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

Διαχείριση ταυτότητας χρηστών και υπηρεσιών σε περιβάλλον υπολογιστικού νέφους με τη χρήση Blockchain

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αθανάσιος Γ. Κώστας

Επιβλέπων : Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2020

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

Διαχείριση ταυτότητας χρηστών και υπηρεσιών σε περιβάλλον υπολογιστικού νέφους με τη χρήση Blockchain

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αθανάσιος Γ. Κώστας

Επιβλέπων : Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Ιουλίου 2020.

.....	.....	.....
Συμεών Παπαβασιλείου	Θεοδώρα Βαρβαρίγου	Ιωάννα Ρουσσάκη
Καθηγητής Ε.Μ.Π.	Καθηγήτρια Ε.Μ.Π.	Επίκουρη Καθηγήτρια Ε.Μ.Π

Αθήνα, Ιούλιος 2020

.....
Αθανάσιος, Γ. Κώστας

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

Περίληψη

Το διαδίκτυο αποτελεί στις μέρες μας κομμάτι της καθημερινότητας των ανθρώπων ενώ οι τεχνολογίες που σχετίζονται με αυτό αναπτύσσονται συνεχώς με γοργούς ρυθμούς. Μία από τις τεχνολογίες αυτές είναι το Blockchain, που παρουσίασε ο Satoshi Nakamoto, όπου η αποκεντρωμένη αρχιτεκτονική του, η αμεταβλητότητα των δεδομένων που έχουν εγγραφεί σε αυτό αλλά και η αδιαβλητότητα του χαρακτήρα του έθεσε τα θεμέλια για την εξερεύνηση των δυνατοτήτων του. Η παρούσα εργασία εστιάζει στην παρουσίαση του Blockchain, των χαρακτηριστικών του, αλλά και τα οφέλη που δημιουργούνται όταν συνδυάζεται με τον κλάδο του υπολογιστικού νέφους. Συγκεκριμένα, παρουσιάζεται ένα πρόγραμμα το οποίο έχει ως βάση το Hyperledger Fabric και είναι υπεύθυνο για τη διαχείριση χρηστών-ενοίκων και υπηρεσιών υπολογιστικού νέφους. Το πρόγραμμα αυτό συνιστά ένα έξυπνο συμβόλαιο και επιτρέπει τον αυτοματισμό διεργασιών. Η προτεινόμενη υλοποίηση έχει ως σκοπό να παρουσιάσει έναν τρόπο αξιοποίησης, αλλά και να αποτελέσει μία βάση για μελλοντικές επεκτάσεις στη διαχείριση υπηρεσιών και χρηστών-ενοίκων υπολογιστικού νέφους.

Λέξεις Κλειδιά: Blockchain, υπολογιστικό νέφος, έξυπνα συμβόλαια, υπηρεσίες, ένοικοι υπηρεσιών, Hyperledger Fabric

Abstract

Internet is nowadays part of people's daily lives, while the technologies related to it are constantly evolving at a rapid pace. Blockchain, which was presented by Satoshi Nakamoto, is one of these technologies and its decentralized architecture, the immutability of the data recorded in it and the inviolability of it laid the foundation for exploring its potential. The present thesis focuses on the presentation of Blockchain and the benefits created when combined with cloud computing. Specifically, a Hyperledger Fabric based program is presented and is responsible for cloud tenant and service management. Process automations are enabled as a result of its smart contract nature. The proposed implementation aims to present a way of utilization of cloud and tenant management and set a basis for future extensions.

Keywords: Blockchain, cloud computing, smart contracts, cloud services, tenants, Hyperledger Fabric

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή κ. Συμεών Παπαβασιλείου, του τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής του Εθνικού Μετσόβιου Πολυτεχνείου, για την εμπιστοσύνη που μου έδειξε με την ανάθεση της παρούσας διπλωματικής εργασίας και την άριστη συνεργασία κατά τη διάρκεια εκπόνησης της.
Στη συνέχεια, θα ήθελα να ευχαριστήσω τον Δρ. Δημήτρη Δεχουνιώτη και τον υποψήφιο Διδάκτορα Κωνσταντίνο Παπαδάκη για τον χρόνο που αφιέρωσαν, καθώς και τη συνεχή βοήθεια και καθοδήγηση που μου παρείχαν.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου, για τη στήριξη που μου παρέχει σε όλα τα βήματα της ζωής μου, καθώς και όσους θεωρώ οικογένεια μέσα από το πέρασμα του χρόνου.

Πίνακας Περιεχομένων:

Κατάλογος Σχημάτων

1. Εισαγωγή

1.1 Κατανεμημένα Συστήματα και Blockchain

Η τελευταία δεκαετία έχει χαρακτηριστεί από την άνοδο των κοινωνικών δικτύων, όπως το Twitter, Facebook, Instagram καθώς και των πολυμεσικών υπηρεσιών διαδικτύου, με χαρακτηριστικά τα παραδείγματα των Netflix, YouTube, Spotify, με αποτέλεσμα να αυξάνεται συνεχώς η κίνηση του διαδικτύου και σε συνδυασμό με την έλευση των έξυπνων συσκευών, του Internet-of-Things (IoT) και της νέας γενιάς στις τηλεπικοινωνίες (5G) αναμένεται να δημιουργηθούν νέες απαιτήσεις και κριτήρια στον τρόπο με τον οποίο λειτουργούν τα δίκτυα. Το γεγονός πως βασιζόμαστε ολοένα και περισσότερο στο διαδίκτυο, δεν επηρεάζει μόνο την ποσότητα των δεδομένων που μεταφέρονται μέσω αυτού, αλλά θέτει και προϋποθέσεις για τη λειτουργία του, όπως η peer-to-peer (p2p) ασφάλεια των δεδομένων, η ταχύτητα απόκρισης, ο υψηλός ρυθμός μετάδοσης των δεδομένων, η αξιοπιστία του συστήματος και πολλές ακόμη οι οποίες τείνουν να καθιερωθούν ως σταθερές τις οποίες οι διάφορες υπηρεσίες οφείλουν να εφαρμόσουν προκείμενου να μπορέσουν να ανταποκριθούν στη συνεχώς μεταβαλλόμενη πραγματικότητα. Μοντέλα χρήσης, όπως αυτό που βασίζεται στην αρχιτεκτονική πελάτη-εξυπηρετητή (client-server model) τείνουν να αντικαθίσταται σταδιακά με νέες καινοτόμες τεχνολογίες οι οποίες βασίζονται στην κατανεμημένη πληροφορική (distributed computing) και την αποκεντροποίηση της αρχιτεκτονικής (decentralized architecture) όπως τα ομότιμα δίκτυα (peer-to-peer, p2p) και το blockchain, το οποίο παρουσίασε ο Satoshi Nakamoto με την εργασία του στην οποία εισήγαγε το ψηφιακό κρυπτονόμισμα Bitcoin [1]. Η παρουσίαση του blockchain, με τα ιδιαίτερα χαρακτηριστικά του κίνησε το ενδιαφέρον της τεχνολογικής κοινότητας, η οποία αναζητά συνεχώς νέους τρόπους να τα εκμεταλλευτεί. Σύμφωνα με τους Cachin και Vukolic [2],

Το Blockchain είναι ένα κατανεμημένο ledger (κατάστιχο συναλλαγών) για την καταγραφή συναλλαγών, που διατηρείται από πολλούς κόμβους, οι οποίοι δεν εμπιστεύονται απόλυτα ο ένας τον άλλο, μέσω ενός κατανεμημένου κρυπτογραφικού πρωτοκόλλου, χωρίς την ύπαρξη κεντρικής αρχής. Οι κόμβοι επικυρώνουν τις πληροφορίες που είναι προς προσθήκη στο Blockchain και ένα consensus protocol (πρωτόκολλο συναίνεσης) διασφαλίζει πως όλοι οι κόμβοι συμφωνούν σε μια μοναδική σειρά στην οποία θα προστεθούν οι

πληροφορίες. Πολλά Blockchain μπορούν να εκτελούν προκαθορισμένες εργασίες μέσω των smart contracts (έξυπνων συμβολαίων).

