. Παρόλα αυτά συνήθως είναι ακριβές.

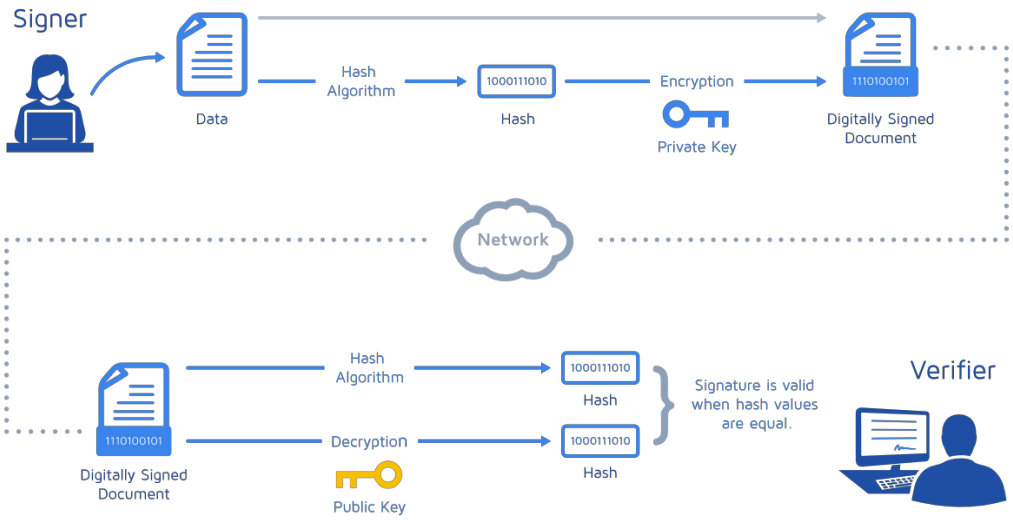
Το AGPS του κινητού επικοινωνεί τόσο με τους τηλεπικοινωνιακούς πύργους, όσο και με τους δορυφόρους όπως είπαμε. Τα Android επιτρέπουν και αποκλειστική χρήση των πύργων, όταν είναι εφικτό αυτό. Αυτό γίνεται με το NETWORK_PROVIDER. Όταν δεν είναι διαθέσιμος αυτός, τότε χρησιμοποιούμε το GPS_PROVIDER.

3.2 Κρυπτογραφία

Κατά την επικοινωνία δύο ή περισσότερων συσκευών είναι απαραίτητο τα κανάλια επικοινωνίας να είναι ασφαλή. Αυτό φυσικά δεν είναι πάντα εφικτό, με αποτέλεσμα πολλές φορές μια συσκευή να επικοινωνεί με κακόβουλο χρήστη. Μπορεί επίσης αυτά που στέλνει η συσκευή να μην ακούγονται μόνο από τον επιθυμητό προορισμό, αλλά να "ακούνε" και τρίτοι χρήστες. Το πρώτο πρόβλημα αντιμετωπίζεται σε μεγάλο βαθμό με τις ψηφιακές υπογραφές, ενώ το δεύτερο με κάποια μορφή κρυπτογράφησης.

3.2.1 Ψηφιακές Υπογραφές

Πέρα από την ανάγκη απόδειξης της φυσικής τοποθεσίας του χρήστη, είναι σημαντικό αυτός να μπορεί να επιβεβαιώσει και την ταυτότητα του. Επίσης είναι απαραίτητο το μήνυμα να μεταφέρεται αμετάβλητο από πιθανούς κακόβουλους χρήστες που επιθυμούν είτε να πουν ψέματα για την ταυτότητα τους και να κοροϊδέψουν κάποιον άλλο χρήστη, είτε απλά να μεταβάλουν το περιεχόμενο του μηνύματος, με κάποια μορφή Man-in-the-Middle attack. Αυτό αντιμετωπίζεται εύκολα με τη χρήση ψηφιακών υπογραφών.



Σχήμα 3.3: Διαδικασία δημιουργίας Ψηφιακής Υπογραφής

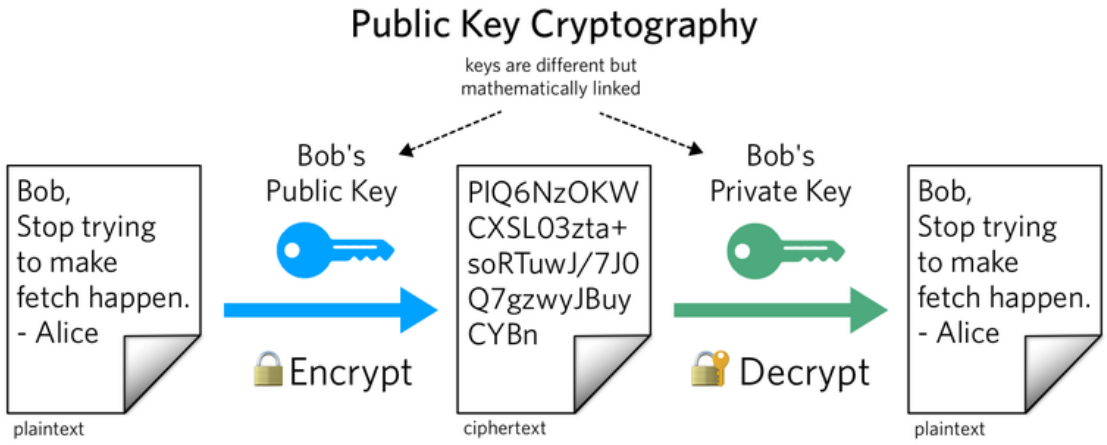
Για τη δημιουργία μιας ψηφιακής υπογραφής απαιτείται ένα ιδιωτικό και ένα δημόσιο κλειδί. Κάθε χρήστης κατέχει ένα ζευγάρι. Το ιδιωτικό κλειδί χρησιμοποιείται για την κρυπτογράφηση ενός μηνύματος, ενώ το δημόσιο κλειδί χρησιμοποιείται για την αποκρυπτογράφηση του κρυπτογραφημένου μηνύματος. Όπως φαίνεται και από τα ονόματα τους, το ιδιωτικό είναι κρυφό και στην κατοχή του χρήστη που αντιπροσωπεύει, ενώ το δημόσιο είναι διαθέσιμο σε όλους. Κάθε ζευγάρι ιδιωτικού και δημόσιου κλειδιού είναι μοναδικό.

Το ιδιωτικό κλειδί βρίσκεται στα χέρια ενός χρήστη και πρέπει να προστατευτεί καλά. Θα χρησιμοποιηθεί για την κρυπτογράφηση του μηνύματος και επειδή είναι μοναδικό ανα χρήστη, στην ουσία επαληθεύει την ταυτότητα του. Παρόλα αυτά χρειάζεται και ένας τρίτος έμπιστος χρήστης, ο οποίος επίσης να γνωρίζει ότι το ιδιωτικό κλειδί αντιστοιχεί σε κάποιον χρήστη και να επαληθεύει την ταυτότητα του.

Στο πλαίσιο της εργασίας αυτής η ψηφιακές υπογραφές που χρησιμοποιούνται είναι αυτές που παρέχονται από το Ethereum. Ο αλγόριθμος συγκεκριμένα είναι ο ECDSA (Elliptic Curve Digital Signature Algorithm) που αναπτύξαμε σε προηγούμενη ενότητα. Κατά τη δημιουργία λογαριασμού στο Blockchain Ethereum ο νέος κάτοχος του λογαριασμού έχει στα χέρια του την διεύθυνση του λογαριασμού του, η οποία αποτελεί το δημόσιο κλειδί του και ένα ιδιωτικό κλειδί. Με το ζευγάρι αυτό μπορεί να κρυπτογραφήσει και να αποκρυπτογραφήσει το μήνυμα του εκτός του Ethereum και το Ethereum μπορεί να επαληθεύσει οποιαδήποτε στιγμή με τη χρήση της διεύθυνσης και του κρυπτογραφημένου μηνύματος ότι όντως ο αποστολέας του μηνύματος είναι αυτός που υποστηρίζει.

3.2.2 Κρυπτογραφία RSA

Στο πλαίσιο της εργασίας, κατά την ανταλλαγή μηνυμάτων θα χρησιμοποιηθεί ασύμμετρη κρυπτογραφία δημοσίου κλειδιού RSA των μηνυμάτων, για να αποφευχθούν παρεμβολές από τρίτους όταν αυτό φτάσει στον προορισμό του. Όντας ο μοναδικός χρήστης με το ιδιωτικό κλειδί, είναι και ο μόνος που μπορεί να αποκρυπτογραφήσει το μήνυμα και επομένως ο προορισμός μπορεί να είναι σίγουρος ότι το μήνυμα δεν έχει αλλαχθεί στην πορεία. Παρόλα αυτά δεν μπορεί να είναι σίγουρος για την ταυτότητα του αποστολέα, για αυτό απαιτούνται και οι ψηφιακές υπογραφές όπως είπαμε παραπάνω.



Σχήμα 3.4: Παράδειγμα κρυπτογράφησης δημοσίου κλειδιού

Η κρυπτογραφία RSA ήταν ιδέα των Whitfield Diffie και Martin Hellman, που υλοποιήθηκε από τους Ron Rivest, Adi Shamir και Leonard Adleman, από τα αρχικά των οποίων προκύπτει η ονομασία RSA.

Ένα κρυπτοσύστημα δημοσίου κλειδιού αποτελείται από αρκετά στοιχεία. Υπάρχει ένα σύνολο από πιθανά κείμενα που το λέμε M. Υπάρχει επίσης ένα σύνολο από κλειδιά, το K.Για κάθε $k \in K$, υπάρχει μια συνάρτηση κρυπτογράφησης $encrypt_k$ και μία αποκρυπτογράφησης $decrypt_k$. Αυτά τα στοιχεία πρέπει να ικανοποιούν τις παρακάτω συνθήκες:

1. $encrypt_k(decrypt_k(M)) = M$ και $decrypt_k(encrypt_k(M)) = M$ για κάθε $m \in M$ και every $k \in K$.

2. Για κάθε M και κάθε k, οι τιμές των $encrypt_k(M)$ και $decrypt_k(M)$ είναι εύκολο να υπολογιστούν.

3. Για σχεδόν κάθε $k \in K$, αν κάποιος γνωρίζει μόνο τη συνάρτηση $encrypt_k$, δεν είναι υπολογιστικά εφικτό να βρει αλγόριθμό για να υπολογίσει το $decrypt_k$.

4. Για δοθέν $k \in K$, είναι εύκολο να βρει κανείς τις συναρτήσεις $encrypt_k$ και $decrypt_k$.

Η κρυπτογραφία RSA βασίζεται στην παραπάνω λογική. Ο αλγόριθμος δεν θα αναλυθεί στην παρούσα εργασία. Η ισχύς και η ασφάλεια του αλγορίθμου RSA βασίζεται στη δυσκολία παραγοντοποίησης μεγάλων αριθμών. Ο αλγόριθμος είναι γενικά ο πιο ευρέως διαδεδομένος αλγόριθμος κρυπτογράφησης.

3.2.3 Μηχανισμός αποφυγής συνεργασίας

Για να αποφευχθεί η χρήση κακόβουλων γειτόνων για την δήλωση εσφαλμένης τοποθεσίας, χρησιμοποιείται ένας μηχανισμός που απαγορεύει στους witnesses να παράξουν Απόδειξη Ταυτότητας στον ίδιο prover πολλές φορές. Κάθε ζευγάρι prover-witness έχει ένα credit score, το οποίο αν περάσει το 20, τότε ο witness δεν μπορεί να παράξει Απόδειξη Τοποθεσίας στον Prover αυτόν. Έτσι αποφεύγεται κατάχρηση των witnesses από τον ίδιο prover. Ο μηχανισμός περιγράφεται παρακάτω.

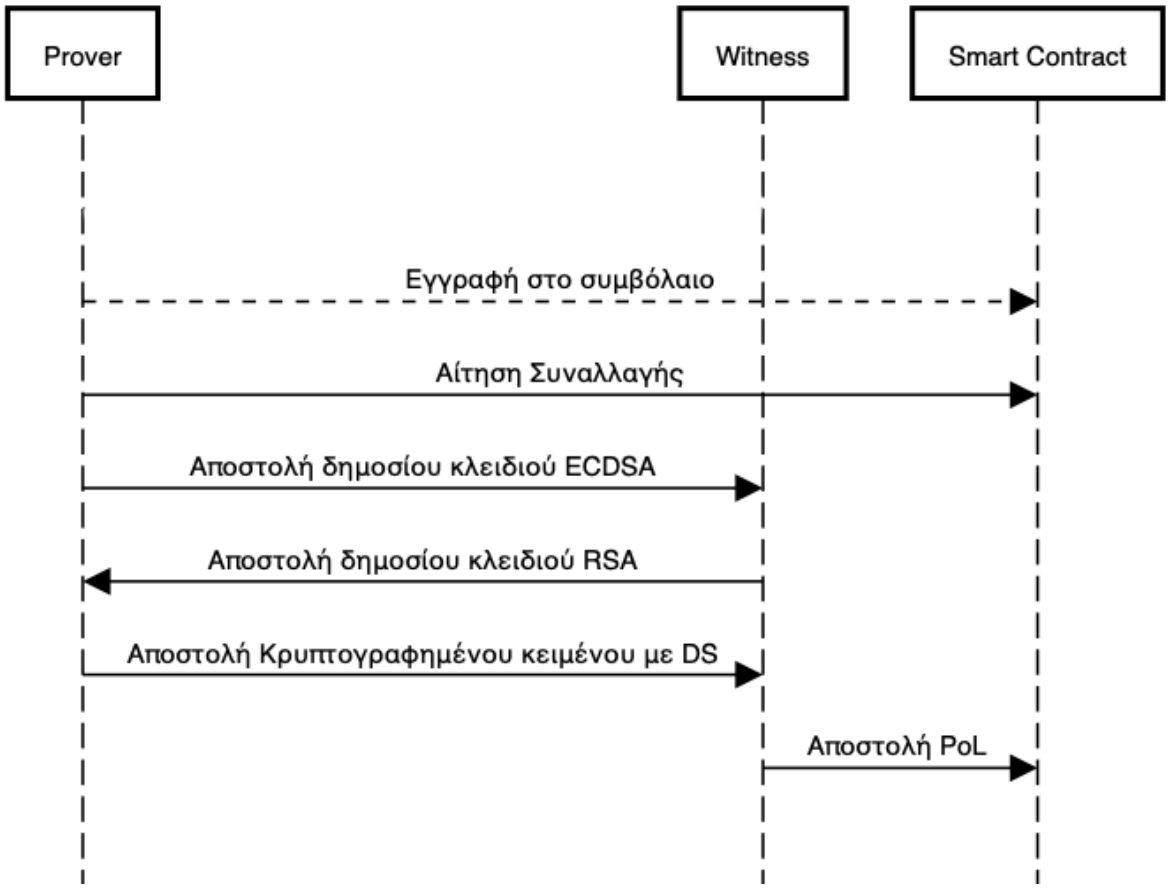
$$Vxy = V + 20\frac{Nxy}{U},$$

όπου V είναι το Vxy πριν τη συναλλαγή, Nxy είναι ο αριθμός των προηγούμενων συναλλαγών με prover X και witness Y και U είναι ο συνολικός αριθμός των Αποδείξεων Ταυτότητας που απαιτούνται για τη συναλλαγή.

Αυτό σημαίνει ότι στην περίπτωση που ένας prover χρησιμοποιήσει μόνο έναν witness για να αποδείξει την τοποθεσία του, δηλαδή U = 1, τότε από την πρώτη κιόλας φορά θα έχει credit score ίσο με 20. Αυτό σημαίνει, ότι δεν θα μπορεί να ξαναχρησιμοποιηθεί ίδιος witness, γεγονός που απαγορεύει συνεργασίες κακόβουλων prover και witnesses πολλές φορές. Το ίδιο ισχύει και αν έχουμε U > 1, καθώς το credit score του witness με τον prover θα αυξάνεται και τελικά θα απαγορευτεί η συνεργασία τους.

3.3 Αναλυτική Περιγραφή Πρωτοκόλλου

Το προτεινόμενο πρωτόκολλο περιγράφεται με το παρακάτω Message Sequence Chart:



Σχήμα 3.5: Message Sequence Chart επικοινωνίας Prover, Witness και Smart Contract

Το πρωτόκολλο περιγράφεται παρακάτω. Οι μέθοδοι που αναφέρονται είναι μέθοδοι Java της εφαρμογής Android που υλοποιήθηκε για την υλοποίηση του πρωτοκόλλου. Θα περιγραφούν και εξηγηθούν πιο αναλυτικά στο 5.2.1.

Βήμα 1: Ο Prover πρέπει να έχει κάνει εγγραφή στο έξυπνο συμβόλαιο. Αυτή η διαδικασία έχει κόστος περίπου 0.02 ether (τη στιγμή της συγγραφής της εργασίας αντιστοιχεί σε περίπου 5$ στο Ethereum mainnet). Το ίδιο ισχύει και για τους witnesses.

Ανάλυση: Η διαδικασία της εγγραφής του χρήστη γίνεται με την κλήση της μεθόδου **addUser** στο smart contract στο Ethereum. Αυτή η μέθοδος είναι η μόνη του συμβολαίου που δεν ελέγχει αν ο χρήστης που την καλεί έχει κάνει εγγραφή. Απαιτείται αποστολή 0.02 ether κατά την κλήση της. Αυτή η κλήση δεν γίνεται από την εφαρμογή που έχει υλοποιηθεί στο πλαίσιο αυτής της διπλωματικής εργασίας, αλλά γίνεται "χειροκίνητα" από τον κάτοχο λογαριασμού Ethereum που επιθυμεί να χρησιμοποιήσει την εφαρμογή. Αυτό μπορεί να γίνει από το Remix IDE, κάνοντας deploy το συμβόλαιο και καλώντας τη μέθοδο με το λογαριασμό που θέλει να εγγραφεί.

Αυτή η διαδικασία γίνεται μία μόνο φορά από τον χρήστη και δεν απαιτείται ξανά μετά. Αν δεν έχει ολοκληρωθεί μια επιτυχημένη κλήση της addUser τότε ο χρήστης δεν μπορεί να κάνει συναλλαγή στο smart contract.

Βήμα 2: Ο Prover κάνει "αίτηση" για συναλλαγή στο Ethereum Blockchain μεταφέροντας τα χρήματα του στο έξυπνο συμβόλαιο, μαζί με τη διεύθυνση του προορισμού της συναλλαγής και τον αριθμό των witnesses. Τώρα το έξυπνο συμβόλαιο περιμένει τις Αποδείξεις Τοποθεσίας από τους witnesses για να ολοκληρώσει τη συναλλαγή.

Ανάλυση: Η διαδικασία της αίτησης γίνεται μέσω της κλήσης της μεθόδου **transferToContract**, από την εφαρμογή του smartphone. Πριν την κλήση της μεθόδου, ο prover πρέπει να έχει επιλέξει τον προορισμό των ether της συναλλαγής, καθώς και το ποσό των ether που επιθυμεί να στείλει. Επίσης, πρέπει να επιλέξει τον αριθμό των witnesses από τη λίστα με τους έμπιστους witnesses που βρίσκονται κοντά του. Ο λόγος που απαιτείται η επιλογή των witnesses από τον prover, είναι επειδή δεν είναι ασφαλές να δίνεται πρόσβαση Bluetooth σε άλλες συσκευές που δεν είναι έμπιστες, καθώς τότε προκύπτουν θέματα ασφαλείας.

Επίσης, αν γινόταν αυτόματα και τυχαία η επιλογή των witnesses από την εφαρμογή, τότε στην περίπτωση που η συσκευή κάποιου witness δεν είχε εγκατεστημένη την εφαρμογη ή στην περίπτωση που δεν ήταν paired, τότε η προσπάθεια σύνδεσης Bluetooth θα έπαιρνε παραπάνω ώρα, καθυστερώντας τη διαδικασία

Με την κλήση αυτή, γίνεται μεταφορά των ether στο συμβόλαιο, ώστε στο μέλλον να ολοκληρωθεί η συναλλαγή, αφού το συμβόλαιο δεχτεί Αποδείξεις Ταυτότητας από τους Witnesses που επιλέχθηκαν. Η διαδικασία αυτή είναι η πιο χρονοβόρα από όλη τη διαδικασία, λόγω της ανάγκης αλληλεπίδρασης με το Ethereum για τη μεταφορά των ether και των πληροφοριών.

Βήμα 3: Ο Prover στέλνει στους Witnesses του μέσω Bluetooth το δημόσιο κλειδί του.

Ανάλυση: Η διαδικασία των **Βημάτων 3-5** γίνεται αυτόματα με την κλήση της sendRequestPublicKey από τον prover. Αυτό γίνεται επίσης από την εφαρμογή και είναι η δεύτερη και τελευταία ενέργεια που απαιτείται από τον prover για την ολοκλήρωση της συναλλαγής. Αυτόματα θα ολοκληρωθούν και τα **Βήματα 6-8** από τη μεριά του witness, χωρίς καμία ανάγκη αλληλεπίδρασης του χρήστη-witness. **Η διαδικασία των Βημάτων 3-7 θα γίνει για κάθε witness σειριακά**.

Κατά την **sendRequestPublicKey**, ο prover ξεκινάει σύνδεση Bluetooth με τη συσκευή του witness. Με το που ολοκληρωθεί η σύνδεση, ο prover στέλνει το δημόσιο κλειδί του που θα χρησιμοποιηθεί ύστερα για τη δημιουργία της Ψηφιακής Υπογραφής ECDSA. Το δημόσιο αυτό κλειδί είναι το ίδιο με τη διεύθυνση του λογαριασμού του prover στο Ethereum.

Βήμα 4: Οι Witnesses απαντούν στέλνοντας το δικό τους δημόσιο κλειδί.

Ανάλυση: Ο witness από τη μεριά του, με το που δεχτεί το δημόσιο κλειδί του prover, καλεί τη μέθοδο **respondPublicKey**. Με αυτήν το αποθηκεύει και ύστερα παράγει ένα ζευγάρι κλειδιών RSA, αποθηκεύει το ιδιωτικό και απαντάει στον prover στέλνοντας του το δημόσιο κλειδί RSA που μόλις παρήγαγε.

Η παραπάνω διαδικασία γίνεται αυτόματα και δεν απαιτεί αλληλεπίδραση με το χρήστη-witness. Η αποστολή του κλειδιού γίνεται πάλι μέσω της προηγούμενης σύνδεσης Bluetooth η οποία παραμένει ανοιχτή.

Βήμα 5: Ο Prover υπογράφει την τοποθεσία του με το ιδιωτικό του κλειδί και μετά το κρυπτογραφεί με το δημόσιο κλειδί του Witness. Αυτό το κάνει ξεχωριστά για κάθε witness, στον οποίο και στέλνει το μήνυμα πάλι μέσω Bluetooth.

Ανάλυση: Αφού δεχτεί το δημόσιο κλειδί RSA του witness, ο prover το χρησιμοποιεί για να δημιουργήσει το κρυπτογραφημένο κείμενο (κρυπτογραφεί τις συντεταγμένες του) και μετά παράγει την Ψηφιακή υπογραφή κρυπτογραφημένου μηνύματος, χρησιμοποιώντας το δικό του ιδιωτικό κλειδί ECDSA. Τα παραπάνω στοιχεία τα στέλνει στον witness πάλι με Bluetooth. Πλέον η αποστολή του μηνύματος είναι ασφαλής, αφού αυτό είναι κρυπτογραφημένο, άρα δεν μπορεί να το διαβάσει κάποιος άλλος πέρα από το witness, ενώ

έχει και την ψηφιακή υπογραφή, επομένως δεν μπορεί να το έχει στείλει κάποιος άλλος πέρα από τον prover, ούτε μπορεί να έχει πειραχτεί από κάποιον τρίτο.

Όλη η παραπάνω διαδικασία γίνεται επίσης αυτόματα, χωρίς την ανάγκη αλληλεπίδρασης του prover με την εφαρμογή. Γίνεται με την κλήση της μεθόδου confirmPoL και αποτελεί το τελικό στάδιο της συμμετοχής του prover σε όλη τη διαδικασία της συναλλαγής. Πλεόν απομένει κάθε witness να στείλει την Απόδειξη Τοποθεσίας στο smart contract.

Βήμα 6: Ο Witness αποκρυπτογραφεί το μήνυμα με το ιδιωτικό του κλειδί και επαληθεύει την ταυτότητα του prover με το δημόσιο κλειδί του. Ύστερα επαληθευει ότι οι συντεταγμένες είναι σωστές.

Ανάλυση: Ο witness τώρα καλεί τη μέθοδο witnessCheck η οποία, αποκρυπτογραφεί το κρυπτογραφημένο μήνυμα χρησιμοποιώντας το ιδιωτικό κλειδί RSA που είχε αποθηκεύσει στο **βήμα 4** και μετά κάνει επαλήθευση της ταυτότητας του prover που έστειλε την τοποθεσία του, με το ιδιωτικό κλειδί που επίσης αποθήκευσε.

Έπειτα, ελέγχει αν οι συντεταγμένες που δέχτηκε είναι όντως κοντά του, συγκρίνοντας με τις δικές του. Αν είναι αποδεκτές, τότε είναι έτοιμος να στείλει Απόδειξη Τοποθεσίας στο Ethereum. Διαφορετικά αγνοεί τον prover και δεν κάνει τίποτα.

Βήμα 7: Ο Witness στέλνει στο Ethereum την Απόδειξη Τοποθεσίας. Εκεί το smart contract αφού ελέγξει αν ο witness είναι εγγεγραμμένος στο συμβόλαιο και το score μεταξύ prover και witness είναι αποδεκτό, το ενημερώνει.

Ανάλυση: Σε αυτό το βήμα ο witness πρέπει να στείλει στο Ethereum την Απόδειξη Τοποθεσίας του. Εδώ πρέπει να γίνει η δεύτερη κλήση μεθόδου του smart contract. Την καλεί στέλνοντας τη διεύθυνση του prover του οποίου προσπαθεί να αποδείξει την τοποθεσία.

Το smart contract με τη σειρά του ανανεώνει το score του Μηχανισμού αποφυγής συνεργασίας και αυξάνει τον αριθμό των επιτυχημένων Αποδείξεων Τοποθεσίας.

Βήμα 8: Όταν και ο τελευταίος witness στείλει την Απόδειξη Τοποθεσίας του στο smart contract, τότε ολοκληρώνεται η συναλλαγή.

Ανάλυση: Οταν ο τελευταίος witness στείλει την Απόδειξη Ταυτότητας εκ μέρους του prover, τότε το smart contract τη δέχεται και εκτελεί αυτόματα τη συναλλαγή μεταφέροντας τα ether στον τελικό προορισμό τους, όπως αυτός είχε οριστεί από τον prover στο **Βήμα 2**.

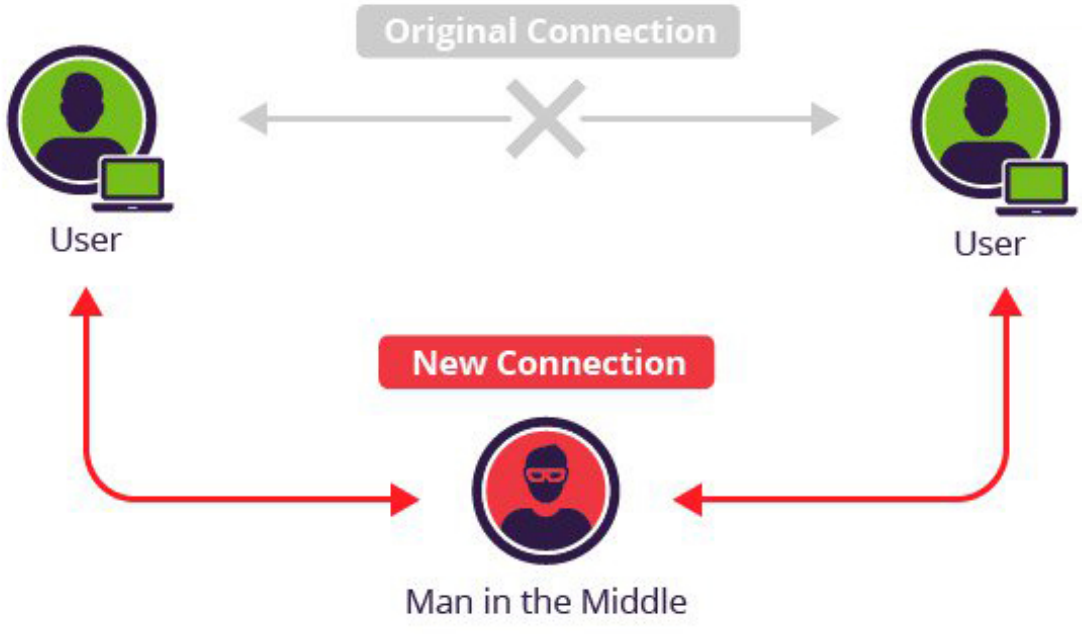
Στην περίπτωση που για κάποιο λόγο κάποιος witness απέρριψε την τοποθεσία του prover και δεν έστειλε Απόδειξη Τοποθεσίας στο smart contract, τότε ο prover θα πρέπει είτε να βρει άλλους witnesses για να ολοκληρώσει τη συναλλαγή, εκτελώντας δηλαδή ξανά τα βήματα ξεκινώντας αυτην τη φορά από το **Βήμα 3** και αφού έχει επιλέξει τον αριθμό των νέων witness (οι οποίοι θα πρέπει να είναι διαφορετικοί, είτε να ακυρώσει τη συναλλαγή πατώντας το κουμπί Cancel Transaction και καλώντας τη μέθοδο **cancelTransaction**.

3.4 Κίνδυνοι

Οι πιθανές επιθέσεις που μπορεί να δεχτεί μια εφαρμογή που βασίζεται στην ασύρματη επικοινωνία περιγράφονται αναλυτικά στα [7], [8], [9], [13]. Παρακάτω περιγράφονται αυτοί οι κίνδυνοι, ενώ εξηγείται και πως το προτεινόμενο πρωτόκολλο τους αντιμετωπίζει αποτελεσματικά.

3.4.1 Επίθεση ενδιαμέσου (Man-in-the-Middle Attack)

Στην MITM επίθεση, το σύνηθες σενάριο περιλαμβάνει δύο θύματα (victims) και έναν τρίτο κακόβουλο χρήστη (attacker). Ο attacker αποκτά πρόσβαση στο κανάλι επικοινωνίας μεταξύ των δύο θυμάτων και μπορεί να μεταχειριστεί τα μηνύματά τους. Η επίθεση μπορεί να περιγραφεί με τη βοήθεια του διαγράμματος παρακάτω.



Σχήμα 3.6: Παράδειγμα MITM επίθεσης.

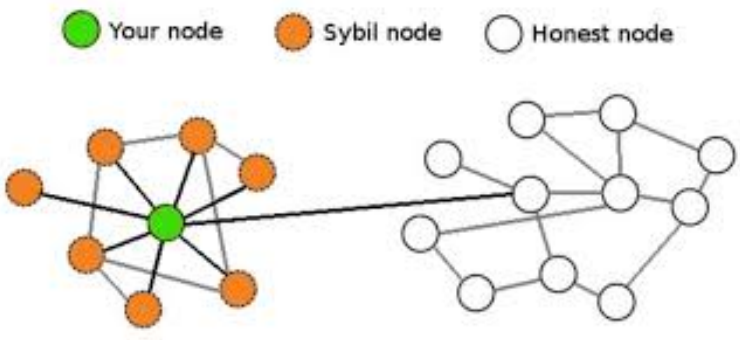
Συγκεκριμένα, τα θύματα προσπαθούν να ξεκινήσουν την ασφαλή επικοινωνία μεταξύ τους ανταλλάσσοντας τα δημόσια κλειδιά τους (M1 και M2). Ο attacker μπαίνει ανάμεσα στην επικοινωνία των θυμάτων, αποκτά αυτά τα κλειδιά και στέλνει πίσω στον καθένα το δικό του δημόσιο κλειδί (M3 και M4). Ύστερα, το πρώτο θύμα κρυπτογραφεί το μήνυμα του με το κλειδί του attacker και το στέλνει στο θύμα 2 (M5). Ο attacker υποκλέπτει το M5 και το αποκρυπτογραφεί με το δικό του ιδιωτικό κλειδί. Ύστερα ο attacker κρυπτογραφεί εκ νεου το αποκρυπτογραφημένο μήνυμα με το δημόσιο κλειδί του δεύτερου θύματος και του το στέλνει (M6). Ως συνέπεια, ο attacker έχει πείσει και τα δύο θύματα ότι χρησιμοποιούν ασφαλές κανάλι επικοινωνίας, ενώ στην πραγματικότητα έχει πρόσβαση σε όλα τα μηνύματα τους.