Η αποκεντρωμένη αρχιτεκτονική ουσιαστικά κατανέμει το φόρτο εργασίας του δικτύου σε πολλαπλούς κόμβους αντί να τον αναθέτει σε κάποιο κεντρικό διακομιστή (server). Το χαρακτηριστικό αυτό προσδίδει στα δίκτυα αυξημένη αξιοπιστία, καθώς υποστηρίζουν συνήθως μηχανισμούς ανοχής σε σφάλματα (Fault Tolerant), καλύτερες δυνατότητες κλιμάκωσης (scalability) όπου νέοι κόμβοι-υπολογιστές μπορούν να προστίθενται συνεχώς προκειμένου να αυξηθεί η υπολογιστική ισχύς του δικτύου, καθώς και μεγαλύτερη ασφάλεια στις συναλλαγές, καθώς οι συναλλαγές και τα δεδομένα δεν περνούν πλέον από κάποια κεντρική αρχή που έχει πρόσβαση στο σύνολο των δεδομένων, αλλά κρυπτογραφούνται και κατανέμονται σε ένα σύνολο κόμβων.

Αντίθετα, η κεντροποιημένη αρχιτεκτονική δομείται γύρω από κάποιον κεντρικό διακομιστή ο οποίος είναι υπεύθυνος για τις εφαρμογές, τις υπηρεσίες και την επεξεργασία των δεδομένων. Η σχεδιαστική αυτή επιλογή, έχει ως αποτέλεσμα σε περίπτωση βλάβης κάποιου διακομιστή να προκληθούν προβλήματα στην εύρυθμη λειτουργία του συστήματος. Οι διάφοροι πελάτες (clients) οι οποίοι συνδέονται στον διακομιστή (server), δε θα είναι σε θέση να ανταποκριθούν στα αιτήματα των χρηστών, καθώς ο διακομιστής δε θα μπορεί να τα επεξεργαστεί, ενώ σε περίπτωση που ο διακομιστής είναι καίριας σημασίας υπάρχει η πιθανότητα έκθεσης προσωπικών δεδομένων. Στον τομέα αυτό υπάρχει ένα σύνολο κινδύνων, όπως επιθέσεις άρνησης εξυπηρέτησης (Denial-of-Service Attack, DDos Attack) αλλά και επιθέσεις που έχουν ως στόχο την απόκτηση δεδομένων από τον διακομιστή. Η δυνατότητα κλιμάκωσης (scalability) του συστήματος είναι επίσης περιορισμένη στις δυνατότητες αναβάθμισης του διακομιστή και είναι μια διαδικασία που από ένα σημείο και μετά αποδεικνύεται οικονομικά ασύμφορη. Τέλος, ο περιορισμός του εύρους ζώνης (bandwidth) λειτουργεί ως εμπόδιο, καθώς σε περιόδους αυξημένης κίνησης στον διακομιστή υπάρχει η πιθανότητα να μην μπορούν να εξυπηρετηθούν όλα τα αιτήματα προς αυτόν.

Με τα δεδομένα αυτά, από την ημέρα που παρουσιάστηκε έως σήμερα, το Blockchain έχει καταστεί αντικείμενο αντιπαράθεσης ανάμεσα στους υποστηρικτές αλλά και τους επικριτές του, παρόλα αυτά γνώρισε και συνεχίζει να γνωρίζει μεγάλη άνθιση καθώς με τις διάφορες παραλλαγές του ολοένα και περισσότερες υπηρεσίες και εφαρμογές στρέφονται προς τη λύση αυτή, αναγνωρίζοντας τα πλεονεκτήματα που προσφέρει.

1.1.1 Υπολογιστικό Νέφος

Η έννοια του υπολογιστικού νέφους (cloud computing) υπάρχει εδώ και δεκαετίες, και περιστρέφεται γύρω από την κεντρική ιδέα ότι διάφοροι χρήστες, μέσω των δικών τους τερματικών, δίχως την ανάγκη ύπαρξης μεγάλης υπολογιστικής ισχύος, μπορούν να έχουν πρόσβαση απομακρυσμένα σε υπηρεσίες ή κεντρικούς υπολογιστές μεγάλων δυνατοτήτων. Αυτή η ιδέα έδωσε ώθηση σε πολλές εταιρείες να ασχοληθούν με το αντικείμενο και να δημιουργήσουν τις δικές τους υπηρεσίες προς τους χρήστες. Χαρακτηριστικό παράδειγμα είναι η δυνατότητα δημιουργίας πολλαπλών εικονικών μηχανών (virtual machines, VMs), όπου οι χρήστες μπορούν μέσω οποιουδήποτε υπολογιστή να χειρίζονται και να πραγματοποιούν εργασίες για τις οποίες θα απαιτούνταν υπό διαφορετικές συνθήκες μεγάλες ανάγκες για τοπική ύπαρξη σε υλικό (hardware) και λογισμικό υπολογιστών (software) γεγονός που θα αύξανε το κόστος, τις ενεργειακές ανάγκες αλλά και τον χρόνο πραγματοποίησης των εργασιών αυτών καθιστώντας ασύμφορο πολλές φορές το συνολικό εγχείρημα για ιδιώτες αλλά και για εταιρείες. Συνεπώς, η δυνατότητα κάποιου χρήστη να πραγματοποιήσει τις εργασίες που επιθυμεί δεν περιορίζεται από το δικό του hardware και software, αλλά με προϋπόθεση μία σύνδεση στο διαδίκτυο μπορεί να έχει πρόσβαση στους πόρους που επιθυμεί, οι οποίοι μπορούν και να μεταβάλλονται δυναμικά προκειμένου να καλύπτουν τις ανάγκες του.

Με το πέρασμα του χρόνου η ζήτηση για cloud υπηρεσίες αυξάνεται συνεχώς, με αποτέλεσμα ολοένα και περισσότερες εταιρείες να ασχολούνται με το αντικείμενο αυτό και να κάνουν τις δικές τους προτάσεις στο χώρο. Χαρακτηριστικά είναι τα παραδείγματα εταιρειών όπως η Google, Amazon, Microsoft, IBM, Oracle κ.α. Τα πλεονεκτήματα του cloud computing εκτείνονται στο κόστος απόδοσης, την έλλειψη ουσιαστικού περιορισμού για την αποθήκευση των δεδομένων και τη διαθέσιμη για τον χρήστη χωρητικότητα, την αυτόματη δημιουργία αντιγράφων ασφάλειας και αποκατάστασης, την εύκολη πρόσβαση από οποιαδήποτε συσκευή έχει σύνδεση στο ίντερνετ, την εύκολη και μεγάλη κλιμάκωση των παρεχόμενων υπηρεσιών, την αυτόματη ενσωμάτωση και ανάπτυξη του λογισμικού, όπου με αυτοματοποιημένες διαδικασίες και σε ελάχιστο χρόνο κάποιος μπορεί να έχει ένα έτοιμο και ενημερωμένο σύστημα με το λογισμικό που επιθυμεί, αλλά και τη γενικότερη συνεχή παροχή εργαλείων διαχείρισης και νέων υπηρεσιών στους χρήστες.

Από την άλλη πλευρά όμως, υπάρχουν περιορισμοί και ανησυχίες για τη χρήση του cloud computing. Αναλύοντας το θέμα αυτό, ξεκινάμε από την ασφάλεια και την ιδιωτικότητα. Συγκεκριμένα, η εμπιστοσύνη προς τον πάροχο είναι καίριας σημασίας. Όταν τα δεδομένα