Αντιμετώπιση Επίθεσης MITM: Το πρωτόκολλο αντιμετωπίζει Man-in-the-Middle Attacks με τη χρήση των Elliptic Curve Digital Signatures που χρησιμοποιούνται στο Ethereum. Σε οποιαδήποτε επικοινωνία γίνεται έλεγχος του αποστολέα του μηνύματος ρωτώντας το Ethereum για την ταυτότητα του αποστολέα. Για την υπογραφή και την επιβεβαίωση της υπογραφής χρησιμοποιούνται το ιδιωτικό κλειδί που δημιουργήθηκε κατά τη δημιουργία του λογαριασμού στο Ethereum και το δημόσιο κλειδί το οποίο είναι η διεύθυνση του λογαριασμού στο Ethereum. Ταυτόχρονα γίνεται επιβεβαίωση από τον witness ότι ο αποστολέας είναι "εγγεγραμμένος" στο έξυπνο συμβόλαιο

Στην περίπτωση που ένας κακόβουλος χρήστης που είναι εγγεγραμμένος στο συμβόλαιο επιχειρήσει MITM επίθεση επιτυχώς, τότε στο μέλλον δεν θα μπορεί να ξαναεπιχειρήσει την επίθεση στον ίδιο χρήστη, λόγω του μηχανισμού αποφυγής συνεργασίας που έχουμε υλοποιήσει.

3.4.2 Sybil Επίθεση

Η Sybil επίθεση μελετήθηκε πρώτη φορά από τον Douceur το 2002 [35]. Αυτός ο τύπος επίθεσης εκμεταλλεύεται το γεγονός ότι σε ένα κατανεμημένο σύστημα οι απομακρυσμένες οντότητες αντιμετωπίζονται ως αφηρημένες πληροφοριακές οντότητες γνωστές ως ταυτότητες.



Σχήμα 3.7: Παράδειγμα Sybil επίθεσης.

Όταν το σύστημα αποτυγχάνει να εγγυηθεί ότι η κάθε λογική οντότητα αναφέρεται σε διαφορετική φυσική οντότητα, ένας επιτιθέμενος μπορεί να δημιουργήσει ένα μεγάλο αριθμό από ταυτότητες και να κυριαρχήσει στο δίκτυο ανώτερου στρώματος κοροϊδεύοντας τα πρωτόκολλα και να υποβιβάζει μηχανισμούς βασιζόμενος στον πλεονασμό του. Το σημαντικότερο συμπέρασμα της μελέτης του Douceur είναι ότι, σε ένα peer-to-peer σύστημα, ο μόνος τρόπος για να υπάρχει εγγύηση της ένα προς ένα αντιστοίχησης ταυτοτήτων σε φυσικές οντότητες που λειτουργούν τους συμμετέχοντες κόμβους είναι η χρήση μιας κεντρικής, έμπιστης αρχής για την έκδοση ταυτοτήτων.

Αντιμετώπιση Sybil Επίθεσης : Οι επιθέσεις τύπου Sibyl Attack δεν είναι εύκολα πραγματοποιήσιμες καθώς επιβάλλεται κατά το πρωτόκολλο όλοι οι witnesses να απαντήσουν με θετική Απόδειξη Τοποθεσίας. Αυτό σημαίνει ότι ο prover θα έπρεπε να μαζέψει πολλούς witnesses.

Επιπρόσθετα το πρωτόκολλο έχει, όπως είπαμε και το μηχανισμό αποφυγής συνεργασίας ο οποίος απαγορεύει σε συσκευές να είναι witness με τον ίδιο prover πολλές φορές. Αυτό δυσκολεύει ακόμα περισσότερο το έργο του prover, ο οποίος θα πρέπει να αποκτήσει πολλούς λογαριασμούς στο Ethereum και να κάνει εγγραφή στο συμβόλαιο με αυτούς, κάτι το οποίο έχει σημαντικό κόστος.

3.4.3 Υποκλοπή απόστασης (Distance hijacking)

Κατά την υποκλοπή απόστασης ένας κακόβουλος χρήστης υποκλέπτει την Απόδειξη Τοποθεσίας ενός ειλικρινούς prover. Με αυτόν τον τρόπο μπορεί να χρησιμοποιήσει αυτήν την απόδειξη που περιέχει λάθος τοποθεσία από την πραγματική του κακόβουλου χρήστη και με αυτόν τον τρόπο να πει ψέματα για την τοποθεσία του.

Αντιμετώπιση Distance hijacking : Τέτοιου είδους επιθέσεις είναι αδύνατες στο πρωτόκολλο μας, καθώς δεν είναι ο prover αυτός που στέλνει την Απόδειξης Τοποθεσίας στο Ethereum, αλλά οι witnesses. Επομένως δεν υπάρχει ενδεχόμενο υποκλοπής Απόδειξης Τοποθεσίας από κάποιον άλλο Prover, καθώς ποτέ κάποιος Prover δεν έχει ή μπορεί να έχει στα χέρια του κάποιας μορφής Απόδειξης.

Επίσης η Απόδειξη Τοποθεσίας του προτεινόμενου πρωτοκόλλου δεν είναι ένα μήνυμα, αλλά απλά μια επιβεβαίωση που πρέπει να προέρχεται από τους witnesses.

3.4.4 Συνεργασία prover-prover (Terrorist fraud)

Εδώ ο prover που βρίσκεται σε μία τοποθεσία συνεργάζεται με κάποιον άλλο prover που βρίσκεται σε διαφορετική τοποθεσία. Αφού ο δεύτερος συλλέξει την Απόδειξη τοποθεσίας του από τους γειτονικούς του witnesses, στέλνει αυτήν την απόδειξη στον prover, που με αυτόν τον τρόπο αποδεικνύει λανθασμένα ότι βρίσκεται σε διαφορετική τοποθεσία από ό,τι πραγματικά βρίσκεται.

Αντιμετώπιση Terrorist fraud : Είναι αδύνατη η συνεργασία μεταξύ δύο Prover ώστε να παράξει ο ένας την Απόδειξη Ταυτότητας και να την χρησιμοποιήσει ο άλλος, λόγω των ψηφιακών υπογραφών. Ο μόνος τρόπος να γίνει κάτι τέτοιο είναι αν ο ένας prover γνωρίζει το ιδιωτικό κλειδί του άλλου. Παρόλα αυτά και σε αυτήν την περίπτωση, η χρήση της τεχνολογίας μικρής εμβέλειας Bluetooth, επιβάλλει στον prover να είναι κοντά στους witnesses, γεγονός που καθιστά τέτοιου είδους προσπάθεια απάτης ανούσια.

3.4.5 Συνεργασία witness-witness

Οι μάρτυρες μπορούν να συνεργαστούν μεταξύ τους ώστε να επιβεβαιώσουν μια ψευδή τοποθεσία του prover

Αντιμετώπιση συνεργασίας W-W : Η συνεργασία μεταξύ witnesses δεν είναι εφικτή καθώς δεν επικοινωνούν μεταξύ τους κατά τη διάρκεια εκτέλεσης του πρωτοκόλλου. Στην περίπτωση που επικοινωνήσουν, τότε στην ουσία το μόνο που μπορούν να κάνουν είναι μια επίθεση τύπου Sybil Attack, στην οποία αναφερθήκαμε παραπάνω.

3.4.6 Συνεργασία prover-witness

Ο prover συνεργάζεται με έναν ή πολλούς witness ώστε αυτοί να επιβεβαιώσουν πως βρίσκεται σε μια ψευδή τοποθεσία.

Αντιμετώπιση συνεργασίας P-W : Ο Prover δεν έχει τη δυνατότητα να δει την απάντηση του Witness καθώς δεν την επιστρέφει στον ίδιο αλλά ο witness τη στέλνει κατευθείαν στο Blockchain. Στην περίπτωση που υπάρξει κάποια συνεργασία, τότε οι υπόλοιποι Witness θα απορρίψουν τη συναλλαγή. Επίσης ο μηχανισμός αποφυγής συνεργασίας, μπλοκάρει τέτοιες συνεργασίες όταν επαναλαμβάνονται**.**

3.4.7 Επανάληψη παλιάς Απόδειξης Ταυτότητας

Ο prover δεν χρειάζεται καν να συνεργαστεί με witnesses καθώς μπορεί απλά να αναπαράξει και να χρησιμοποιήσει μια παλιά Απόδειξη Ταυτότητας.

Αντιμετώπιση επανάληψης παλιάς ΑΤ : Η επαναχρησιμοποίηση παλιάς Απόδειξης Ταυτότητας είναι αδύνατη καθώς ο prover δεν μπορεί να αποδείξει τη γεωγραφική του θέση στέλνοντας μια Απόδειξη Ταυτότητας. Αντίθετα πρέπει οι witnesses να την αποδείξουν.

Επίσης δεν αποθηκεύεται κάπου κάποια μορφή Απόδειξης Ταυτότητας, ώστε ο prover να μπορεί να την αναπαράξει.

4. Περιγραφή Εφαρμογής

Συνοπτικά, στο πλαίσιο της εργασίας αυτής, φτιάχτηκε μια εφαρμογή σε Android, που να μπορεί εύκολα να χρησιμοποιηθεί σε κινητά τηλέφωνα τύπου "smartphone", η οποία να εκτελεί μια συναλλαγή σε Ethereum Blockchain, αφού πρώτα αποδείξει την τοποθεσία του.

Η απόδειξη της τοποθεσίας γίνεται αφού γίνει αίτηση για συναλλαγή στο Ethereum, με την επικοινωνία μέσω Bluetooth με άλλους ιδιοκτήτες κινητών συσκευών, οι οποίοι θα έχουν την ίδια εφαρμογή εγκατεστημένη και θα επιβεβαιώνουν την τοποθεσία στέλνοντας πίσω στον χρήστη την Απόδειξη Ταυτότητας PoL. Ο χρήστης με τη σειρά του στέλνει την PoL στο Ethereum Network το οποίο με τη χρήση smart contract εκτελεί τη συναλλαγή ή την απορρίπτει αν το PoL δεν ήταν αποδεκτό. Φυσικά κατά τη διάρκεια της επικοινωνίας όλα τα μηνύματα είναι ψηφιακά υπογεγραμμένα και κρυπτογραφημένα.

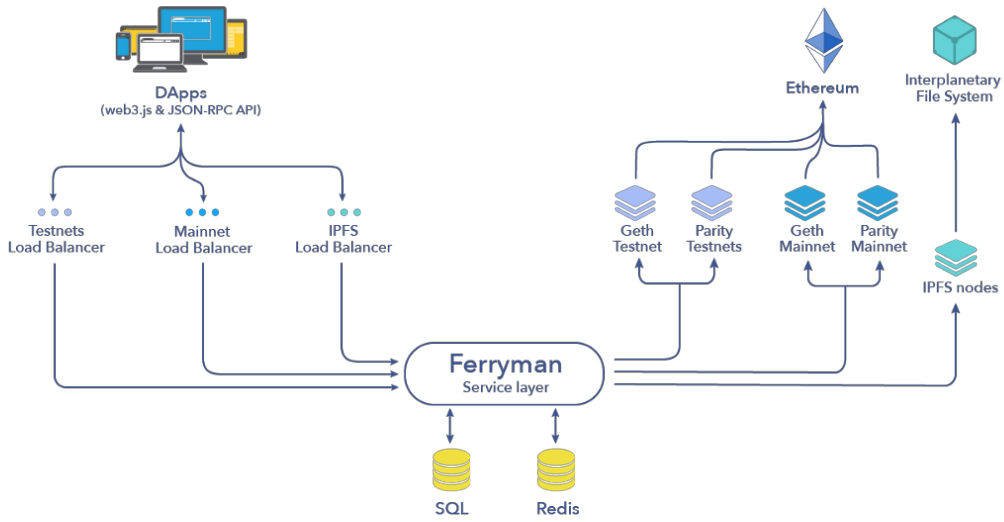
Από εδώ και πέρα, θα χρησιμοποιούμε τον όρο *prover* για τον χρήστη που προσπαθεί να αποδείξει την τοποθεσία του, τον όρο *witness* για τους χρήστες που επικοινωνούν με τον *prover* για να του προσφέρουν την απόδειξη της τοποθεσίας του, και ως *verifier* τον χρήστη που θα διαβάσει και θα αποδεχτεί ή θα απορρίψει την απόδειξη. Στην περίπτωση της εφαρμογής μας ο *verifier* θα είναι το smart contract στο Ethereum.

4.1 Εργαλεία και τεχνολογίες

Στο σημείο αυτό θα παρουσιαστούν τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Συγκεκριμένα για την ανάπτυξη της εφαρμογής σε Android χρησιμοποιήθηκε Java στο Android Studio, ενώ για όλα τα υπόλοιπα που έχουν να κάνουν με το Ethereum και την επικοινωνία με αυτό, χρησιμοποιήθηκαν τα Geth, Web3j, Solidity και Metamask, ενώ για το testing τοπικά στον υπολογιστή χρησιμοποιήθηκε το Ganache και Truffle.

4.1.1 Infura

Το Infura [29] είναι μια κλιμακώσιμη back-end υποδομή για την κατασκευή dapps στο Ethereum blockchain. Πρόκειται για μια μέθοδο σύνδεσης με το δίκτυο Ethereum χωρίς την ανάγκη εκτέλεσης ενός πλήρους κόμβου και η υπηρεσία παρέχεται από την εταιρεία Consensys. Η πιο απλή διεπαφή για την πρόσβαση στο Ethereum φιλοξενείται μέσω των cloud servers του Amazon και είναι η πιο κοινή μέθοδος που χρησιμοποιούν οι προγραμματιστές dapp για τη σύνδεση με το δίκτυο Ethereum.



Σχήμα 4.1: Αρχιτεκτονική ενός πλήρη Infura node

Το Infura είναι μια συλλογή από πλήρεις κόμβους στο δίκτυο Ethereum που επιτρέπουν στους προγραμματιστές να συνδεθούν με αυτούς μέσω της διεπαφής τους. Ως εκ τούτου, ένα σημαντικό μέρος της κυκλοφορίας dapp τρέχει μέσω του Infura - λόγω της ευκολίας χρήσης του, χωρίς την απαίτηση από τους προγραμματιστές να τρέχουν έναν πλήρη κόμβο σε τοπικό επίπεδο και να τον συντηρούν.

Η ιδέα της εκφόρτωσης της ανάγκης εκτέλεσης ενός πλήρους κόμβου κυριαρχεί μεταξύ των προγραμματιστών, οι οποίοι στη συνέχεια μπορούν να εστιάσουν περισσότερο τις προσπάθειές τους στη δημιουργία των dapps και στους άλλους τομείς της εφαρμογής τους, παρά στη συνεπή διαχείριση της σύνδεσης του πλήρους κόμβου στο δίκτυο. Το Infura παρέχει πολυάριθμα εργαλεία ανάπτυξης, τεκμηρίωση και κλειδιά API για συνεργασία με το Ethereum - επιτρέποντας ακόμη και την κατανεμημένη αποθήκευση μέσω του IPFS. Η πύλη IPFS της Infura είναι ένα χρήσιμο χαρακτηριστικό του σχεδιασμού της και η συμφωνία του IPFS με blockchains θα πρέπει να συνεχίσει να τροφοδοτεί την ανάπτυξη της χρήσης του μεταξύ των developers της dapp.

Η χρήση του Infura προτιμήθηκε διότι το τρέξιμο ενός πλήρους κόμβου σε τοπικό επίπεδο μιας κινητής συσκευής απαιτεί πολλή μνήμη και επομένως το τρέξιμο της εφαρμογής θα ήταν μόνο εφικτό αποκλειστικά σε πολύ μοντέρνα smartphones.

4.1.2 Geth

Το geth [30] είναι μια command line διεπαφή που έχει υλοποιηθεί στη γλώσσα προγραμματισμού Go και εκτελεί έναν πλήρη Ethereum κόμβο. Με το geth μπορεί κανείς να γίνει μέλος του Ethereum δικτύου και να φτιάξει λογαριασμό, είτε στο mainnet, είτε σε testnets τα οποία αποτελούν δίκτυα που είναι παρόμοια με το mainnet, αλλά χρησιμοποιούν διαφορετικά consensus και τα Ether δεν αντιστοιχούν πραγματικά χρήματα. Τα testnets επιτρέπουν στους προγραμματιστές να κάνουν deploy τα συμβόλαιά τους και να εξετάσουν τη σωστή λειτουργία τους σε ένα ρεαλιστικό περιβάλλον, χωρίς να χρειαστεί να πληρώσουν. Εμείς κάναμε deploy το smart contract μας στο Rinkeby testnet.

Μεταξύ άλλων με το geth μπορούμε επίσης να:

- Εξορύξει ether
- Μεταφέρει ether
- Δημιουργήσει συμβόλαια και να καλέσει μεθόδους αυτών
- Δημιουργήσει ιδιωτικό Ethereum blockchain
- Γενικότερα να εξερευνήσει το blockchain του Ethereum.

4.1.3 Web3j

Το πακέτο web3j[31] είναι μια συλλογή βιβλιοθηκών που επιτρέπουν την αλληλεπίδραση με ένα τοπικό ή απομακρυσμένο κόμβο του Ethereum blockchain, χρησιμοποιώντας HTTP ή IPC σύνδεση. Με άλλα λόγια είναι μια διεπαφή επικοινωνίας της Java και του Ethereum blockchain. Είναι δηλαδή ο πιο διαδεδομένος για να κάνει αναφορές η Java σε αντικείμενα στο Blockchain, όπως τα έξυπνα συμβόλαια, τα δεδομένα τους, τις μεθόδους τους, τις διευθύνσεις και τα υπολοιπα λογαριασμών, και πολλά άλλα. Επιλέχθηκε το Web3j και όχι το Web3.js που είναι για javascript, για το γεγονός ότι η java μπορούσε να χρησιμοποιηθεί ευκολότερα με τη java που χρησιμοποιείται για τις Android εφαρμογές στο Android Studio.

4.1.4 Solidity

Η Solidity [32] είναι μία “συμβολαιοστρεφής”, high-level γλώσσα προγραμματισμού που εκτελείται στο Ethereum Virtual Machine και έχει δημιουργηθεί για να μεγεθύνει τις δυνατότητες της μηχανής αυτής. Χρησιμοποιείται κυρίως για τη

δημιουργία έξυπνων συμβολαίων για το Ethereum blockchain. Ο ίδιος ο πηγαίος κώδικας της γλώσσας Ethereum είναι γραμμένος σε Solidity

Έχει παρόμοια σύνταξη με αυτήν της JavaScript, επομένως είναι εύκολα κατανοήσιμη από έναν μεγάλο αριθμό προγραμματιστών. Είναι μια στατικού τύπου γλώσσα, που υποστηρίζει την κληρονομικότητα με παρόμοιο τρόπο με άλλες γλώσσες προγραμματισμού (όπως η C++).

4.1.5 Metamask

Το Metamask [33] είναι ένα plugin διαθέσιμο για διαφόρους browsers, όπως το Google Chrome, το Mozilla Firefox, το Opera και το Brave Browser. Πρόκειται στην ουσία για μία γέφυρα η οποία δίνει στον browser πρόσβαση στις κατανεμημένες εφαρμογές, που βασίζονται για τη λειτουργία τους στο δίκτυο Ethereum. Το μεγάλο πλεονέκτημα που προσφέρει είναι το γεγονός ότι με την χρήση του, η πρόσβαση στις εφαρμογές αυτές, δεν απαιτεί την εκτέλεση του πλήρους Ethereum κόμβου στον υπολογιστή του χρήστη, όπως απαιτεί το ειδικό browser για το Ethereum, Mist.

Στο πλαίσιο της εργασίας, χρησιμοποιήθηκε για την εύχρηστη διεπαφή που προσφέρει στον χρήστη και πιο συγκεκριμένα για τη δημιουργία και την παρακολούθηση τον λογαριασμών Ethereum στο Rinkeby testnet.

4.1.6 Ganache

Το Ganache [34] είναι μια εφαρμογή που χρησιμοποιείται για την ανάπτυξη εφαρμογών blockchain σε ιδιωτικό blockchain και όχι σε κάποιο testnet ή το mainnet. Αυτό σημαίνει ότι είναι εξαιρετικά χρήσιμο για το αρχικό testing των συμβολαίων που αναπτύσσονται από τους προγραμματιστές, πριν τα κάνουν deploy σε κάποιο testnet ή mainnet, αφού δεν χρειάζεται να κατεβάσουν τοπικά στον υπολογιστή τους όλα τα blocks κάποιου δικτύου, επιτρέποντας ταχύτερο προγραμματισμό συμβολαίων και ευκολότερο debugging αυτών. Επίσης περιλαμβάνει όλες τις δημοφιλείς RPC (Remote Procedure Call) συναρτήσεις και δυνατότητες (πχ events) και μπορεί να εκτελεστεί αιτιοκρατικά, καθιστώντας έτσι την διαδικασία ανάπτυξης πολύ γρηγορότερη. Το Ganache CLI δηλαδή είναι ένα εργαλείο που προσφέρει μεγάλη ευκολία και πολλές δυνατότητες στον προγραμματιστή.

Παρέχει στον χρήστη workspaces με πολλούς λογαριασμούς με 100 Ether με τους οποίους μπορεί ο προγραμματιστής να πειραματιστεί, εκτελώντας συναλλαγές μεταξύ των λογαριασμών και συμβολαίων που κάνει deploy ο ίδιος.

Υπάρχει και έκδοση Ganache σε command line που λέγεται Ganache CLI.

4.1.7 Truffle

Το Truffle [35] είναι ένα παγκόσμιας κλάσης περιβάλλον ανάπτυξης εφαρμογών και τεσταρίσματος αυτών για blockchains που χρησιμοποιούν το Ethereum Virtual Machine, που στοχεύει να κάνει ευκολότερη τη ζωή των προγραμματιστών. Με το Truffle έχουμε τις εξής δυνατότητες:

- Compilation έξυπνων συμβολαίων, σύνδεση και deployment αυτών σε κάποιο δίκτυο, είτε ιδιωτικό, είτε δημόσιο. Αυτό γίνεται με την εντολή truffle migrate — compile-all
- Αυτόματο testing για γρήγορο deployment με την εντολή truffle test
- Διαχείριση δικτύων για deployment συμβολαίων σε οποιονδήποτε αριθμό δημοσίων και ιδιωτικών δικτύων.
- Διαχείριση πακέτων με το EthPM και NPM, χρησιμοποιώντας το standard ERC190
- Interactive console για άμεση επικοινωνία με τα συμβόλαια και τους λογαριασμούς. Αυτό γίνεται με την εντολή truffle console.
- Διάφορες άλλες λειτουργίες.

5. Σχεδιασμός και Υλοποίηση

Η εφαρμογή που υλοποιήθηκε στο πλαίσιο της εργασίας, αποτελείται από δύο βασικά components:

- Το Ethereum Smart Contract που θα είναι γραμμένο σε Solidity και θα είναι υπεύθυνο για την ασφαλή εκτέλεση των συναλλαγών. Θα προσφέρει δυνατότητα εγγραφής των χρηστών στο Smart Contract έναντι αντιτίμου, μεταφοράς ether κατά την αίτηση συναλλαγών και κατά την ολοκλήρωση τους, ενώ θα λειτουργεί και ως βάση δεδομένων, κρατώντας ιστορικό όλων των συναλλαγών, PoL, χρηστών κ.α.

- Την Android εφαρμογή που θα είναι γραμμένη σε Java στο Android Studio και θα είναι υπεύθυνη για την ασφαλή επικοινωνία μεταξύ smartphones με τη χρήση Bluetooth, καθώς και με την απευθείας επικοινωνία με το Ethereum blockchain.

5.1 Υλοποιηση Smart Contract

Στο πλαίσιο αυτό θα εξηγηθούν αναλυτικά οι δομές δεδομένων και οι βασικές μέθοδοι του έξυπνου συμβολαίου. Η έκδοση της Solidity που χρησιμοποιήθηκε είναι η Solidity 0.5.9+. Το αρχικό τεστάρισμα της εφαρμογής έγινε τοπικά με τη χρήση του Ganache και του Truffle, ενώ μετά έγινε deploy στο Rinkeby Testnet, με σκοπό τον έλεγχο της σωστής λειτουργίας του σε πιο ρεαλιστικό περιβάλλον.

5.1.1 Δομές Δεδομένων

```
// Store users, creditScores, number of transactions between pair addresses
mapping(address => bool) public users;
mapping(address => mapping(address => uint)) private
 usersCreditScore;
mapping(address => mapping(address => uint)) private usersNoTrans;

// Store transactions
mapping(uint => Transactions) private transactions;
mapping(address => uint) private lastTransaction;
uint public transactionsCount;
```

users: Mapping που παίρνει τη διεύθυνση ενός λογαριασμού Ethereum και επιστρέφει false αν ο λογαριασμός δεν έχει εγγραφεί στο smart contract ή true αν έχει εγγραφεί. Άμα ο χρήστης δεν είναι εγγεγραμμένος δεν μπορεί να εκτελέσει συναλλαγές μέσω του συμβολαίου.

usersCreditScore: Mapping που παίρνει τις διευθύνσεις δύο λογαριασμών και επιστρέφει το credit score του ζευγαριού. Χρησιμοποιείται από το συμβόλαιο για την αποφυγή συχνής συνεργασίας κοινών ζευγαριών prover - witness.

usersNoTrans: Mapping που παίρνει τις διευθύνσεις δύο λογαριασμών και επιστρέφει τον αριθμό των παλιών συναλλαγών του ζευγαριού.

transactions: Mapping που κρατάει ιστορικό όλων των συναλλαγών που έχουν εκτελεστεί στο συμβόλαιο. Παίρνει έναν θετικό ακέραιο n και επιστρέφει την n-οστή συναλλαγή που έχει εκτελεστεί. Το struct της συναλλαγής φαίνεται παρακάτω:

```
struct Transactions {
        uint id;
        string status;
        address from;
        address to;
        uint price;
        uint totalProofs;
        uint numProofs;
    }
```

Το struct Transactions περιέχει το id της συναλλαγής, ένα string που αντιστοιχεί στο status της συναλλαγής (“pending”, “received”, “returned”), ένα address του λογαριασμού που την εκτέλεσε, ένα address του προορισμού, ένα uint του αριθμού των Milliether που πρόκειται να μεταφερθούν, ένα uint του συνολικού αριθμού των PoL που θα πρέπει να περιμένει το συμβόλαιο και ένα uint των PoL που έχει ήδη δεχτεί το συμβόλαιο.

lastTranaction: Mapping που παίρνει τη διεύθυνση ενός λογαριασμού και επιστρέφει το id της τελευταίας συναλλαγής αυτού του λογαριασμού. Χρησιμοποιείται για την εύρεση της τελευταίας συναλλαγής κάθε λογαριασμού.

transactionsCount: Uint που κρατά τον συνολικό αριθμό των συναλλαγών που έχουν εκτελεστεί στο πλαίσιο του έξυπνου συμβολαίου.

5.1.2 Μέθοδοι Συμβολαίου

```
function addUser () public payable {
        if (msg.value < 20000000000000000) return;
        if (!(users[msg.sender])) {
            users[msg.sender] = true;
        }
    }
```

Η μέθοδος **addUser** είναι payable, που σημαίνει ότι για να κληθεί από κάποιο χρήστη, απαιτείται μια επιπλέον παράμετρος που αποτελεί το ποσό που μεταφέρει ο χρήστης στο συμβόλαιο. Για να καταφέρει να εγγραφεί, ο χρήστης πρέπει να μεταφέρει το ποσό των 0.02 ether στο συμβόλαιο. Η επιλογή της ανάγκης μεταφοράς κάποιου αντιτίμου για την εγγραφή στο έξυπνο συμβόλαιο έγινε για την αποφυγή δημιουργίας πολλών διαφορετικών διευθύνσεων από malicious users για την χρήση ως witness. Το ποσό των 0.02 ether επιλέχτηκε αυθαίρετα, καθώς κατά τη συγγραφή της εργασίας αντιστοιχεί σε περίπου 5$.

```
function transferToContract (address _to, uint price, uint numProofs)
public payable
    returns(bool success) {
        if (msg.value != price*1000000000000000) return false;
        if (!users[msg.sender]) return false;
        transactionsCount++;
        lastTransaction[msg.sender] = transactionsCount;
        transactions[transactionsCount].id = transactionsCount;
        transactions[transactionsCount].price = price;
        transactions[transactionsCount].status = "pending";
        transactions[transactionsCount].from = msg.sender;
        transactions[transactionsCount].to = _to;
        transactions[transactionsCount].totalProofs = numProofs;
        transactions[transactionsCount].numProofs = 0;
        transactions[transactionsCount].numScams = 0;
        return true;
    }
```

Η μέθοδος **transferToContract** είναι η μέθοδος μεταφοράς ether από τον prover στο έξυπνο συμβόλαιο. Είναι επίσης payable, ώστε να επιτρέπεται η μεταφορά ether. Αρχικά γίνεται ένας έλεγχος για το αν τα χρήματα που έστειλε ο χρήστης (msg.value) είναι ίδια με αυτά που δήλωσε (price). Ύστερα ελέγχεται αν ο χρήστης που την κάλεσε είναι όντως εγγεγραμμένος στην υπηρεσία. Μετά, αποθηκεύονται όλα τα απαραίτητα δεδομένα στην επόμενη θέση του mapping transactions, ενώ αποθηκεύεται και η θέση της τελευταίας συναλλαγής του prover στο lastTransaction.

```
function witnessProof (address proverAddress) public
        returns (bool success) {
                if (!users[msg.sender]) return false;
                uint id = lastTransaction[proverAddress];
                uint limit = transactions[id].totalProofs;
                uint score = increaseCreditScore(limit, proverAddress);

                if (transactions[id].numProofs == limit) return false;
                transactions[id].numProofs++;

                if (score >= 20) {
                        transactions[id].numProofs--;
                        return false;
                }
                else if (transactions[id].numProofs == limit) {
                        confirmTransactionbyWitness(proverAddress);
                        return true;
                }
                return false;
        }
```

Η μέθοδος **witnessProof** είναι η μέθοδος με την οποία ο witness επαληθεύει την τοποθεσία του prover στο έξυπνο συμβόλαιο. Αφού γίνει έλεγχος για το αν ο witness που καλεί τη μέθοδο είναι εγγεγραμμένος χρήστης, αυξάνεται το credit score μεταξύ του prover και του

witness. Αν ο τωρινός αριθμός των PoL που έχουν καταχωρηθεί είναι ο ίδιος με το συνολικό, τότε η μέθοδος δεν κάνει τίποτα. Αν όχι, τότε θα είναι μικρότερος και θα γίνει έλεγχος αν το score μεταξύ prover και witness είναι αποδεκτό (μικρότερο του 20). Αν είναι και το συμβόλαιο έχει δεχτεί όλα τα PoL, τότε καλείται η confirmTransactionByWitness και γίνεται επιτυχημένη ολοκλήρωση της συναλλαγής. Διαφορετικά τελειώνει η μέθοδος και η μεταφορά των χρημάτων από το συμβόλαιο στον προορισμό δε γίνεται μέχρι να μαζευτούν όλα τα PoL που έχουν δηλωθεί από τον prover.

Στην μέθοδο witnessProof καλούνται οι μέθοδοι increaseCreditScore και ύστερα η μέθοδος confirmTransactionByWitness , όπως φαίνονται παρακάτω:

```
function increaseCreditScore (uint numProofs, address proverAddress)
private
        returns (uint result) {
                if (!users[msg.sender]) return 0;
                usersNoTrans[msg.sender][proverAddress]++;
                usersCreditScore[msg.sender][proverAddress] +=
20*usersNoTrans[msg.sender][proverAddress] / numProofs;
                result = usersCreditScore[msg.sender][proverAddress];
                return result;
        }
```

Η μέθοδος **increaseCreditScore**, αφού επίσης ελέγξει ότι ο λογαριασμός που την κάλεσε είναι εγγεγραμμένος, ανανεώνει το Credit Score μεταξύ των λογαριασμών prover και witness, σύμφωνα με τη σχέση:

$$Vxy = V + 20\frac{Nxy}{U},$$

όπου, όπως είπαμε και παραπάνω, V είναι το Vxy πριν τη συναλλαγή, Nxy είναι ο αριθμός των προηγούμενων συναλλαγών με prover X και witness Y και U είναι ο συνολικός αριθμός των PoL που απαιτούνται για τη συναλλαγή.