μεταφέρονται στο cloud, οι πάροχοι μπορούν να επιλέξουν που θα τα αποθηκεύσουν. Η φυσική αποθήκευση των δεδομένων καθορίζει και τη νομοθεσία γύρω από τα δεδομένα. Για παράδειγμα σε κάποιες χώρες δεν επιτρέπονται συγκεκριμένες τεχνικές κρυπτογραφίας, ενώ κρατικοί νόμοι μπορεί να υποχρεώνουν τους παρόχους να έχουν ευαίσθητα δεδομένα, όπως το ιστορικό υγείας ενός ασθενή εντός της επικράτειας της χώρας. Αντίστοιχο θέμα είναι η έλλειψη δυνατότητας μεταφοράς από πάροχο σε πάροχο. Συγκεκριμένα, μπορεί ένας χρήστης να επιθυμεί να μεταφέρει τα δεδομένα και τις εφαρμογές του από έναν πάροχο σε έναν άλλο, για παράδειγμα επειδή σταματά πλέον να τον καλύπτει στις ανάγκες του ή εξαιτίας μιας αλλαγής στην πολιτική επεξεργασίας των δεδομένων του, παρόλα αυτά, ο τρόπος αποθήκευσης των δεδομένων δεν είναι τυποποιημένος, με αποτέλεσμα να μην είναι εύκολη η φορητότητα των δεδομένων από πάροχο σε πάροχο. Ένα ακόμη θέμα προς εξέταση είναι αυτό της διαθεσιμότητας και της απόδοσης. Όπως είναι αναμενόμενο οι χρήστες έχουν συγκεκριμένες απαιτήσεις από τους παρόχους σχετικά με το επίπεδο των υπηρεσιών που παρέχονται. Οι απαιτήσεις αυτές περιλαμβάνουν τη διαθεσιμότητα των υπηρεσιών, τη γενική απόδοση του συστήματος αλλά και την αντιμετώπιση σε περίπτωση κάποιας βλάβης ή δυσλειτουργίας και τα μέτρα που θα λάβει ο πάροχος για να ανταποκριθεί σε αυτές τις περιπτώσεις. Με λίγα λόγια επιθυμούν κάποιου είδους εξασφάλιση για τα δεδομένα τους πριν τα μεταφέρουν στο cloud, για την διαθεσιμότητα που θα έχουν και την ταχύτητα του συστήματος, ζητώντας εγγυήσεις οι οποίες μπορούν να καθορίζονται σε συμφωνητικό (Service-Level Agreement, SLA) μεταξύ των ενδιαφερόμενων μελών. Επίσης, οι πάροχοι επιθυμούν να μεγιστοποιήσουν τη χρησιμοποίηση των διαθέσιμων πόρων τους με σκοπό το μέγιστο κέρδος. Αυτό σημαίνει όμως πως πόροι όπως πυρήνες από κεντρικές μονάδες επεξεργασίας (CPU cores), μονάδες αποθήκευσης (disk space) και το εύρος ζώνης του δικτύου (bandwidth) πρέπει να κατανέμονται μεταξύ εικονικών μηχανών που μπορούν να τρέχουν ετερογενείς φόρτους εργασίας οι οποίοι μπορεί να ανήκουν και σε κακόβουλους χρήστες. Συνεπώς πρέπει να χρησιμοποιείται κάποιου είδους μηχανισμός που θα αποφασίζει πως θα κατανέμονται οι διαθέσιμοι πόροι στους ενδιαφερόμενους χρήστες, προκειμένου να μην έχουμε κακόβουλους χρήστες να μοιράζονται πόρους με καλόβουλους χρήστες.

1.1.2 Blockchain και Cloud Computing

Η βελτίωση της ποιότητας των υποδομών των δικτύων έδωσε μία ώθηση στη χρήση υπηρεσιών που βασίζονται στο cloud, καθώς πλέον η ταχύτητα και η ποιότητα των δικτύων,

ειδικά με την έλευση του 5G, επιτρέπει τη μεταφορά μεγαλύτερου όγκου δεδομένων, καθιστώντας εφικτές υπηρεσίες που μέχρι πρότινος δεν ήταν λόγω τεχνικών περιορισμών. Παράλληλα η εμφάνιση του blockchain, με τα ιδιαίτερα χαρακτηριστικά του, το κατέστησαν ενδιαφέρουσα λύση για μια πληθώρα εφαρμογών. Μια από αυτές είναι και το cloud computing. Αναλυτικά, στο blockchain υπάρχουν αντίγραφα του συνόλου των συναλλαγών που έχουν πραγματοποιηθεί σε όλους τους κόμβους, με αποτέλεσμα να είναι εξαιρετικά δύσκολο να χειραγωγήσει κάποιος μη εξουσιοδοτημένος χρήστης τα δεδομένα του δικτύου. Συνεπώς, η αποκεντροποίηση της καταγραφής των δεδομένων δημιούργησε ένα ισχυρό εργαλείο ελέγχου του ιστορικού. Το εργαλείο αυτό συνεισφέρει και στην γενικότερη αποδοτικότητα του συστήματος, καθώς πλέον σε ελάχιστο χρόνο μπορεί να ανατρέξει κάποιος στο ιστορικό των δεδομένων του συστήματος αλλά και να καταχωρήσει νέα δεδομένα. Η κρυπτογραφία που χρησιμοποιείται παίζει επίσης σημαντικό ρόλο στην ασφάλεια των δεδομένων. Η χρήση blockchain καθιστά τα συστήματα ανεξάρτητα από τη φυσική τοποθεσία που βρίσκονται αποθηκευμένα τα δεδομένα. Αυτό έχει ως αποτέλεσμα τα δεδομένα να μην κινδυνεύουν σε περίπτωση τεχνικής βλάβης αλλά ακόμη και φυσικής καταστροφής στο μέρος που είναι αποθηκευμένα, καθώς σε μία τέτοια περίπτωση υπάρχουν ακριβή αντίγραφα στους υπόλοιπους κόμβους του συστήματος. Με αυτό το τρόπο το σύστημα δεν είναι επιρρεπές σε αποτυχίες κόμβων και όλοι οι υπόλοιποι κόμβοι συνεχίζουν κανονικά τη λειτουργία τους. Αντίστοιχα μόλις ο κόμβος που αντιμετώπισε τη βλάβη επανέλθει στο δίκτυο θα ενημερωθεί με όλες τις συναλλαγές που πραγματοποιήθηκαν ακόμη και όσο αυτός δεν ήταν σε λειτουργία. Συνεπώς, ο αμετάβλητος χαρακτήρας του blockchain, η διαφάνεια στη χρήση του, η ασφάλεια, η αξιοπιστία, η εύκολη και γρήγορη πρόσβαση στο σύνολο του ιστορικού συναλλαγών, σε συνδυασμό με την ύπαρξη έξυπνων συμβολαίων (smart contracts), τα οποία κάνουν εφικτή την αυτοματοποίηση σε μεγάλο βαθμό της διαχείρισης cloud υπηρεσιών, και τον έλεγχο του χρήστη στις παρεχόμενες υπηρεσίες καθώς πλέον μπορεί να γνωρίζει εκ των προτέρων τις λεπτομέρειες λειτουργίας του συστήματος, το καθιστούν ιδανικό για εφαρμογές όπως το cloud computing.

1.2 Αντικείμενο Διπλωματικής

Σύμφωνα με όσα ειπώθηκαν ανωτέρω, η έλευση του blockchain έδωσε έναν εναλλακτικό τρόπο προσέγγισης σε μια πληθώρα εφαρμογών και υπηρεσιών, μεταξύ των οποίων και η διαχείριση υπηρεσιών σε cloud περιβάλλον. Συνεπώς, με βάση και το θεωρητικό υπόβαθρο

που θα αναλυθεί στη συνέχεια, δημιουργήθηκε μία εφαρμογή διαχείρισης cloud χρηστών (tenants) και υπηρεσιών (services) βασιζόμενη σε smart contract αξιοποιώντας την κατανεμημένη αρχιτεκτονική που προσφέρει ένα ιδιωτικό (private, permissioned) blockchain.