Η συνάρτηση είναι private ώστε να μην μπορεί να κληθεί εκτός του συμβολαίου από χρήστες.

```
function confirmTransactionByWitness (address proverAddress) private
        returns (bool) {
```

```
                uint id = lastTransaction[proverAddress];
                if (!stringsEqual(transactions[id].status, "pending")) return
false;
                address addr = transactions[id].to;
                address payable destination = address(uint160(addr));
        destination.transfer(transactions[id].price * 1000000000000000);
        transactions[id].status = "received";
        return true;
        }
```

Η μέθοδος **confirmTransactionByWitness** είναι επίσης private για να μην μπορεί να την καλέσει κάποιος χρήστης και να εκμεταλλευτεί το σύστημα. Η μέθοδος καλείται μόνο όταν έχουν μαζευτεί όλα τα PoL.

Όταν μαζευτεί και το τελευταίο PoL, τότε χρησιμοποιείται η μέθοδος της Solidity transfer για την τελική μεταφορά των δηλωμένων milliether από το συμβόλαιο στον προορισμό που είχε επίσης δηλωθεί από τον prover που έκανε αρχικά την αίτηση συναλλαγής μέσω της transferToContract. Το status της συναλλαγής γίνεται “received” αφού πλέον έχει λάβει τα ether ο προορισμός και η συναλλαγή έχει στην ουσία κλειδωθεί και δεν μπορεί να επεξεργαστεί ξανά στο μέλλον.

```
    function returnTransaction () public
        returns (bool) {
                if (!users[msg.sender]) return false;
                uint id = lastTransaction[msg.sender];
                if (!stringsEqual(transactions[id].status, "pending")) return
false;
                address addr = transactions[id].from;
                address payable destination = address(uint160(addr));
        destination.transfer(transactions[id].price * 1000000000000000);
        transactions[id].status = "returned";
        return true;
        }
```

Η **returnTransaction** καλείται από τον prover για να κάνει επιστροφή των milliether της τελευταίας δηλωμένης συναλλαγής του από το συμβόλαιο στο λογαριασμό του. Είναι στην ουσία ίδια με την confirmTransactionByWitness, απλά με προορισμό της transfer το λογαριασμό του prover και αλλαγή του status της συναλλαγής σε “returned”.

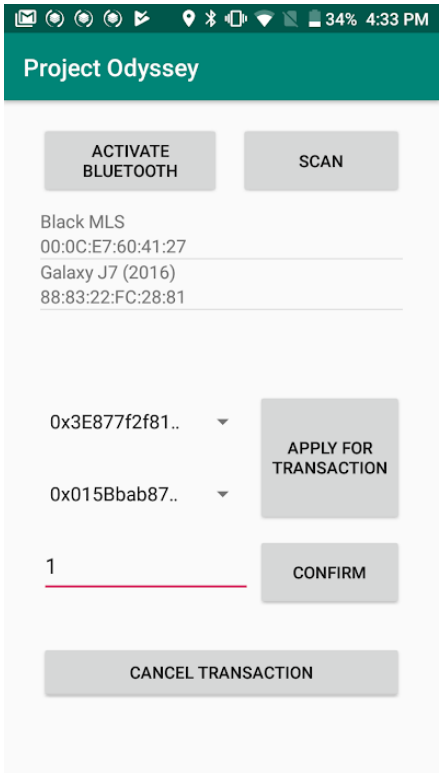
Χρησιμοποιείται από τον prover αν αυτός μετανιώσει τη συναλλαγή ή για ασφάλεια αν για κάποιο λόγο δεν μπορεί να βρει witness να του αποδείξουν την τοποθεσία.

5.2 Υλοποιηση Android Εφαρμογής

Στο πλαίσιο αυτό θα εξηγηθούν οι πιο σημαντικές συναρτήσεις της εφαρμογής που υλοποιήθηκε σε Java 8.0. Η εφαρμογή τρέχει σε minimum SDK version API 24 το οποίο είναι το Android 7.0 Nougat, αλλά έχει τεσταριστεί και σε API 27 που αντιστοιχεί στο Android 8.1 Oreo.

Ο πυρήνας της εφαρμογής αποτελείται από τρία classes, το MainPage το οποίο είναι κυρίως υπεύθυνο για το UI της εφαρμογής, το BluetoothConnectionService που είναι υπεύθυνο για τις συνδέσεις με Bluetooth και την επικοινωνία μέσω αυτού, και το Transaction που ευθύνεται για τα remote calls στο έξυπνο συμβόλαιο μέσω web3j.

Η οθόνη της εφαρμογής φαίνεται στην παρακάτω εικόνα:



Σχήμα 5.1: Κεντρική οθόνη εφαρμογής Dapp.

5.2.1 Βασικές Συναρτήσεις Εφαρμογής

Αρχικά, είναι απαραίτητο η εφαρμογή να συνδεθεί με τον κατάλληλο κόμβο στο δίκτυο του Ethereum Blockchain. Αυτό, όπως είπαμε, γίνεται με τη βοήθεια του Infura. Παρακάτω φαίνεται ο κώδικας για την εδραίωση της σύνδεσης αυτής.

```
myPrivateKeys = MainPage.this.getResources().getStringArray(R.array.privateKeys);
myPublicKey = mySpinner.getSelectedItem().toString();
myPrivateKey = myPrivateKeys[mySpinner.getSelectedItemPosition()];
credentials = Credentials.create(myPrivateKey);
fastRawTxMgr = new FastRawTransactionManager(web3j, credentials);
transaction = Transaction.load(contractAddress, web3j, fastRawTxMgr, gasPrice, gasLimit);
```

Κατά την onCreate που τρέχει με το που ξεκινήσει η εφαρμογή, αρχικοποιούνται τα **credentials** του λογαριασμού στο Ethereum με το δημόσιο κλειδί, που αποτελεί και τη διεύθυνση του λογαριασμού, καθώς και με το ιδιωτικό κλειδί που απαιτείται για την εκτέλεση των μεθόδων του έξυπνου συμβολαίου μέσω της web3j. Τα public και private keys του λογαριασμού πρέπει να εισαχθούν στην εφαρμογή προγραμματιστικά πριν εγκατασταθεί η εφαρμογή στο smartphone.

Έπειτα γίνεται deploy το έξυπνο συμβόλαιο με τη μέθοδο load από το Transaction class. Αφού γίνει deploy το συμβόλαιο και γίνουν οι απαραίτητες αρχικοποιήσεις των γραφικών στοιχείων του συμβολαίου, δημιουργείται, πάλι στην onCreate, ένας Location Listener που "ακούει" τις αλλαγές στην τοποθεσία του κινητού και τις αποθηκεύει στη μεταβλητή **currentLocation**.

```
activateLocationProvider();
enableBT(null);
btnEnableDisable_Discoverable(0);

// Start discovering other devices
mBluetoothConnection = new BluetoothConnectionService(MainPage.this);
btnDiscover(null);
```

Ύστερα ενεργοποιείται ο LocationProvider, ο οποίος όπως είπαμε είναι ο GPS_PROVIDER αν η συσκευή βρίσκεται σε εξωτερικό χώρο, ή διαφορετικά ενεργοποιείται ο NETWORK_PROVIDER αν είμαστε σε εσωτερικό. Μετά ενεργοποιείται το Bluetooth της συσκευής, αν δεν έχει ενεργοποιηθεί ήδη, το discoverability της και τέλος αρχίζει να ψάχνει για άλλους υποψήφιους witnesses που έχουν ενεργοποιημένο το Bluetooth τους και είναι paired με τη συσκευή μας. Οι παραπάνω λειτουργίες απαιτούν

έγκριση από τον χρήστη για να ενεργοποιηθούν, ενώ είναι διαθέσιμες στο χρήστη και με ξεχωριστό κουμπί στην κορυφή της εφαρμογής.

Αφού επιλεχθούν οι witnesses από το ListView και ο προορισμός και το ποσό των milliether που θέλει να μεταφέρει ο χρήστης από το λογαριασμό του στον προορισμό, ο prover κάνει Apply for Transaction ώστε να στείλει τα milliether στο συμβόλαιο, το οποίο μετά θα περιμένει τους witnesses να επαληθεύσουν την τοποθεσία του prover με τα PoL τους.

```
public void applyTransaction(View view) {
    Thread tProcedure = new Thread( new Runnable() {
        @Override
        public void run() {
            int i = 0;
            int numOfProofs = 0;
            for (boolean isClicked : colors){
                if (isClicked) numOfProofs++;
                i++;
            }
            int finalNumOfProofs = numOfProofs;

            Thread tMakeTransaction = new Thread( new Runnable() {
                @Override
                public void run() {
                    Log.d(TAG, "Making transaction...");
                    startProgressDialogTx.sendEmptyMessage(0);
                    transferToContract(finalNumOfProofs);
                    stopProgressDialogTx.sendEmptyMessage(0);
                }
            });
            if (finalNumOfProofs > 0) {
                fastRawTxMgr.setNonce( BigInteger.valueOf( -1 ) );
                tMakeTransaction.start();
                try {
                    tMakeTransaction.join();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    } );
    tProcedure.start();
}
```

Η **applyTransaction** καλείται σε ξεχωριστό Thread για να μπορεί να κάνει αλλαγές στο UI της εφαρμογής. Συγκεκριμένα πρέπει να μπορεί να ανοίξει ένα ProgressDialog ώστε να μην επιτρέπει στον χρήστη να κάνει άλλες ενέργειες όσο γίνεται η αίτηση στο Ethereum.

Ύστερα αφού μετρηθεί το πόσες συσκευές έχουν επιλεχθεί από τον prover ως witnesses, δημιουργείται νέο thread για την κλήση της μεθόδου του smart contract. Ο λόγος που απαιτείται καινούριο thread είναι για το σωστό συγχρονισμό των ενεργειών. Πρέπει πρώτα να ολοκληρωθεί συναλλαγή πριν επιτραπεί στον χρήστη να κάνει άλλες ενέργειες. Αυτό γίνεται με το tMakeTransaction.join().

Μέσα στο thread καλείται η transferToContract της κλάσης Transaction. Αυτή στην ουσία καλεί την transferToContract της Solidity μέσω κλήσης JSON-RPC over HTTP.

Αφού ολοκληρωθεί η αίτηση του prover για τη συναλλαγή, μετά μπορεί είτε να ζητήσει από τους witnesses να αποδείξουν την τοποθεσία του ή να την ακυρώσει. Αυτό γίνεται πατώντας το κουμπί Confirm που καλεί την **confirmPoL**. Για κάθε witness, μέσα στη μέθοδο αυτή, γίνεται:

1. Ανταλλαγή κλειδιών μεταξύ Prover-Witness
2. Κρυπτογράφηση RSA της τοποθεσίας
3. Παραγωγή ψηφιακή υπογραφή του μηνύματος
4. Αποστολή μηνύματος

Ο κώδικας για την **ανταλλαγή κλειδιών** είναι ο ακόλουθος.

```
public void sendRequestPublicKey (View view) {
    Log.d(TAG, "Sending and requesting Public Key...");
   if (mBTDevice == null) return;
   Log.d(TAG, "Compatibility: " + mBluetoothConnection.isIncompatibleDevice + " for Device: " +
mBTDevice.getName());
    if (!mBluetoothConnection.isIncompatibleDevice) {
       String reqPOL = "My Public Key is: " + myPublicKey;
       byte[] bytes = reqPOL.getBytes(Charset.defaultCharset());
       mBluetoothConnection.write(bytes);
      new Thread( new Runnable() {
          @Override
          public void run() {
             witnessKey = getCurrentMessage();
             System.out.println(witnessKey);
             String temp = "" + reqPOL + "\nHis Public key is: " + witnessKey.substring(0, 10);
             Log.d(TAG, temp);
```

```
            }
        }).start();
    } else {
        Log.d(TAG, "Device " + mBTDevice + " is incompatible... Can't write here!");
    }
}
```

Στη μέθοδο **sendRequestPublicKey** παράγεται το μήνυμα που περιέχει ένα identifier “My public key is: ” και το δημόσιο κλειδί του λογαριασμού (που ταυτίζεται με τη διεύθυνση του στο Ethereum). Ύστερα το μήνυμα στέλνεται στο witness και ο prover περιμένει για απάντηση με την **getCurrentMessage**. Ο witness δέχεται το μήνυμα μέσω της κλάσης BluetoothConnectionService και απαντά αναλόγως σύμφωνα με τον παρακάτω κώδικα.

```
public void respondPublicKey (String message){
    Log.d(TAG, "Responding to Public Key request...");
    proverPublicKey = message.replace("My Public Key is: ", "");
   currentKeyPair = getKeyPair();
    PublicKey publicKey = currentKeyPair.getPublic();
    PrivateKey privateKeyTemp = currentKeyPair.getPrivate();
    byte[] privateKeyBytes = privateKeyTemp.getEncoded();
    String privateKeyBytesBase64 = new String(Base64.encode(privateKeyBytes,
Base64.DEFAULT));
    privateKey = privateKeyBytesBase64;
    byte[] publicKeyBytes = publicKey.getEncoded();
    String publicKeyBytesBase64 = new String( Base64.encode(publicKeyBytes,
Base64.DEFAULT));
    String response = "" + publicKeyBytesBase64;
    byte [] bytes2 = response.getBytes(Charset.defaultCharset());
    write(bytes2);
}

public KeyPair getKeyPair() {
    KeyPair kp = null;
    try {
        KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");
        kpg.initialize(2048);
        kp = kpg.generateKeyPair();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return kp;
}
```

Στην **respondPublicKey** ο witness αποθηκεύει το δημόσιο κλειδί του prover, και ύστερα

παράγει ένα ζευγάρι δημοσίου και ιδιωτικού κλειδιού RSA καλώντας την **getKeyPair**. Τέλος, ετοιμάζει το μήνυμα της απάντησης που αποτελείται από το δημόσιο κλειδί RSA. Το μήνυμα αυτό το υποδέχεται ο prover με την **getCurrentMessage** στη μέθοδο **requestPublicKey** της κλάσης MainPage που είδαμε πιο πριν.

Ύστερα στην **confimPoL** καλείται η συνάρτηση **sendLocation** στην οποία γίνεται η κρυπτογράφηση του μηνύματος μέσω της **encryptRSAToString**, η παραγωγή ψηφιακής υπογραφής και η αποστολή στο witness.

Το κομμάτι του κώδικα της **encryptRSAToString** φαίνεται παρακάτω

```
static String encryptRSAToString(String clearText, String publicKey) {
   String encryptedBase64 = "";
   try {
      KeyFactory keyFac = KeyFactory.getInstance("RSA");
      KeySpec keySpec = new X509EncodedKeySpec(Base64.decode(publicKey.trim().getBytes(),
Base64.DEFAULT));
      Key key = keyFac.generatePublic(keySpec);

      // get an RSA cipher object and print the provider
      final Cipher cipher = Cipher.getInstance("RSA/ECB/
OAEPWITHSHA-256ANDMGF1PADDING");
      // encrypt the plain text using the public key
      cipher.init(Cipher.ENCRYPT_MODE, key);

      byte[] encryptedBytes = cipher.doFinal(clearText.getBytes("UTF-8"));
      encryptedBase64 = new String(Base64.encode(encryptedBytes, Base64.DEFAULT));
   } catch (Exception e) {
      e.printStackTrace();
   }

   return encryptedBase64.replaceAll("(\\r|\\n)", "");
}
```

Η μέθοδος **encryptRSAToString** παίρνει το String publicKey το μετατρέπει σε τύπο Key και με τη βοήθεια ενός Cipher "RSA/ECB/OAEPWITHSHA-256ANDMGF1PADDING" παράγει και επιστρέφει το κρυπτογραφημένο μήνυμα.

Ύστερα πρέπει να παραχθεί η ψηφιακή υπογραφή του μηνύματος με τη χρήση του ιδιωτικού κλειδιού του λογαριασμού στο Ethereum. Ο κώδικας αυτός βρίσκεται στη **sendLocation** και φαίνεται παρακάτω.

```
Log.d(TAG, "Signing Message...");
byte[] bytes = encrypted.getBytes(Charset.defaultCharset());
```

```
Sign.SignatureData signature = Sign.signMessage(bytes, credentials.getEcKeyPair(), true);
byte [] v = new byte[] {signature.getV()};
byte [] r = signature.getR();
byte [] s = signature.getS();
byte[] bytes2 = new byte[r.length + s.length + v.length + encrypted.getBytes().length];

System.arraycopy(encrypted.getBytes(), 0, bytes2, 0, encrypted.getBytes().length);
System.arraycopy(r, 0, bytes2, encrypted.getBytes().length, r.length);
System.arraycopy(s, 0, bytes2, encrypted.getBytes().length + r.length, s.length);
System.arraycopy(v, 0, bytes2, encrypted.getBytes().length + r.length + s.length, v.length);
Log.d(TAG, "Finished signing...");
```

Εδώ παίρνουμε το κρυπτογραφημένο κείμενο (String encrypted), το μετατρέπουμε σε τύπο byte[] και δημιουργούμε την υπογραφή τύπου Sign.SignatureData, καλώντας τη Sign.signMessage της web3j, με παραμέτρους το μήνυμα, το KeyPair (δημόσιο και ιδιωτικό κλειδί) και με SHA3 hashing ενεργοποιημένο. Το hashing είναι απαραίτητο και για το verification από το Ethereum. Όπως έχουμε πει, η Sign.signMessage χρησιμοποιεί ECDSA για την παραγωγή της υπογραφής.

Αφού παραχθεί η υπογραφή, η μέθοδος την ετοιμάζει για να τη στείλει στο witness. Συγκεκριμένα η υπογραφή χωρίζεται στις μεταβλητές τύπου Byte r, s και v. Τα r και s αποτελούν το output μιας ECDSA υπογραφής, ενώ το v είναι το recovery ID. Έχοντας τα r, s και το μήνυμα, κάποιος μπορεί να επαληθεύσει στο Ethereum ποιος έστειλε το μήνυμα. Το v βοηθάει ώστε να επιτευχθεί γρηγορότερα αυτή η διαδικασία.

Τώρα παράγεται το μήνυμα που θα σταλεί στο witness περιλαμβάνει το κρυπτογραφημένο με RSA μήνυμα και την υπογραφή στη μορφή των r, s και v, αφού προστεθεί στην αρχή το identifier “PoL Request: ” ώστε να αναγνωρίσει ο witness το σκοπό του μηνύματος.

Όταν ο witness δεχτεί το μήνυμα τρέχει τον παρακάτω κώδικα.

```
public void witnessCheck (byte [] bytes, int size){
   // Verify
   byte [] r, s, v, encrypted;
   r = new byte[32];
   s = new byte[32];
   v = new byte[1];
   encrypted = new byte[size-65];

   System.arraycopy(bytes, 0, encrypted, 0, size-65);
   System.arraycopy(bytes, size-65, r, 0, 32);
   System.arraycopy(bytes, size-33, s, 0, 32);
```

```
        System.arraycopy(bytes, size-1, v, 0, 1);
        String encryptedMessage = new String(encrypted, 0, encrypted.length);
        byte vByte = v[0];

        // Prepare Signature
        String signerAddress;
        Sign.SignatureData signature = new Sign.SignatureData(vByte, r, s);
        signerAddress = verifySignature(encryptedMessage.getBytes(Charset.defaultCharset()),
    signature);

        // Check if signature is OK
        if (signerAddress.equals("null")) {
            Log.e(TAG, "Signature does not match");
            return;
        }

        // Check if coords are OK
        if (!checkCoords(encryptedMessage)) {
            Log.e(TAG, "Coordinates do not match");
            return;
        }

        TransactionReceipt transactionReceipt = null;
        try {
            transactionReceipt = transaction.witnessProof(signerAddress).send();
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (transactionReceipt != null) {
            Log.d(TAG, "PoL approved... Transaction confirmed successfully");
            Log.d(TAG, "Gas used: " + transactionReceipt.getGasUsed());
        }
    }
```

Στη μέθοδο **witnessCheck** αρχικά αποκρυπτογραφείται το μήνυμα που δέχτηκε ο witness. Τώρα έχουμε τα r, s, v και encrypted. Τα τρία πρώτα θα χρησιμοποιηθούν για να δημιουργηθεί μια νέα υπογραφή τύπου Sign.SignatureData, η οποία με τη verifySignature θα επιβεβαιωθεί ότι ο αποστολέας της είναι όντως αυτός που αντιστοιχεί στη διεύθυνση του δημοσίου κλειδιού που δέχτηκε πιο πριν ο witness. Στην ουσία μέσα στη verifySignature καλείται η **Sign.signedMessageToKey**, η οποία επιστρέφει τη διεύθυνση του λογαριασμού στο Ethereum (το οποίο όπως είπαμε ταυτίζεται με το δημόσιο κλειδί).

Αν το κλειδί που δέχτηκε ο witness στην αρχική ανταλλαγή κλειδιών δεν ταυτίζεται με τη διεύθυνση του λογαριασμού που παράξαμε τώρα, τότε σημαίνει ότι ο prover δεν ήταν

ειλικρινής, δηλαδή προσπαθεί να πείσει τους witnesses ότι είναι κάποιος άλλος και ενδεχομένως προσπαθεί να παρέμβει στην απόδειξη τοποθεσίας κάποιου άλλου λογαριασμού. Τότε η εφαρμογή δεν καλεί την **WitnessProof** και συνεπώς δεν στέλνει PoL στο Smart Contract.

Αν η ταυτότητα του prover ταυτοποιηθεί επιτυχώς, τότε γίνεται ο έλεγχος των συντεταγμένων με την checkCoords, η οποία πρέπει να επιστρέψει true. Ο κώδικας της φαίνεται παρακάτω.

```
public Boolean checkCoords (String encryptedMessage) {
    String encryptedMessageOnly = encryptedMessage.replace("PoL Request: ", "");

    String coordinates = decryptRSAToString(encryptedMessageOnly, privateKey);
    Log.d(TAG, "Decrypted Message: " + coordinates);
    String resPoL = coordsToString( coordinates );

    (...)

    double difX, difY;
     difX = abs(myX - hisX);
     difY = abs(myY - hisY);

     if (difX <= 0.0001 && difY <= 0.0001) return true;
     return false;
}
```

Η μέθοδος **checkCoords** έχει ως σκοπό την επαλήθευση των συντεταγμένων που στέλνει ο prover. Αρχικά το μήνυμα που είχε κρυπτογραφηθεί με το δημόσιο κλειδί του witness από τον prover καλώντας την **encryptRSAToString**, τώρα αποκρυπτογραφείται καλώντας τη **decryptRSAToString**.

Ύστερα οι συντεταγμένες χωρίζονται κατάλληλα. Ο κώδικας αυτής της διαδικασίας δεν παρουσιάζεται παραπάνω.

Μετά υπολογίζεται η διαφορα των X και Y συντεταγμένων, που αν και για τα δύο είναι μικρότερη από 0.0001, τότε η μέθοδος επιστρέφει true, και αποστέλλεται μετά στο Ethereum το PoL. Διαφορετικά επιστρέφει false και δεν στέλνει PoL.

Τέλος, η **witnessCheck**, αν οι συντεταγμένες του prover είναι αποδεκτές, τότε ο witness καλεί τη witnessProof του smart contract και στην ουσία στέλνει το PoL του στο Ethereum. Όταν το smart contract δεχτεί PoL από όλους τους witnesses, τότε ολοκληρώνει αυτόματα

τη συναλλαγή, όπως έχουμε ήδη δει, στέλνοντας τα milliether στον προορισμό που είχε δηλώσει ο prover. Η συναλλαγή έχει πλέον ολοκληρωθεί.

Αντίστοιχα, ο prover θα μπορούσε αντί να κάνει confirm τη συναλλαγή πατώντας το αντίστοιχο κουμπί, να πατήσει το Cancel Transaction, που καλεί τη returnTransaction και επιστρέφει αυτόματα τα milliether πίσω στο λογαριασμό του και στην ουσία ακυρώνει τη συναλλαγή.

5.3 Παρατηρήσεις

Παρακάτω ακολουθούν ορισμένες παρατηρήσεις σχετικά με την εφαρμογή και την υλοποίησή της.

5.3.1 Συνθήκες δοκιμής εφαρμογής

Αρχικά τα κινητά που έχουν χρησιμοποιηθεί για τον έλεγχο της λειτουργίας της εφαρμογής είναι δύο MLS iQW553N, που τρέχουν Android 7.0 και ένα Samsung Galaxy J7 το οποίο τρέχει Android 8.1. Τα τρία κινητά έχουν δυνατότητα σύνδεσης με Bluetooth, ενώ η εφαρμογή εκτελέστηκε μόνο με ενεργή σύνδεση με το Διαδίκτυο. Δοκιμάστηκε η λειτουργία της εφαρμογής σε σενάριο συνεργασίας του prover και ενός witness, καθώς και του prover με δύο witnesses.

Το έξυπνο συμβόλαιο βρίσκεται στο Rinkeby testnet. Η επιλογή του Rinkeby έγινε φυσικά για αποφυγή του mainnet του Ethereum, το οποίο θα απαιτούσε σημαντική χρήση πραγματικού χρήματος. Από τα διαθέσιμα testnets (Rinkeby, Ropsten, κτλ) το Rinkeby επιλέχθηκε αυθαίρετα. Δεν παίζει σημαντικό ρόλο στο πλαίσιο αυτής της διπλωματικής εργασίας το testnet, επομένως κάλλιστα θα μπορούσε να είχε επιλεχθεί και το Ropsten ή κάποιο άλλο.

Κατά τη χρήση της εφαρμογής, εκτελέστηκε το πρωτόκολλο από τα κινητά, και ελεγχόταν από το etherscan (https://rinkeby.etherscan.io) αν γίνονται οι συναλλαγές. Οι δοκιμές έγιναν και εσωτερικό χώρο, ελέγχοντας έτσι την απόκτηση των συντεταγμένων από τον Network provider του Android, αλλά και σε εξωτερικό χώρο, όπου χρησιμοποιείται ο GPS Provider του Android λειτουργικού.

5.3.2 Gas Price

Η τιμή του gas που χρησιμοποιείται κατά τις διάφορες συναλλαγές και γενικότερα με τις αλληλεπιδράσεις με το smart contract, έχει τεθεί στα 22 gwei. Αυτή είναι μια ικανοποιητικά υψηλή τιμή, ώστε να γίνει mine γρήγορα η συναλλαγή, με μέσο χρόνο τα 15 δευτερόλεπτα ανά συναλλαγή. Στην περίπτωση που το συμβόλαιο και οι συναλλαγές με αυτό μεταφερθούν στο Ethereum mainnet, τότε πιθανά στο μέλλον, ανάλογα με τις συνθήκες, η τιμή αυτή να χρειαστεί να αλλάξει και συγκεκριμένα να αυξηθεί για να μην αγνοούνται οι συναλλαγές από τους miners και επομένως να μην καθυστερούν αισθητά.

Η αλλαγή της τιμής γίνεται αλλάζοντας τον παρακάτω κώδικα στο MainClass.java και στο BluetoothConnectionService.java.

```
BigInteger gasLimit = BigInteger.valueOf(6700000L);
BigInteger gasPrice = BigInteger.valueOf(22_000_000_000L);
```

5.3.3 Αντίτιμο Εγγραφής

Όπως έχουμε αναφέρει, για να εκτελέσει κάποια συναλλαγή μέσω του συμβολαίου κάποιος χρήστης πρέπει πρώτα να έχει εγγραφεί. Για να επιτευχθεί αυτό, πρέπει ο χρήστης να έχει μεταφέρει από το λογαριασμό στο smart contract το ποσό των 0.02 ether. Τη στιγμή της συγγραφής αυτής της διπλωματικής εργασίας, το ποσό αυτό αναλογεί περίπου στα 5$.

Το αντίτιμο αυτό είναι απαραίτητο για να έχει αξία η επιβολή του μηχανισμού αποφυγής συνεργασίας για την αντιμετώπιση κακόβουλων χρηστών. Αν δεν υπήρχε αυτό το αντίτιμο κατά την εγγραφή, τότε κακόβουλοι provers θα μπορούσαν πολύ εύκολα να φτιάχνουν καινούριους λογαριασμούς στο Ethereum και να τους χρησιμοποιούν ως witnesses για να αντιμετωπίσουν το μηχανισμό που μπλοκάρει τους κακόβουλους witnesses από το λογαριασμό του prover. Η ύπαρξη του αντίτιμου δυσκολεύει αισθητά το έργο αυτό καθώς το κόστος γίνεται όλο και μεγαλύτερο με κάθε προσπάθεια απάτης.

Το ποσό του 0.02 και επομένως των 5$ έχει επιλεγεί αυθαίρετα. Ανάλογα με τις ανάγκες της εφαρμογής, καθώς και ανάλογα με τη συμπεριφορά των χρηστών, θα μπορούσε να χρησιμοποιηθεί μεγαλύτερο ποσό για την εγγραφή.

5.3.4 Ακρίβεια συντεταγμένων

Ένας σημαντικός παράγοντας της εφαρμογής είναι η ακρίβεια των συντεταγμένων του χρήστη. Μεγάλη ακρίβεια μπορεί να είναι επιθυμητή ανάλογα με την εφαρμογή, ενώ

μικρή ακρίβεια προστατεύει την ακριβή γεωγραφική θέση του χρήστη, που αποτελεί ευαίσθητο δεδομένο που ιδανικά δεν θέλουμε να είναι γνωστή από τους διάφορους witnesses.

Στο πλαίσιο της εφαρμογής που δημιουργήσαμε, επιθυμούμε ακρίβεια τεσσάρων δεκαδικού στη σύγκριση συντεταγμένων. Αυτό σημαίνει ότι οι συντεταγμένες (0, 0) και (0.0001, 0.0001) είναι στο αποδεκτό όριο, αφού αυτό σημαίνει ότι έχουν απόσταση περίπου 11.12 μέτρα στον κάθε άξονα. Συνεπώς, η απόσταση τους θα ήταν περίπου 15.73 μέτρα. Αν λάβουμε υπόψη και το σφάλμα της ακρίβειας 4 δεκαδικών που έχουμε ορίσει το GPS να κρατάει, τότε στη χειρότερη θα έχουμε απόσταση 3*15.73 = 47.19 μέτρα, η οποία είναι μία αποδεκτή ακρίβεια.

Δεκαδικά	**Ακρίβεια απόστασης**
1.0	111.32 km
0.1	11.132 km
0.01	1.1132 km
0.0001	111.32 m
0.00001	11.132 m
0.000001	1.1132 m
0.0000001	111.32 mm

Σχήμα 5.2: Πίνακας ακρίβειας συντεταγμένων

Αν θέλουμε να προστατέψουμε παραπάνω την τοποθεσία του prover, τότε μπορούμε να μειώσουμε τα δεκαδικά που στέλνονται στο witness. Παρόλα αυτά η χρήση τεχνολογίας μικρής εμβέλειας όπως είναι το Bluetooth προδίδει τη γενική τοποθεσία του prover, με αποτέλεσμα η μεγάλη μείωση των δεκαδικών να μην προσφέρει σημαντική αύξηση της προστασίας της τοποθεσίας του χρήστη από τους witnesses. Αντίστοιχα αν έχουμε ανάγκη μεγαλύτερη ακρίβεια στην τοποθεσία του prover, τότε μπορούμε να αυξήσουμε των αριθμό των δεκαδικών στις συντεταγμένες.

5.4 Εγκατάσταση και χρήση

Παρακάτω ακολουθούν οι οδηγίες εγκατάστασης της εφαρμογής σε Android κινητά, καθώς και ο τρόπος λειτουργίας της εφαρμογής σε αυτά. Αξίζει να αναφερθεί ότι για τη σωστή λειτουργία της εφαρμογής απαιτούνται τουλάχιστον 2 κινητά με την εφαρμογή εγκατεστημένη.