Συγκεκριμένα, το permissioned blockchain που χρησιμοποιήθηκε είναι το Hyperledger Fabric (HLF). Το HLF έχει αρκετές διαφορές σε σχέση με το Bitcoin που αναφέρθηκε πιο πάνω. Συγκεκριμένα, το HLF αποτελείται από γνωστούς χρήστες, συνεπώς υπάρχει μία σχετική εμπιστοσύνη ανάμεσα τους, σε αντίθεση με το Bitcoin, όπου οι χρήστες είναι ανώνυμοι και δεν υπάρχει εμπιστοσύνη ανάμεσα στα συναλλασσόμενα μέλη. Επίσης, το Bitcoin περιλαμβάνει την ύπαρξη κρυπτονομισμάτων, τα οποία αποδίδονται ως ανταμοιβή για μία διαδικασία που ονομάζεται εξόρυξη (mining) και έχει ως σκοπός την επιβεβαίωση των συναλλαγών που θα συμπεριληφθούν στο blockchain. Η διαδικασία αυτή απαιτεί μεγάλη υπολογιστική ισχύ και δαπάνη ενέργειας. Αντίθετα, η διαδικασία αυτή πραγματοποιείται με διαφορετικό τρόπο στο HLF, το οποίο δε περιλαμβάνει κρυπτονομίσματα και χρησιμοποιεί διαφορετικούς, αλλά ταχύτερους και λιγότερο δαπανηρούς αλγορίθμους για την επίτευξη του ίδιου στόχου. Η επίτευξη της συναίνεσης (consensus) για την επαλήθευση των συναλλαγών, αν και ταχύτερη, επιτυγχάνεται με την απαραίτητη ασφάλεια που απαιτείται ενώ η διαφάνεια, η αμεταβλητότητα, η εύκολη πρόσβαση στο ιστορικό των συναλλαγών και η χρήση smart contracts, όπου στο HLF ονομάζονται chaincode, προσέφεραν τα απαιτούμενα χαρακτηριστικά προκειμένου να αναπτυχθεί η εφαρμογή μας και να παρέχει τους απαιτούμενους αυτοματισμούς στη διαχείριση των χρηστών και των υπηρεσιών.

Οι διαδικασίες που πραγματοποιεί το σύστημα εκτείνονται στη διαχείριση της ταυτότητας του κάθε χρήστη, την πιστοποίηση των χρηστών, τις εξουσιοδοτήσεις ανάλογα τις ανάγκες, καθώς και ένα σύστημα χρέωσης σύμφωνα με τη χρήση των υπηρεσιών που γίνονται. Η ανάπτυξη του chaincode έγινε σε Go, καθώς το HLF, σε αντίθεση με πολλά άλλα blockchain που υποστηρίζουν μόνο γλώσσες ειδικού σκοπού, υποστηρίζει διαδεδομένες γλώσσες προγραμματισμού, όπως η Java, Go και Node.js. Επίσης, το HLF υποστηρίζει τρία πρωτόκολλα συναίνεσης, το Solo, το Kafka και το νεότερο Raft. Για την πραγματοποίηση της παρούσας εργασίας, έγινε χρήση του Raft. Περισσότερες λεπτομέρειες σχετικά με τη λειτουργία του blockchain αλλά και ειδικότερα του HLF θα παρουσιαστούν στο τρίτο κεφάλαιο.

1.3 Οργάνωση Κειμένου

Το κείμενο αναπτύσσεται με την ακόλουθη λογική. Το παρόν κεφάλαιο, λειτουργεί ως μια εισαγωγή στο αντικείμενο της διπλωματικής εργασίας. Στο κεφάλαιο 2 παρουσιάζεται ένα σύνολο από σχετικές εργασίες, είτε με τη χρήση τεχνολογίας blockchain είτε χωρίς. Στο κεφάλαιο 3 γίνεται μια γενική περιγραφή της τεχνολογίας του blockchain αλλά και ειδικότερα του HLF, όπου παρουσιάζονται τα βασικά στοιχεία του και τα πλεονεκτήματα που προσφέρει. Το κεφάλαιο 4 περιλαμβάνει μία αναλυτική παρουσίαση του συστήματος που δημιουργήθηκε, όπου παρουσιάζεται με λεπτομέρειες η δομή του smart contract καθώς και των στοιχείων που περιλαμβάνει. Στο κεφάλαιο 5 γίνεται μία ανάλυση του τρόπου λειτουργίας του προγράμματος και παρουσιάζονται κάποιες τεχνικές λεπτομέρειες. Το κεφάλαιο 6 περιλαμβάνει τα συμπεράσματα που αποκομίσαμε από την εργασία και την μελλοντική εξέλιξη που θα μπορούσε να έχει. Ενώ, στο κεφάλαιο 7 παρουσιάζεται η βιβλιογραφία που χρησιμοποιήθηκε.

2. Σχετικές Εργασίες

Στο παρόν κεφάλαιο θα παρουσιαστούν εργασίες που πραγματεύονται το αντικείμενο και τις τεχνολογίες που εξετάζει και η παρούσα διπλωματική εργασία. Ο χωρισμός των εργασιών θα γίνει σε δύο τμήματα, αρχικά θα εξετάσουμε εργασίες που ασχολούνται με την οργάνωση, διαχείριση και αυτοματοποίηση σε cloud περιβάλλον, ενώ στη συνέχεια θα εξετάσουμε αντίστοιχες εργασίες που βασίζονται σε blockchain υλοποιήσεις και είναι πιο κοντά σε αυτό που υλοποιήθηκε στην τρέχουσα εργασία.

2.1 Trusted Cloud Orchestration Management

2.1.1 A Robust Reputation Management Mechanism in the Federated Cloud [3]

Η εργασία αυτή πραγματεύεται τη διάθεση υπολογιστικών πόρων προς ενοικίαση στο πλαίσιο των υπηρεσιών του cloud computing. Συγκεκριμένα, αναλύει πως η λύση του cloud computing προσφέρει μία οικονομική επιλογή για τις απαιτήσεις σε υπολογιστικούς πόρους. Οι πάροχοι από τη μεριά τους, προκειμένου να ρίξουν το κόστος των υπηρεσιών τους, μοιράζουν τους διαθέσιμους πόρους τους σε πολλαπλούς χρήστες-ενοίκους (tenants). Η τεχνική αυτή όμως, δημιουργεί ζητήματα ασφάλειας και ιδιωτικότητας καθώς κανάλια επικοινωνίας και άλλοι πόροι μοιράζονται. Χαρακτηριστικά είναι τα παραδείγματα Side-Channel επιθέσεων, Probe επιθέσεων και άλλων, λειτουργώντας ως αποτρεπτικός παράγοντας για τη χρήση υπηρεσιών cloud computing. Οι χρήστες βασίζονται στον πάροχο (Cloud Provider, CP), προκειμένου να ελέγχει την αξιοπιστία των συν-ενοικιαστών. Ο πάροχος όμως από την πλευρά του, ενδιαφέρεται για τη μεγιστοποίηση της αξιοποίησης των διαθέσιμων πόρων, συνεπώς επιτρέπει τη μέγιστη δυνατή συν-ενοικίαση, ανεξαρτήτως της συμπεριφοράς των χρηστών που ενοικιάζουν τις υπηρεσίες. Η εμπιστοσύνη στον πάροχο, όπως βλέπουμε, είναι πολύ σημαντική για τους χρήστες, συνεπώς οι συγγραφείς στη συνέχεια της εργασίας προτείνουν έναν μηχανισμό διαχείρισης της φήμης (Reputation Management Mechanism, RMM), ο οποίος ενθαρρύνει τους παρόχους να πραγματοποιούν διακρίσεις μεταξύ καλόβουλων και κακόβουλων χρηστών με τέτοιο τρόπο έτσι ώστε να μην μοιράζονται υπολογιστικούς πόρους μεταξύ τους. Το άρθρο καταλήγει, υποστηρίζοντας

πως η συν-μίσθωση καθιστά το cloud computing προσιτό αλλά εισάγει ένα νέο κίνδυνο, ο οποίο προέρχεται από τους κακόβουλους συν-ενοικιαστές, ενώ το RMM που προτείνουν οι συγγραφείς ενθαρρύνει τη τμηματοποίηση μεταξύ καλόβουλων και κακόβουλων χρηστών, δηλαδή ένας καλόβουλος χρήστης να έχει μόνο άλλους καλόβουλους χρήστες ως συν-ενοικιαστές.