5.4.1 Εγκατάσταση Εφαρμογής

Για την εγκατάσταση της εφαρμογής απαιτείται API 24 και πάνω, που αντιστοιχεί σε Android Nougat και πάνω. Τα βήματα της εγκατάστασης φαίνονται παρακάτω.

1. Κατέβασμα του κώδικα από το https://github.com/tsikos7/PoL-Android-Application

2. Άνοιγμα του project από το Android Studio. Τώρα απαιτούνται μερικές αλλαγές στο κώδικα και συγκεκριμένα η εισαγωγή της διεύθυνσης λογαριασμού του prover και το ιδιωτικό του κλειδί.

```
String myPrivateKey = "YOUR_PRIVATE_KEY";
Credentials credentials = Credentials.create("YOUR_PRIVATE_KEY");
String myPublicKey = "YOUR_ACCOUNT_ADDRESS";
```

Στο MainPage.java και στο BluetoothConnectionService.java πρέπει να γίνει αντικατάσταση των προεπιλεγμένων κλειδιών με αυτά του λογαριασμού Ethereum που θέλουμε να εισάγουμε στην εφαρμογή. Αυτό θα πρέπει να γίνει και στο strings.xml, όπου αν επιθυμούμε μπορούμε να προσθέσουμε και άλλους λογαριασμούς, που να μπορεί να τους αλλάζει ο χρήστης.

3. Στην περίπτωση που θέλουμε να κάνουμε deploy το συμβόλαιο κάπου εκτός του rinkeby testnet, όπως στο Ethereum mainnet, τότε πρέπει ο κώδικας του συμβολαίου να αντιγραφεί και να γίνει deploy. Αυτό μπορεί να γίνει γρήγορα από το https://remix.ethereum.org/. Ύστερα, η διεύθυνση του συμβολαίου πρέπει να αντιγραφεί και να τοποθετηθεί στη μεταβλητή contractAddress που βρίσκεται στο MainPage.java και στο BluetoothConnectionService.java.

```
// BLOCKCHAIN CONTRACT
String contractAddress = "YOUR_CONTRACT_ADDRESS";
String url = "https://rinkeby.infura.io/v3/671362fca54b42b0a7c7f3c3126dc47b";
Web3j web3j = Web3j.build(new InfuraHttpService(url));
```

Σε αυτήν την περίπτωση θα πρέπει να γίνει αλλαγή και της διεύθυνσης στο Infura, από rinkeby.infura.io/v3/YOUR_ID σε mainnet.infura.io/v3/YOUR_ID, αντικαθιστώντας το ID με αυτό του λογαριασμού Infura που θα πρέπει να έχει δημιουργηθεί. Φυσικά, το προεπιλεγμένο 671362fca54b42b0a7c7f3c3126dc47b θα λειτουργεί.

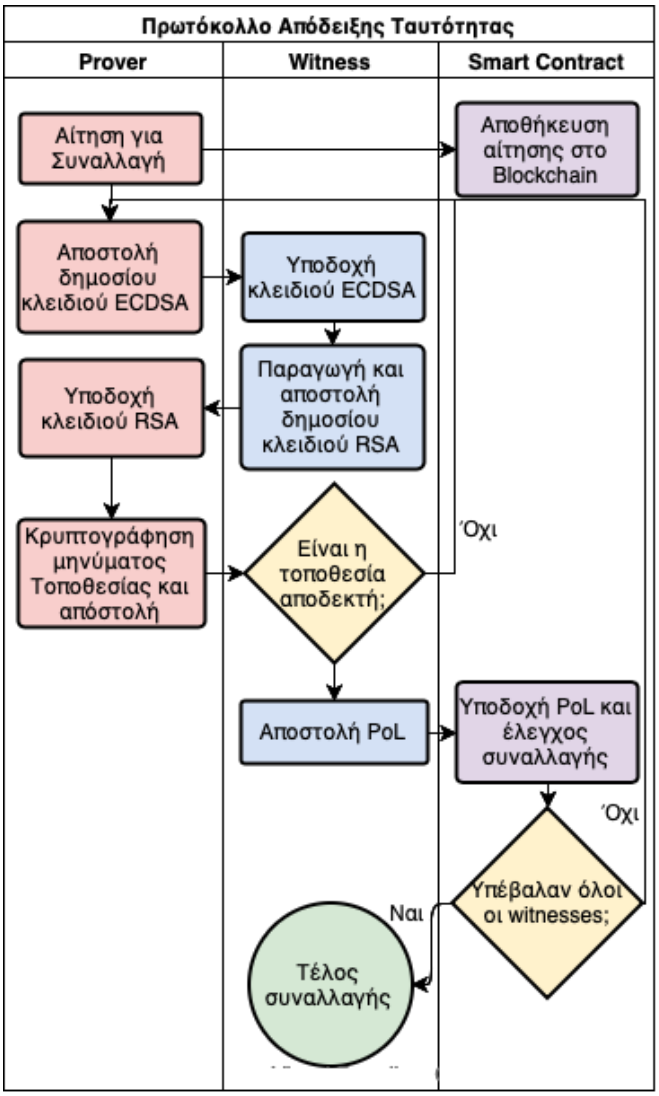
4. Τώρα αρκεί να γίνει το Build και Run για να τρέξει η εφαρμογή στο συνδεδεμένο κινητό. ΣΗΜΑΝΤΙΚΟ: Η εφαρμογή δε θα λειτουργεί σωστά στο emulator του Android Studio, διότι δεν έχει δυνατότητες Bluetooth.

5.4.2 Σενάριο Χρήσης

Ένα πλήρες σενάριο χρήσης της εφαρμογής φαίνεται στο παρακάτω swimlane diagram. Για τη σωστή λειτουργία της εφαρμογής και του πρωτοκόλλου έχουμε κάνει τις εξής υποθέσεις:

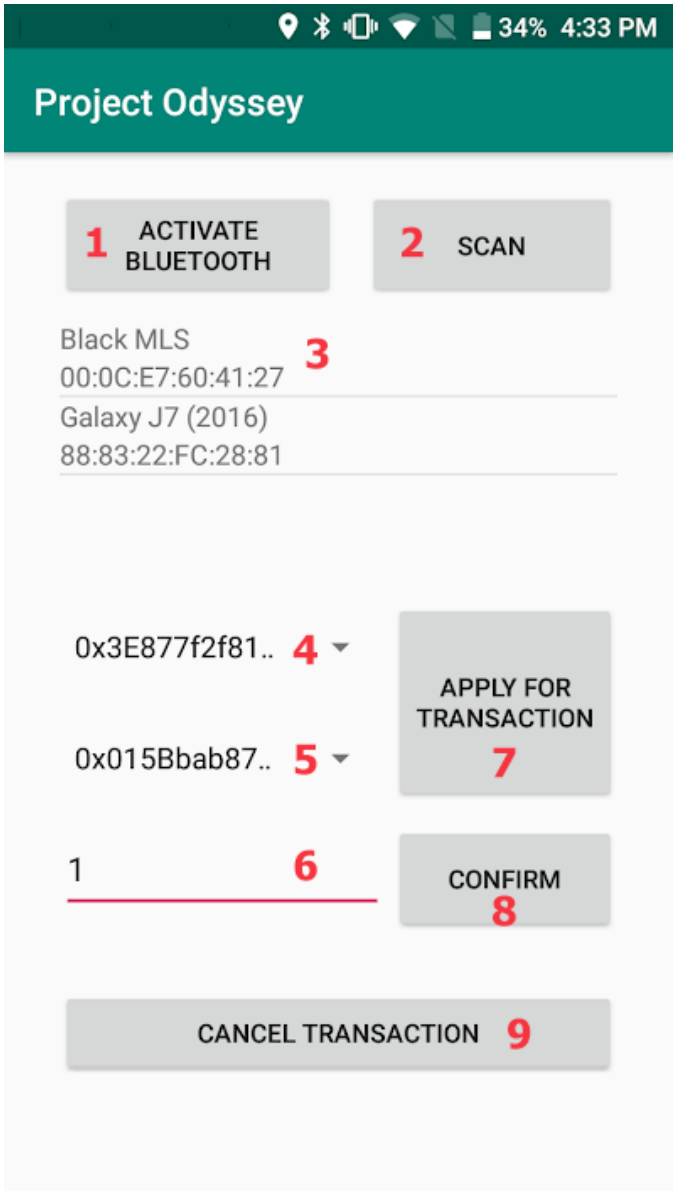
1. Η συσκευή έχει λογισμικό Android με API 24 ή μεγαλύτερο
2. Η συσκευή έχει πρόσβαση στο Internet
3. Η συσκευή έχει δυνατότητες Bluetooth
4. Η συσκευή βρίσκεται κοντά σε 2 ή περισσότερες άλλες συσκευές που είναι paired και έχουν εγκατεστημένη την εφαρμογή

Παρακάτω φαίνεται το swimlanes diagram της πλήρους λειτουργίας της εφαρμογής:



Σχήμα 5.3: Swimlanes diagram πλήρης διαδικασίας χρήσης εφαρμογής

Παρακάτω περιγράφεται το σενάριο χρήσης.



Σχήμα 5.4: Κεντρική οθόνη Dapp με αρίθμηση στοιχείων

1. Αρχικά ο χρήστης, έχοντας κλειστό το Bluetooth ανοίγει την εφαρμογή

2. Στο άνοιγμα η συσκευή θα ανοίξει παράθυρο για να ενεργοποιήσει το Bluetooth ο χρήστης. Πατώντας Allow και μετά ξανά Allow για να γίνει discoverable, η συσκευή είναι έτοιμη να εκτελέσει το πρωτόκολλο. Εναλλακτικά ο χρήστης μπορεί να πατήσει το κουμπί Activate Bluetooth (1) για να ενεργοποιήσει το Bluetooth.

3. Ύστερα πατώντας το κουμπί Scan (2) εμφανίζονται όλες οι paired κινητό συσκευές. Αξίζει να αναφερθεί ότι **η διαδικασία των βημάτων 2 και 3 είναι αυτοματοποιημένη στο άνοιγμα της εφαρμογής** και δεν χρειάζεται το πάτημα κουμπιών από τον χρήστη.

4. Ο χρήστης επιλέγει τις συσκευές που θα χρησιμοποιηθούν ως witness από τη λίστα που εμφανίζεται στο ListView (3). Το χρώμα των επιλεγμένων συσκευών θα αλλάξει σε πράσινο.

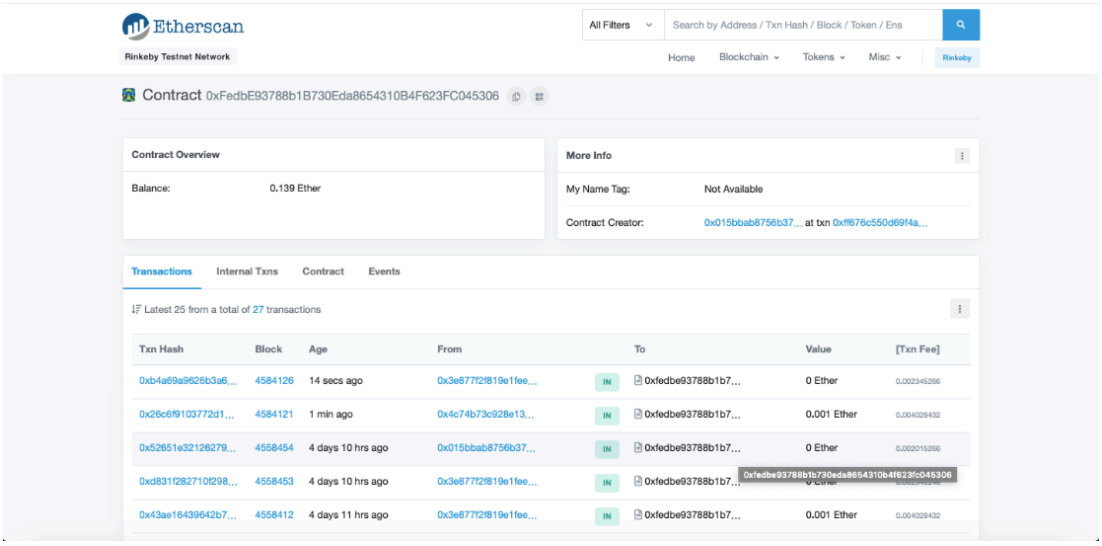
5. Ύστερα θα επιλέξει από το dropdown (4) το λογαριασμό από τον οποίο επιθυμεί να κάνει τη μεταφορά. Οι λογαριασμοί που εμφανίζονται πρέπει να έχουν τοποθετηθεί κατά την εγκατάσταση.

6. Μετά θα πληκτρολογήσει τη διεύθυνση του προορισμού της συναλλαγής, καθώς και το ποσό της συναλλαγής σε milliether στα (5) και (6) αντίστοιχα.

7. Τώρα μπορεί να κάνει "αίτηση" για συναλλαγή στο smart contract πατώντας το κουμπί Apply for Transaction (7). Θα εμφανιστεί ένα παράθυρο αναμονής του χρήστη. Ανάλογα με την ισχύ της σύνδεσης στο διαδίκτυο, το παράθυρο θα παραμείνει από 7 μέχρι 30 δευτερόλεπτα.

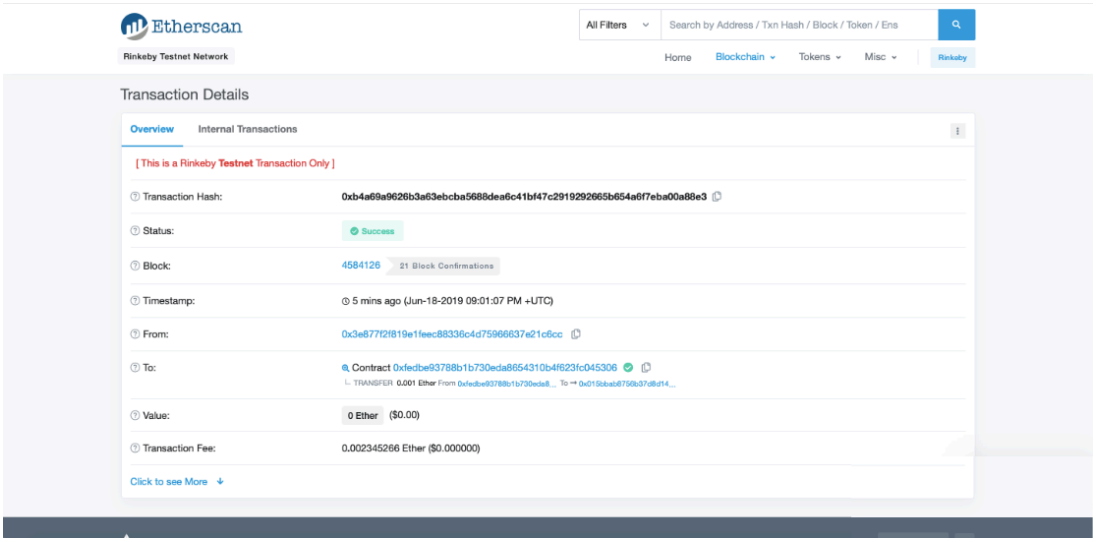
8. Τέλος, ο χρήστης μπορεί είτε να ολοκληρώσει τη συναλλαγή πατώντας το κουμπί Confirm (8) και στέλνοντας έτσι στους witnesses που έχει επιλέξει την τοποθεσία του, οι οποίοι θα επαληθεύσουν τη συναλλαγή, είτε να ακυρώσει τη συναλλαγή πατώντας το κουμπί Cancel Transaction (9) και να πάρει πίσω τα milliether που μετέφερε στο συμβόλαιο στο βήμα 7.

Όλη η παραπάνω διαδικασία δημιουργεί n+1 συναλλαγές στο έξυπνο συμβόλαιο, όπου n είναι ο αριθμός των witnesses. Οι συναλλαγές αυτές μπορούν να παρακολουθηθούν από το https://rinkeby.etherscan.io και συγκεκριμένα από τη σελίδα του έξυπνου συμβολαίου. Παρακάτω φαίνονται οι συναλλαγές που προέκυψαν κατά τη μεταφορά ενός milliether με τη χρήση ενός witness.



Σχήμα 5.5: Smart Contract στο Rinkeby Etherscan

Παραπάνω φαίνεται στη δεύτερη συναλλαγή, η αίτηση συναλλαγής από τον prover. Συγκεκριμένα, πρόκειται για τη συναλλαγή που γίνεται η μεταφορά του 0.001 ether στο έξυπνο συμβόλαιο.