2.1.2 Multi Cloud IaaS with Domain Trust in OpenStack [4]

Το θέμα της εργασίας αυτής είναι και πάλι το επίπεδο εμπιστοσύνης στις cloud υπηρεσίες και το πως μπορούν πολλαπλές cloud υπηρεσίες να μοιραστούν τους διαθέσιμους πόρους τους δημιουργώντας cloud ομοσπονδίες (federations). Αναλυτικά, η εργασία αναφέρει πως καθώς οι cloud υπηρεσίες διεισδύουν ολοένα και περισσότερο στους επιχειρηματικούς τομείς, η σύγχρονη πρόκληση που καλούνται να αντιμετωπίσουν είναι το πως μπορούν να μοιραστούν οι διαθέσιμοι πόροι προκειμένου να ανταποκριθούν στις σύγχρονες απαιτήσεις. Επί του παρόντος, υπηρεσίες όπως το OpenStack και το Amazon Web Services (AWS) προσφέρουν περιορισμένες δυνατότητες για πρόσβαση σε πολλαπλές cloud υπηρεσίες. Στη συνέχεια παρουσιάζεται ένα μοντέλο εμπιστοσύνης (trust model) μεταξύ πολλαπλών cloud υπηρεσιών, το οποίο επιτρέπει τη κοινή χρήση πόρων μέσα από ομοιογενή clouds. Σκοπός των συγγραφέων είναι να δημιουργήσουν μία trust δομή (framework) ως επέκταση του OpenStack. Τέλος, γίνεται περαιτέρω παρουσίαση του μοντέλου της επέκτασης που δημιούργησαν οι συγγραφείς.

2.1.3 Towards Dynamic Tenant Management for Microservice based Multi-Tenant SaaS Applications [5]

Το άρθρο αυτό αναφέρεται στις cloud εφαρμογές πολλαπλών ενοίκων, όπου περισσότεροι του ενός ετερογενείς ενοικιαστές μοιράζονται μία εκτέλεση (instance) μίας εφαρμογής. Όσο αυξάνεται ο βαθμός της κατανομής των πόρων μεταξύ των ενοικιαστών αντίστοιχα μειώνεται το λειτουργικό κόστος της υπηρεσίας. Συνεπώς, η εργασία αυτή προτείνει ένα μεθοδολογικό πλαίσιο για τη διαχείριση των ενοικιαστών για μία υπηρεσία βασισμένη σε μια cloud εφαρμογή πολλαπλών ενοίκων. Επίσης, παρουσιάζεται μία μελέτη για να αποδειχτεί πως μπορούν υπηρεσίες να διανεμηθούν μεταξύ πολλαπλών ενοικιαστών προκειμένου να αυξηθεί η αξιοποίηση των πόρων και ταυτόχρονα να ανταποκρίνεται με τις ετερογενείς απαιτήσεις τον ενοικιαστών. Η προτεινόμενη προσέγγιση μεγιστοποιεί τη

χρήση των διαθέσιμων πόρων, μειώνοντας με αυτό το τρόπο το κόστος ενώ ταυτόχρονα διαχειρίζεται τις απαιτήσεις για απομόνωση ετερογενών ενοικιαστών, οδηγώντας σε μια υβριδική ανάπτυξη της αρχιτεκτονικής σε επίπεδο εφαρμογής, όπου ορισμένες υπηρεσίες μπορούν να είναι πολλαπλών ενοίκων ενώ άλλες να είναι πολλαπλών instances.

2.1.4 Unified Cloud Access Control Model for Cloud Storage Broker [6]

Το άρθρο αυτό εξετάζει το Cloud Storage Broker (CSB), το οποίο παρέχει προστιθέμενη αξία σε υπηρεσίες cloud αποθήκευσης για επιχειρηματική χρήση, αξιοποιώντας μία multi-cloud αρχιτεκτονική αποθήκευσης. Συγκεκριμένα, το CSB είναι μία εφαρμογή (Software-as-a-Service, SaaS), που διαχειρίζεται τη σχέση μεταξύ ενός χρήστη και πολλαπλών παρόχων cloud υπηρεσιών, (Cloud Service Providers, CSP). Παρέχει υπηρεσίες για τον χρήστη, όπως διαχείριση ταυτότητας (Identity Management) και κρυπτογράφηση των δεδομένων χωρίς να χρειάζεται ο χρήστης να διαχειρίζεται άμεσα τα δεδομένα του σε πολλαπλούς CSP. Πολλές προκλήσεις όμως προέρχονται από το τρόπο διαχείρισης των πόρων και τον έλεγχο πρόσβασης σε πολλαπλούς παρόχους cloud υπηρεσιών (Cloud Service Providers, CSP) για εξουσιοδοτημένα CSB μέλη. Στο άρθρο αυτό λοιπόν, προτείνεται ένα ενοποιημένο μοντέλο ελέγχου πρόσβασης στο cloud, το οποίο βοηθάει στη διαχείριση των cloud πόρων και τον έλεγχο πρόσβασης, διαχείρισης και αποθήκευσης για το Amazon Web Services (AWS) και το Google Cloud Platform (GCP). Η πρόταση των συγγραφέων προσφέρει έλεγχο πρόσβασης των cloud πόρων βασισμένο σε ρόλους, για τα ενδιαφερόμενα CSB μέλη, αναθέτοντας τα απαραίτητα προνόμια και μία λίστα ελέγχου πρόσβασης για τους πόρους του cloud και τα μέλη του CSB. Εφαρμόστηκε ένα ενοποιημένο μοντέλο σε ένα CSB σύστημα το οποίο ονομάζεται CloudRAID για επιχειρήσεις (CfB), με τα αποτελέσματα της αξιολόγησης να δείχνουν ότι παρέχει ασφάλεια στο επίπεδο του συστήματος και του cloud για CfB, καθώς και κεντρική διαχείριση ελέγχου πόρων και ελέγχου πρόσβασης σε πολλαπλά CSP.

2.2 Blockchain and Trusted Cloud Orchestration Management

2.2.1 The Nubo Virtual Services Marketplace [7]

Στο άρθρο αυτό περιγράφεται μία αγορά εικονικών υπηρεσιών, με την ονομασία Nubo, η οποία είναι σχεδιασμένη για τη σύνδεση αγοραστών ή ενοικιαστών εικονικών υπηρεσιών

με τους παρόχους των υπηρεσιών αυτών σε μία cloud computing πλατφόρμα. Η εικονική αυτή αγορά υλοποιείται ως μια συλλογή από κατανεμημένες μικρό-υπηρεσίες μαζί με μία αγορά που λειτουργεί ως εφαρμογή διαδικτύου. Το Nubo βασίζεται στο σύστημα διαχείρισης ενοικιαστών και υπηρεσιών Saranyu [8] και περιλαμβάνει υποστήριξη για την ενσωμάτωση υπηρεσιών που υποστηρίζουν εγγενώς την πολύ-ενοικίαση, καθώς και για τις υπηρεσίες που δε την υποστηρίζουν. Οι χρήστες και οι υπηρεσίες αντιπροσωπεύονται με στατικά (μη τροποποιήσιμα) έξυπνα συμβόλαια. Στη συνέχεια παρουσιάζονται οι σχεδιαστικοί στόχοι και η συνολική αρχιτεκτονική του Nubo, αναλύοντας λεπτομέρειες σχετικά με το πως το Saranyu χρησιμοποιεί το blockchain και τα smart contracts για να παρέχει απόδοση και επεκτασιμότητα.

2.2.2 Saranyu: Using Smart Contracts and Blockchain for Cloud Tenant Management [8]

Η εργασία αυτή παρουσιάζει μία εφαρμογή που αποτελείται από smart contracts σε ένα permissioned blockchain (όπου μόνο εγκεκριμένοι χρήστες έχουν δικαίωμα για ανάγνωση και εγγραφή στο ledger) για τη διαχείριση χρηστών (Tenants) και υπηρεσιών (Services) σε ένα κέντρο δεδομένων υπολογιστικού νέφους (cloud computing data center). Το σύστημα υποστηρίζει τέσσερις υπηρεσίες: Identity Management (διαχείριση ταυτότητας), Authentication (πιστοποίηση), Authorization (εξουσιοδότηση), Charging (χρέωση). Η υλοποίηση του συστήματος γίνεται βασιζόμενο στο Quorum blockchain, μια permissioned υλοποίησης του Ethereum και σκοπός είναι η εκμετάλλευση των βασικών στοιχείων του blockchain, όπως: Ασφάλεια (Security), μη-αποκήρυξη (Non-Repudiation), αντοχή σε παραβιάσεις (Tamper-Resistance), εύκολη πρόσβαση στο ιστορικό συναλλαγών (Easy transaction history access), προκειμένου να αυξηθεί η διαφάνεια και η εμπιστοσύνη στη διαχείριση χρηστών και υπηρεσιών στο cloud. Συνεπώς το Saranyu λειτουργεί ως μια κατανεμημένη εφαρμογή βασιζόμενη στο Quorum. Βασικά στοιχεία του συστήματος είναι η ικανότητα να δημιουργεί προσαρμοσμένα smart contracts για συγκεκριμένες κατηγορίες χρηστών και διαπιστευτήρια για την ανάθεση εξουσιοδότησης για τη χρήση υπηρεσιών σε ενοικιαστές, υπό-ενοικιαστές και άλλες υπηρεσίες.

2.2.3 SCPKI: A Smart Contract-based PKI and Identity System [9]

Η εργασία αυτή ασχολείται με την υποδομή δημόσιου κλειδιού (Public Key Infrastructure,PKI). Συγκεκριμένα παρουσιάζει το SCPKI, ένα εναλλακτικό PKI σύστημα το οποίο βασίζεται σε αποκεντρωμένο σχεδιασμό χρησιμοποιώντας ένα μοντέλο χρήσης που απαρτίζεται από ένα δίκτυο εμπιστοσύνης (web-of-trust) και ένα έξυπνο συμβόλαιο (smart contract) βασισμένο στο Ethereum blockchain. Σκοπός είναι να καταστεί εύκολα ικανό να ανιχνεύονται τα πλαστά πιστοποιητικά (rogue certificates) όταν αυτά εκδίδονται. Το web-of-trust μοντέλο είναι σχεδιασμένο με τέτοιο τρόπο, ώστε μία οντότητα ή μία αρχή (authority) στο σύστημα να μπορεί να επιβεβαιώσει ή να εγγυηθεί για τις ιδιότητες της ταυτότητας μίας άλλης οντότητας. Η ανάγκη για τη δημιουργία αυτής της εναλλακτικής, σε σχέση με το συγκεντρωτικό τρόπο έκδοσης πιστοποιητικών για την επιβεβαίωση της ταυτότητας που χρησιμοποιείται ακόμη και σήμερα, είναι η αντιμετώπιση των μειονεκτημάτων που προκύπτουν από τον συγκεντρωτικό και αδιαφανή τρόπο που αυτά είναι σχεδιασμένα. Συγκεκριμένα, έχουν υπάρξει κατά το παρελθόν περιπτώσεις όπου αρχές έκδοσης πιστοποιητικών εξέδιδαν rogue certificates για στοχευμένες επιθέσεις, κάτι που ήταν δύσκολο να ανιχνευθεί άμεσα, καθώς οι αρχές έκδοσης πιστοποιητικών δεν έχουν ένα διαφανές σύστημα για τα πιστοποιητικά που εκδίδουν. Επιπλέον, το συγκεντρωτικό σύστημα έκδοσης πιστοποιητικών, καθιστά μη πρακτικό να σταματά κάποιος να εμπιστεύεται όλα τα πιστοποιητικά που εκδίδει μια αρχή η οποία εξέδωσε κάποιο rogue certificate, καθώς αυτό θα διατάρασσε την TLS (Transport Layer Security).

2.2.4 Blockchain-Based Decentralized Applications for Multiple Administrative Domain Networking [10]

Η εργασία αυτή πραγματεύεται τα εξελισσόμενα σενάρια δικτύωσης, τα οποία περιλαμβάνουν πολύ-διοικητικές υπηρεσίες δικτύων. Τέτοια σενάρια, όπως η επαυξημένη πραγματικότητα (Augmented Reality), η επικοινωνία των αυτοκινήτων και το IoT, οδηγούν σε νέες επιχειρηματικές ευκαιρίες. Παράλληλα όμως, εμφανίζονται προκλήσεις που πρέπει να αντιμετωπιστούν. Ως πιθανή προσέγγιση των επερχόμενων απαιτήσεων και παρέχοντας βασικές αρχές για να συμπεριληφθούν αναλυτικά στοιχεία, αυτοματισμοί και κατανεμημένες ενορχηστρώσεις (orchestration) πολύ-διοικητικών εφαρμογών, οι συγγραφείς της εργασίας διερευνούν αποκεντρωμένες εφαρμογές (DApps) βασισμένες στο

blockchain, στο πλαίσιο λειτουργικών φάσεων για την υποστήριξη δικτύωσης μεταξύ πολλαπλών διοικητικών τομέων.

2.2.5 Securing Virtual Machine Orchestration with Blockchains [11]

Οι συγγραφείς της εργασίας δημιούργησαν το VMOA blockchain, ένα private blockchain που ειδικεύεται στη διαχείριση του ελέγχου ταυτότητας των εντολών ενορχήστρωσης εικονικών μηχανών στο cloud computing και λειτουργιών εικονικοποίησης υπηρεσιών του δικτύου. Στο πλαίσιο αυτό καθορίστηκαν βασικές αρχές λειτουργίας, ενώ γίνεται χρήση παραδειγμάτων προκειμένου να εξηγηθούν οι σχεδιαστικές επιλογές. Ο λόγος που βασίστηκαν στο blockchain ήταν πως η αποκεντρωμένη αρχιτεκτονική του, που βασίζεται σε ένα κατανεμημένο ψηφιακό κατάστιχο συναλλαγών καταγραφής των δεδομένων (ledger) μπορεί να χρησιμοποιηθεί για να διαχειριστεί με ασφάλεια κάθε είδος στοιχείων, κατασκευάζοντας ένα σύστημα που είναι ανεξάρτητο από οποιαδήποτε κεντρική αρχή.

2.2.6 Trustworthy Orchestration of Container Based Edge Computing Using Permissioned Blockchain [12]

Η εργασία αυτή πραγματεύεται την ανάγκη επεξεργασίας των δεδομένων που δημιουργούνται στις μέρες μας από IoT συσκευές, και πως έχει ωθήσει τόσο τον ακαδημαϊκό κλάδο όσο και την βιομηχανία να ερευνήσουν νέες εναλλακτικές αρχιτεκτονικές λύσεις για την υποστήριξη των νέων προκλήσεων που προκύπτουν. Ως αποτέλεσμα των ανωτέρω, έχει προκύψει το Edge Computing (EC) προκειμένου να αντιμετωπίσει τα ζητήματα αυτά, τοποθετώντας μέρος των cloud πόρων (όπως υπολογιστική ισχύς, αποθηκευτικός χώρος) πιο κοντά στην αιχμή του (edge) του δικτύου, το οποίο επιτρέπει γρηγορότερη και εξαρτώμενη από το περιεχόμενο ανάλυση και αποθήκευση δεδομένων. Ωστόσο, όσο οι EC υποδομές αναπτύσσονται, διαφορετικοί πάροχοι οι οποίοι δεν εμπιστεύονται απαραίτητα ο ένας τον άλλο πρέπει να συνεργαστούν προκειμένου να εξυπηρετούν διαφορετικές IoT συσκευές. Σε αυτό το πλαίσιο , οι EC υποδομές, οι IoT συσκευές και τα μεταδιδόμενα δεδομένα πρέπει να υπόκεινται σε έλεγχο ταυτότητας και προέλευσης, προκειμένου να αυξηθεί η εμπιστοσύνη και η απόδοση ευθυνών. Κάθε συσκευή στο δίκτυο πρέπει να αναγνωρίζεται και η προέλευση της δραστηριότητάς της να παρακολουθείται και να καταγράφεται. Σε αυτό το πλαίσιο, οι συγγραφείς προτείνουν μία αρχιτεκτονική που βασίζεται στο blockchain, τους

υπολογιστικούς πόρους στην αιχμή του δικτύου και τις τεχνολογίες των κοντέινερ (container), η οποία εφαρμόζει το W3C-PROV μοντέλο δεδομένων. Η προτεινόμενη αρχιτεκτονική προσφέρει νέες μορφές αλληλεπίδρασης μεταξύ των διάφορων ενδιαφερόμενων, και προσφέρει αξιόπιστες συναλλαγές οδηγώντας σε ένα νέο αποκεντρωμένο μοντέλο αλληλεπίδρασης για εφαρμογές που βασίζονται στο IoT. Συγκεκριμένα η λύση που παρουσιάζεται βασίζεται στο Hyperledger Fabric, και υποστηρίζει επαλήθευση ταυτότητας σε πραγματικό χρόνο, υψηλή διαθεσιμότητα και ανοχή σε σφάλματα. Παρουσιάζονται επίσης διάφορες επιπτώσεις εξαιτίας την χρήσης του blockchain, οι οποίες ομαδοποιούνται σε τρεις κατηγορίες: (1) το κατά πόσο η p2p αρχιτεκτονική επιτρέπει τη δημιουργία δια-επαγγελματικών υπηρεσιών χωρίς να βασίζεται σε ένα αξιόπιστο τρίτο μέλος, (2) τη διαφάνεια, καθώς το blockchain παρέχει τον τρόπο για τον λεπτομερή έλεγχο κάθε γεγονότος που λαμβάνει χώρα στο δίκτυο και τέλος (3) την ασφάλεια, όπου αν και υπάρχουν ακόμη ζητήματα προς επίλυση, η αποκεντρωμένη φύση του blockchain και οι μηχανισμοί κρυπτογράφησης που χρησιμοποιούνται από τις περισσότερες πλατφόρμες επιτρέπουν τον ανταγωνισμό με γνωστές ήδη υπάρχουσες τεχνολογίες σε πολλούς τομείς, όπως τα χρηματοοικονομικά και οι αυτοματισμοί.