Σχήμα 5.6: Τελική Συναλλαγή τελευταίου Witness

Παραπάνω φαίνεται η δεύτερη συναλλαγή της διαδικασίας, δηλαδή η αποστολή Απόδειξης Τοποθεσίας στο Ethereum από τον witness. Επειδή αποτελεί τον τελευταίο witness, φαίνεται η μεταφορά του 0.001 ether από το συμβόλαιο στον προορισμό. Αυτή η συναλλαγή είναι εσωτερική, δηλαδή πραγματοποιείται μόνο μέσα από το συμβόλαιο.

5.5 Συμπεράσματα

Παρακάτω ακολουθούν τα συμπεράσματα που προέκυψαν από τη λειτουργία του πρωτοκόλλου Απόδειξης Ταυτότητας, καθώς και από τη λειτουργία της Android εφαρμογής στο Blockchain.

- Όσον αφορά την εφαρμογή στα Android, δουλεύει όπως σχεδιάστηκε, χωρίς να παρουσιάζεται κανένα πρόβλημα κατά την λειτουργία

- Η διαδικασία μιας συναλλαγής, μαζί με την Απόδειξη Τοποθεσίας, παίρνει κάποιο χρόνο. Συγκεκριμένα διαρκεί από 15 δευτερόλεπτα εώς και 1 λεπτό. Ο χρόνος αυτός οφείλεται στην εκτέλεση των συναλλαγών στο Ethereum, δηλαδή κάθε φορά που γίνεται επικοινωνία με το έξυπνο συμβόλαιο, καθώς και όταν γίνεται επικοινωνία με άλλες συσκευές. Ο χρόνος επικοινωνίας εξαρτάται από το πόσοι witnesses είναι διαθέσιμοι, αλλά είναι αμελητέος. Αντίθετα ο χρόνος της κλήσης

μεθόδων στο έξυπνο συμβόλαιο μπορεί να πάρει σημαντικό χρόνο, ο οποίος εξαρτάται από το πόσο καλή είναι η σύνδεση στο διαδίκτυο.

- Ο λόγος που ο χρόνος της σύνδεσης και επικοινωνίας με άλλες συσκευές είναι μικρός, είναι επειδή απαιτούμε οι συσκευές να είναι paired και να επιλέγονται από το χρήστη με κριτήριο να είναι εγκατεστημένη η εφαρμογή στις συσκευές των witnesses. Οι απαιτήσεις αυτές προκύπτουν από την ανάγκη δυο συσκευές να είναι paired για να συνδεθούν με Bluetooth, η οποία προκύπτει από το λειτουργικό Android. Είναι αδύνατο κάποια συσκευή να συνδεθεί με κάποια άλλη χωρίς να είναι paired.

- Η αποκλειστική χρήση του μηχανισμού αποφυγής συγκρούσεων στην ουσία επιτρέπει σε κακόβουλους χρήστες να συνεργαστούν και να πουν ψέματα, αλλά τους απαγορεύει να το κάνουν πολλές φορές. Αυτό σημαίνει οτι καμία ομάδα χρηστών δεν μπορεί να εκμεταλλεύεται το πρωτόκολλο, χωρίς το σημαντικό κόστος (0.02 ether) που προκύπτει από την ανάγκη δημιουργίας πολλών νέων λογαριασμών. Παρόλα αυτά, ο μηχανισμός δεν αντιμετωπίζει τα διάφορα θέματα ασφαλείας, γεγονός που απαιτεί την ύπαρξη κρυπτογραφίας για την ασφαλή επικοινωνία των συσκευών.

- Το πρωτόκολλο που αναπτύχθηκε και συγκεκριμένα το έξυπνο συμβόλαιο και η εφαρμογή στα Android θα μπορούσαν εύκολα να δουλέψουν στο mainnet του Ethereum και να χρησιμοποιηθούν από χρήστες, μεταφέροντας πραγματικά χρήματα. Δηλαδή το πρωτόκολλο δεν είναι καθαρά ερευνητικό, αλλά είναι άμεσα διαθέσιμο να χρησιμοποιηθεί σε πραγματικές καταστάσεις.

6. Μελλοντική Εργασία

Στην παρούσα εργασία αναπτύχθηκε και υλοποιήθηκε ένα πρωτόκολλο Απόδειξης τοποθεσίας με χρήση συσκευών witnesses, με εφαρμογή στο Ethereum blockchain. Για την επικοινωνία των συσκευών χρησιμοποιήθηκε η κοντινής εμβέλειας τεχνολογία Bluetooth. Τόσο το πρωτόκολλο, όσο και η εφαρμογή παρουσίασαν κάποια μικρά προβλήματα που απαιτούν επίλυση, ενώ υπάρχει περιθώριο τόσο για βελτίωση κάποιων λειτουργιών, αλλά και για επέκταση της εργασίας.

6.1 Επίλυση Προβλημάτων

- Η εφαρμογή απαιτεί από τον prover να είναι paired με τον χρήστη με τον οποίο επικοινωνεί. Αυτό σημαίνει ότι θα πρέπει να γνωρίζει τον witness και να τον εμπιστεύεται για να συνδεθεί μαζί του με Bluetooth. Αυτή η διαδικασία απαιτεί αλληλεπίδραση μεταξύ των χρηστών των συσκευών η οποία δεν είναι επιθυμητή. Ενδεχομένως, κάτι τέτοιο να μπορούσε να αντιμετωπιστεί με τη χρήση κάποιας άλλης τεχνολογίας για την επικοινωνία, όπως wifi.

- Η εφαρμογή απαιτεί από τους witnesses να έχουν ανοιχτή την εφαρμογή στα κινητά τους ώστε να μπορέσει ο prover να επικοινωνήσει μαζί τους. Δεν έχουν όμως κανένα κίνητρο για να λειτουργήσουν ως witnesses και να του παρέχουν Απόδειξη Τοποθεσίας. Αυτό θα μπορούσε να γίνει με την επιβολή ενός μικρού αντιτίμου στον prover, το οποίο θα μοιραστεί στους witnesses.

- Επειδή τα κινητά δεν έχουν την επεξεργαστική ικανότητα και τη μνήμη να τρέξουν ένα ολόκληρο κόμβο του Ethereum blockchain, απαιτείται η επικοινωνία με κάποιον κόμβο του Infura. Αυτό πάει κόντρα στην αποκεντροποίηση που υπόσχεται το Ethereum blockchain, καθώς όλες οι συναλλαγές περνάνε από το Infura, που στην περίπτωσή μας αποτελεί μια κεντρική οντότητα. Με τη συνεχή ανάπτυξη της τεχνολογίας των κινητών τύπου smartphone, είναι λογικό να υποθέσουμε ότι στο μέλλον θα μπορούν τα smartphones να τρέχουν κόμβο Ethereum και δεν θα απαιτείται το Infura πλέον από την εφαρμογή.

6.2 Βελτιώσεις

- Το πρωτόκολλο επαληθεύει την τοποθεσία του prover χρησιμοποιώντας τους witnesses, όμως δεν επαληθεύει ποτέ την τοποθεσία των witnesses. Επομένως, στην παρούσα φάση, απαιτεί έναν μεγάλο αριθμό witnesses για να είναι σημαντικά

έγκυρη η αναφορά της τοποθεσίας απο τον prover. Η εγκυρότητα του πρωτοκόλλου θα μπορούσε να ενισχυθεί σημαντικά με την επανάληψη του από τους witnesses για την Απόδειξη Τοποθεσίας τους με νέους witnesses και η διαδικασία αυτή να συνεχιστεί σε πολλά στρώματα. Δηλαδή το πρωτόκολλο θα ενισχυόταν με τη χρήση της επικοινωνίας ενός δικτύου peer-to-peer από κοντινούς witnesses.

- Η εφαρμογή χρησιμοποιεί τον μηχανισμό αποφυγής συγκρούσεων σε σημαντικό βαθμό για την αποφυγή επιθέσεων MITM. Παρόλα αυτά, αν ο κακόβουλος χρήστης καταφέρει να μπει ανάμεσα στον prover και τον witness, υποκλέπτοντας τα κλειδιά, τότε θα μπορεί να εκτελέσει μια επιτυχή επίθεση. Ο μηχανισμός θα τον αποτρέψει από το να τον ξανακάνει. Παρόλα αυτά, η εφαρμογή θα μπορούσε να επωφεληθεί σημαντικά από περισσότερα μέτρα ασφαλείας από MITM επιθέσεις, για την αποφυγή αυτών των μεμονωμένων περιστατικών που μπορούν να δημιουργηθούν πριν την ενεργοποίηση του μηχανισμού.

6.3 Επεκτάσεις

Η εφαρμογή του πρωτοκόλλου έχει υλοποιηθεί στο πλαίσιο μιας εφαρμογής συναλλαγών στο Ethereum blockchain το οποίο είναι public. Παρόλα αυτά θα μπορούσε να χρησιμοποιηθεί και σε κάποιο permissionless blockchain που δεν θα προσέφερε δυνατότητα συναλλαγών αλλά κάποια άλλη λειτουργία. Θα μπορούσε αυτό το blockchain για παράδειγμα να χρησιμοποιηθεί ως βάση δεδομένων για κάποιο σύστημα supply chain, που θα μπορεί να γνωρίζει οποιαδήποτε στιγμή που βρίσκονται τα φορτηγά που μεταφέρουν τα προϊόντα.

Επιπρόσθετα, με μερικές αλλαγές, το πρωτόκολλο θα μπορούσε να χρησιμοποιηθεί σε οποιαδήποτε άλλη εφαρμογή που θα απαιτούσε επιβεβαίωση τοποθεσίας κάποιου χρήστη. Για να γίνει αυτό, πιθανότατα η εφαρμογή θα χρειαζόταν κάποια βάση δεδομένων εκτός του blockchain για την αποθήκευση πληροφοριών.

7. Βιβλιογραφία

[1] Haber, Stuart; Stornetta, W. Scott (January 1991). "How to time-stamp a digital document". Journal of Cryptology. Available: https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.8740

[2] «What is the Web3? The Decentralized Web - Blockchain» [Ηλεκτρονικό]. Available: https://blockchainhub.net/web3-decentralized-web/.

[3] «Elliptic Curve Cryptography (ECC),» [Ηλεκτρονικό]. Available: https://www.certicom.com/content/certicom/en/ecc.html.

[4] J. Ray, «Ethereum introduction,» [Ηλεκτρονικό]. Available: https://github.com/ethereum/wiki/wiki/Ethereum-introduction.

[5] «What is Ethereum?,» [Ηλεκτρονικό]. Available: http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html.

[6] S. Saroiu και A. Wolman, «Saroiu, Stefan, and Alec Wolman. "Enabling new mobile applications with location proofs.,» σε Proceedings of the 10th workshop on Mobile Computing Systems and Applications. , 2009.

[7] W. Luo και U. Hengartner, «VeriPlace: A Privacy-Aware Location Proof Architecture,» σε Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (pp. 23-32), 2010.

[8] S. Gambs, M.-O. Killijian, M. Roy και M. Traoré, «PROPS: A PRivacy-Preserving Location Proof System,» σε 2014 IEEE 33rd International Symposium on Reliable Distributed Systems, 2014.

[9] W. Luo και U. Hengartner, «Proving your location without giving up your privacy.,» σε Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications, 2010.

[10] Z. Zhu και G. Cao, «Applaus: A privacy-preserving location proof updating system for location-based services.,» σε 2011 Proceedings IEEE INFOCOM, 2011.

[11] B. Davis, H. Chen και M. Franklin, «Privacy-Preserving Alibi Systems,» σε Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security., 2012.

[12] N. Sastry, U. Shankar και D. Wagner, «Secure verification of location claims.,» σε Proceedings of the 2nd ACM workshop on Wireless security., 2003.

[13] X. Wang, A. Pande, J. Zhu και P. & Mohapatra, «STAMP: enabling privacy- preserving location proofs for mobile users.,» IEEE/ACM transactions on networking, τόμ. 24, αρ. 6, pp. 3276-3289, 2016.

[14] R. Khan, S. Zawoad, M. M. Haque και R. Hasan, «OTIT: Towards Secure Provenance Modeling for Location Proofs,» σε Proceedings of the 9th ACM symposium on Information, computer and communications security., 2014.

[15] Y. Zheng and W. Lou, "Location based handshake and private proximity test with location tags.," IEEE Transactions on Dependable and Secure Computing 14.4, pp. 406-419, 2017.

[16] Foamspace Corp, «FOAM Whitepaper,» 5 January 2018. [Ηλεκτρονικό]. Available: https://foam.space/publicAssets/FOAM_Whitepaper.pdf. [Πρόσβαση 4 March 2019].

[17] Foamspace Corp, «FOAM Technical Whitepaper Draft 0.4,» 4 May 2018. [Ηλεκτρονικό]. Available: https://github.com/f-o-a-m/public- research/raw/master/ FOAM%20Techincal%20Whitepaper%20Draft.pdf. [Πρόσβαση 4 March 2019].

[18] Y. Zhang, C. C. Tan, F. Xu, H. Han και Q. Li, «Vproof: Lightweight privacy-preserving vehicle location proofs,» IEEE Transactions on Vehicular Technology, τόμ. 64, αρ. 1, pp. 378-385, 2015.

[19] R. Khan, S. Zawoad, M. M. Haque και R. Hasan, «WORAL: A Witness Oriented Secure Location Provenance Framework for Mobile Devices,» σε IEEE Transactions on Emerging Topics in Computing, 2016.

[20] R. Hasan και R. Burns, «Where have you been? secure location provenance for mobile devices.,» arXiv preprint arXiv:1107.1821, 2011.

[21] M. Miettinen, N. Asokan, F. Koushanfar, T. D. Nguyen, J. Rios, A.-R. Sadeghi, M. Sobhani και S. Yellapantula, «I know where you are: Proofs of presence resilient to malicious provers,» σε Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, 2015.

[22] C. Lyu, A. Pande, X. Wang, J. Zhu, D. Gu και P. Mohapatra, «CLIP: Continuous location integrity and provenance for mobile phones,» σε 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, 2015.

[23] J. Manweiler, R. Scudellari και L. P. Cox, «SMILE: encounter-based trust for mobile social services.» σε Proceedings of the 16th ACM conference on Computer and communications security., 2009.

[24] G. Brambilla, M. Amoretti και F. Zanichelli, Using Blockchain for Peer-to-Peer Proof-of-Location, arXiv preprint arXiv:1607.00174, 2016

[25] C. Javali, G. Revadigar, K. B. Rasmussen, W. Hu και S. Jha, «I am Alice, I was in wonderland: secure location proof generation and verification protocol.,» σε 2016 IEEE 41st conference on local computer networks (LCN)., 2016.

[26] Joao Ferreira και Miguel L. Pardal, «Witness-based Location Proofs for Mobile Devices» σε 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), 2018

[27] «How does GPS work in my phone? | Android Central» [Ηλεκτρονικό], Available: https://www.androidcentral.com/how-does-gps-work-my-phone

[28] Douceur, John R (2002), «The Sybil Attack», Peer-to-Peer Systems. Lecture Notes in Computer Science, ISBN 978-3-540-44179-3.

[29] «What is Ethereum's Infura? Scalable Access to Ethereum and IPFS» [Ηλεκτρονικό]. Available: https://blockonomi.com/ethereum-infura/

[30] «Documentation | Go Ethereum» [Ηλεκτρονικό], Available: https://geth.ethereum.org/docs/

[31] «web3j — web3j 4.1.0 documentation» [Ηλεκτρονικό], Available: https://web3j.readthedocs.io/en/latest/

[32] «Solidity — Solidity 0.5.9 documentation» [Ηλεκτρονικό], Available: https://solidity.readthedocs.io/en/v0.5.9/

[33] «Metamask/metamask-docs: Metamask project documentation» [Ηλεκτρονικό], Available: https://github.com/MetaMask/metamask-docs

[34] «Ganache | Overview | Documentation | Trufflesuite» [Ηλεκτρονικό], Available: https://www.trufflesuite.com/docs/ganache/overview

[35] «Truffle | Overview | Documentation | Trufflesuite» [Ηλεκτρονικό], Available: https://www.trufflesuite.com/docs/truffle/overview