2.2.7 Bubbles of Trust: A decentralized blockchain-based authentication system for IoT [13]

Οι συγγραφείς της εργασίας αυτής, εξετάζουν το σημαντικό ρόλο που παίζει το IoT στη καθημερινότητά μας. Πολλά από τα αντικείμενα που χρησιμοποιούμε είναι εφοδιασμένα με ηλεκτρονικές συσκευές και πρωτόκολλα προκειμένου να συνδέονται στο διαδίκτυο. Χαρακτηριστικά παραδείγματα σε οικιακό επίπεδο είναι τα έξυπνα τηλέφωνα (smartphones), οι έξυπνες τηλεοράσεις (smart TVs), έξυπνα ρολόγια (smartwatches), καθώς και σχεδόν όλες οι οικιακές ηλεκτρικές συσκευές οι οποίες αποκτούν, σταδιακά, λειτουργίες όπως η ενεργοποίηση και ο έλεγχος εξ αποστάσεως λευκών συσκευών, του φωτισμού και της θέρμανσης του σπιτιού, κάμερες παρακολούθησης του χώρου καθώς και πολλά ακόμη ενώ σε βιομηχανικό περιβάλλον τα παραδείγματα εκτείνονται από απλούς αισθητήρες, για παράδειγμα φωτός ή υγρασίας μέχρι ρομπότ και διάφορα εργαλεία προκειμένου να αυτοματοποιηθεί και να βελτιωθεί η γραμμή παραγωγής της επιχείρησης. Στο IoT συσκευές και αισθητήρες ανταλλάσσουν δεδομένα δίχως την ανάγκη ανθρώπινης παρέμβασης, συνεπώς λόγω της αυτονομίας τους αυτής, οι οντότητες αυτές πρέπει να αναγνωρίζουν η μία την άλλη και να μπορούν να επαληθεύουν τις ταυτότητες τους και να

διασφαλίζουν την ακεραιότητα των ανταλλασσόμενων δεδομένων τους. Το μέγεθος και τα χαρακτηριστικά όμως του IoT καθιστούν πρακτικά αδύνατη τη δημιουργία ενός κεντρικού συστήματος ελέγχου της ταυτότητας των οντοτήτων. Για να αντιμετωπίσουν αυτό το πρόβλημα, οι συγγραφείς προτείνουν ένα πρωτότυπο αποκεντρωμένο σύστημα που ονομάζεται φυσαλίδες εμπιστοσύνης (bubbles of trust), το οποίο διασφαλίζει αξιόπιστη αναγνώριση και έλεγχο της ταυτότητας των συσκευών. Επιπρόσθετα, προστατεύει την ακεραιότητα και τη διαθεσιμότητα των δεδομένων. Η προσέγγιση αυτή αξιοποιεί τα πλεονεκτήματα του blockchain και δημιουργεί ψηφιακές ασφαλείς ζώνες, όπου οι οντότητες μπορούν να αναγνωρίζουν και να εμπιστεύονται η μία την άλλη. Τα αποτελέσματα που προέκυψαν από την προσέγγιση αυτή αποδεικνύουν πως παρέχει την ικανότητα να ικανοποιεί τις απαιτήσεις ασφαλείας των IoT συσκευών, την αποδοτικότητα και το χαμηλό κόστος που απαιτείται.

2.2.8 Blockchains and Smart Contracts for the Internet of Things [14]

Στην εργασία αυτή, οι συγγραφείς εξετάζουν κατά πόσο το blockchain μπορεί να συνδυαστεί με το IoT. Συγκεκριμένα, εξετάζεται ο τρόπος λειτουργίας των blockchain αλλά και των smart contract τα οποία επιτρέπουν τη δημιουργία αυτοματισμών πολύ-επίπεδων διαδικασιών. Στη συνέχεια παρουσιάζεται πως μπορεί να λειτουργήσει ο τομέας του IoT σε συνδυασμό με το blockchain προκειμένου να διευκολύνεται η κοινή χρήση υπηρεσιών και πόρων, οδηγώντας στη δημιουργία μιας αγοράς υπηρεσιών ανάμεσα στις συσκευές. Επίσης παρουσιάζεται το πως μπορούν να αυτοματοποιηθούν, με ένα κρυπτογραφικά επαληθεύσιμο τρόπο πολλαπλές χρονοβόρες ροές εργασιών. Γίνεται αναφορά σε προβλήματα που πρέπει να εξεταστούν για τη χρήση του blockchain στο IoT, από την αξιοπιστία των συναλλαγών, στην αναμενόμενη αξία των ψηφιακών αγαθών που συναλλάσσονται στο δίκτυο. Στη συνέχεια εντοπίζονται τρόποι εφαρμογής και καταλήγουν αναγνωρίζοντας τις δυνατότητες που προκύπτουν από τον συνδυασμό του blockchain με το IoT.

2.2.9 Blockchain-based Automated Certificate Revocation for 5G IoT [15]

Η εργασία αυτή ασχολείται με την υποδομή δημόσιου κλειδιού (Public Key Infrastructure, PKI) και την εξάλειψη των κινδύνων ασφαλείας μέσω της διασφάλισης του ελέγχου ταυτότητας και της ακεραιότητας της επικοινωνίας χρησιμοποιώντας πιστοποιητικά

δημόσιου κλειδιού. Το πρόβλημα όμως σε αυτή τη περίπτωση είναι πως η αποθήκευση των πιστοποιητικών δημιουργεί σημαντικό πρόβλημα στους περιορισμένους πόρους των IoT συσκευών. Συνεπώς, οι συγγραφείς προτείνουν μία εφαρμογή πιστοποιητικών ελλειπτικής καμπύλης Qu Vanstone (Elliptic Curve Qu Vanstone, ECQV) η οποία έχει το μικρό μέγεθος που απαιτείται για τις IoT συσκευές. Επιπλέον, έγινε ενσωμάτωση των έξυπνων συμβολαίων του blockchain για τις διαδικασίες έκδοσης και διαχείρισης των πιστοποιητικών. Υλοποιήθηκε ένα σύστημα από έξυπνα συμβόλαια προκειμένου να εκδίδονται τα πιστοποιητικά και ένας μηχανισμός αυτόματης βαθμολόγησης για την αυτόματη ανάκληση αυτών.

2.2.10 Securing and Trustworthy Blockchain-based Multi-Tenant Cloud Computing [16]

Οι πολλαπλοί ένοικοι στις υπηρεσίες cloud computing είναι κρίσιμης σημασίας για τους cloud παρόχους, καθώς έχουν πολλαπλά οφέλη κατανέμοντας τους περιορισμένους πόρους που διαθέτουν σε όσο περισσότερους χρήστες μπορούν. Αυτή η λογική όμως, της τοποθέτησης πολλαπλών ενοίκων στους ίδιους πόρους προκαλεί προβλήματα, όπως η διαρροή δεδομένων, η οποία μπορεί να προκληθεί από κακόβουλους χρήστες. Επίσης οι κακόβουλοι χρήστες θα μπορούσαν να τροποποιήσουν τα αιτήματα και τις συναλλαγές τους προκειμένου να καταστήσουν δύσκολο να εντοπιστούν. Οι λόγοι αυτοί οδήγησαν τους συγγραφείς να χρησιμοποιήσουν το blockchain προκειμένου να υποστηρίξουν το cloud computing. Το blockchain παρέχει τη δυνατότητα να μπορεί κανείς να παρατηρεί και να ερευνά την κακόβουλη συμπεριφορά από τους χρήστες, παρέχοντας αδιαμφισβήτητη ακεραιότητα στην καταγραφή των συναλλαγών. Το ιστορικό του συνόλου των συναλλαγών καταγράφεται στο ledger , με χρονολογική σειρά, και δε μπορεί να τροποποιηθεί, συνεπώς είναι εύκολο κανείς να εντοπίσει τις ενέργειες του κάθε χρήστη. Επίσης, με τη χρησιμοποίηση smart contracts γίνεται εφικτή η προσομοίωση του μηχανισμού κατανομής, του ελέγχου πολιτικής και άλλων λειτουργιών μεταξύ του cloud παρόχου και των ενοικιαστών. Οι συγγραφείς καταλήγουν, υποστηρίζοντας πως το cloud computing μπορεί να εφαρμοστεί και να επωφεληθεί με τη χρήση του blockchain.

2.2.11 Integration of Blockchain and Cloud of Things: Architecture, Applications and Challenges [17]

Σε αυτή την εργασία οι συγγραφείς εξετάζουν τη δυνατότητα συνδυασμού του blockchain με το Cloud of Things (CoF), το οποίο αποτελεί ένα συνονθύλευμα του cloud computing και του IoT. Στο πλαίσιο αυτό το blockchain παρέχει καινοτόμες λύσεις για την αντιμετώπιση προκλήσεων του CoF, όσον αφορά την αποκέντρωση, το απόρρητο των δεδομένων και την ασφάλεια του δικτύου, ενώ το CoF προσφέρει λειτουργίες με ελαστικότητα και κλιμάκωση για την βελτίωση της αποδοτικότητας του blockchain. Στη συνέχεια, παρουσιάζουν ένα νέο παράδειγμα συνδυασμού του blockchain και του Cloud of Things, το οποίο αποκαλείται BcoT, και το οποίο θεωρείται ως ένα πολλά υποσχόμενο εργαλείο για μια πληθώρα εφαρμογών. Παρέχεται μία εις βάθος έρευνα των εφαρμογών BcoT με εκτενή συζήτηση σχετικά με τους τομείς χρήσης, καθώς και τις ευκαιρίες χρήσης του σε δίκτυα 5G αλλά και πέραν αυτών.

3 Blockchain

3.1 Εισαγωγή

Το blockchain είναι μια κατανεμημένη βάση δεδομένων, η οποία διατηρεί ένα αμετάβλητο ledger για όλες τις συναλλαγές και τα γεγονότα που πραγματοποιούνται στο δίκτυο. Οι συναλλαγές ομαδοποιούνται σε block, όπου το κάθε block αποτελείται από δύο μέρη:

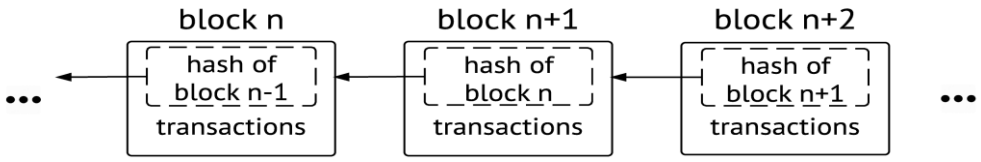
- Το σώμα (body), το οποίο περιέχει, αναλόγως την εφαρμογή, είτε ως απλό κείμενο είτε κρυπτογραφημένα, γεγονότα και συναλλαγές, όπως χρηματικές συναλλαγές, δεδομένα συστήματος, δεδομένα στον τομέα της υγείας κ.α.
- Την κεφαλίδα (header), η οποία περιέχει πληροφορίες για το ίδιο το block, όπως τη χρονική σφραγίδα (timestamp), κατακερματισμό των συναλλαγών (hashing) καθώς και κρυπτογραφημένο κατακερματισμό του προηγούμενου block, κρατώντας με αυτό τον τρόπο τη διεύθυνση του προηγούμενου block.

Με αυτόν τον τρόπο δημιουργείται μια αλυσίδα από τα υφιστάμενα block, τα οποία είναι συνδεδεμένα και διατεταγμένα. Κάθε κόμβος του δικτύου περιέχει ένα αντίγραφο του blockchain, ενώ όταν υπάρχει consensus ανάμεσα στους χρήστες σχετικά με το επόμενο block, αυτό προστίθεται στην αλυσίδα και όλα τα αντίγραφα ενημερώνονται και επικυρώνονται.

Δημιουργείται λοιπόν ένα ομότιμο (p2p) κατανεμημένο δίκτυο όπου τα μέλη μπορούν να αλληλοεπιδρούν μεταξύ τους δίχως την ανάγκη για μεσολάβηση ενός έμπιστου ενδιάμεσου, με εξακριβώσιμο και αξιόπιστο τρόπο. Το ρόλο αυτό αναλαμβάνει ένας consensus μηχανισμός, ο οποίος διαφοροποιείται ανάλογα το blockchain, και μέσω του μηχανισμού αυτού, επιβεβαιώνονται οι συναλλαγές που πραγματοποιούνται στο δίκτυο και περιλαμβάνονται στα επόμενα block που θα προστεθούν στο blockchain.

Με το blockchain, εφαρμογές που μέχρι πρότινος μπορούσαν να λειτουργούν μόνο μέσω ενός έμπιστου ενδιάμεσου, πλέον μπορούν να λειτουργούν με αποκεντρωμένο τρόπο, δίχως την ανάγκη για ύπαρξη κεντρικής αρχής., καθώς και να επιτύχουν το ίδιο επίπεδο λειτουργικότητας και βεβαιότητας για το αποτέλεσμα, ενώ η εκτεταμένη χρήση κρυπτογραφίας εξασφαλίζει την αυθεντικότητα σε όλες τις συναλλαγές του δικτύου. Σημαντικό χαρακτηριστικό του blockchain είναι και η ύπαρξη smart contracts, τα οποία

είναι αυτό-εκτελούμενα σύνολα εντολών που χρησιμοποιούνται για να καταστήσουν δυνατή την ύπαρξη κατανεμημένων, έντονα αυτοματοποιημένων ροών εργασιών. Στο σχήμα 3.1 που ακολουθεί μπορούμε να δούμε μια σχηματική αναπαράσταση ενός blockchain. Παρατηρούμε πως το κάθε block αποτελείται από δύο μέρη, τη κεφαλίδα και το σώμα, ενώ το κάθε block συνδέεται με το προηγούμενό του σχηματίζοντας μια αλυσίδα.



Σχήμα 3.1 - Παράδειγμα Blockchain

3.1.1 Ιστορική Αναδρομή

Η πρώτη ιστορικά εφαρμογή της τεχνολογίας του blockchain εισήχθη με το ψηφιακό κρυπτονόμισμα του Satoshi Nakamoto, το Bitcoin [1]. Το Bitcoin έδωσε λύση στο πρόβλημα των Βυζαντινών Στρατηγών, καθώς καταφέρνει να επιτευχθεί συναίνεση ανάμεσα σε πολλαπλά μέλη, τα οποία όμως δεν εμπιστεύονται το ένα το άλλο, μέσω ανταλλαγής μηνυμάτων, χωρίς όμως να είναι γνωστές οι προθέσεις του κάθε αποστολέα. Στην περίπτωση των κρυπτονομισμάτων, το υπολογιστικό πρόβλημα σχετίζεται με το πρόβλημα της διπλής σπατάλης, το οποίο αντιμετωπίζει το πως μπορούμε να εξακριβώσουμε εάν ένα ποσό των ψηφιακών χρημάτων δεν έχει ήδη δαπανηθεί, χωρίς την επιβεβαίωση ενός αξιόπιστου ενδιάμεσου, όπως για παράδειγμα μίας τράπεζας, η οποία κρατάει αρχείο με όλες τις συναλλαγές και τα υπόλοιπα των λογαριασμών.

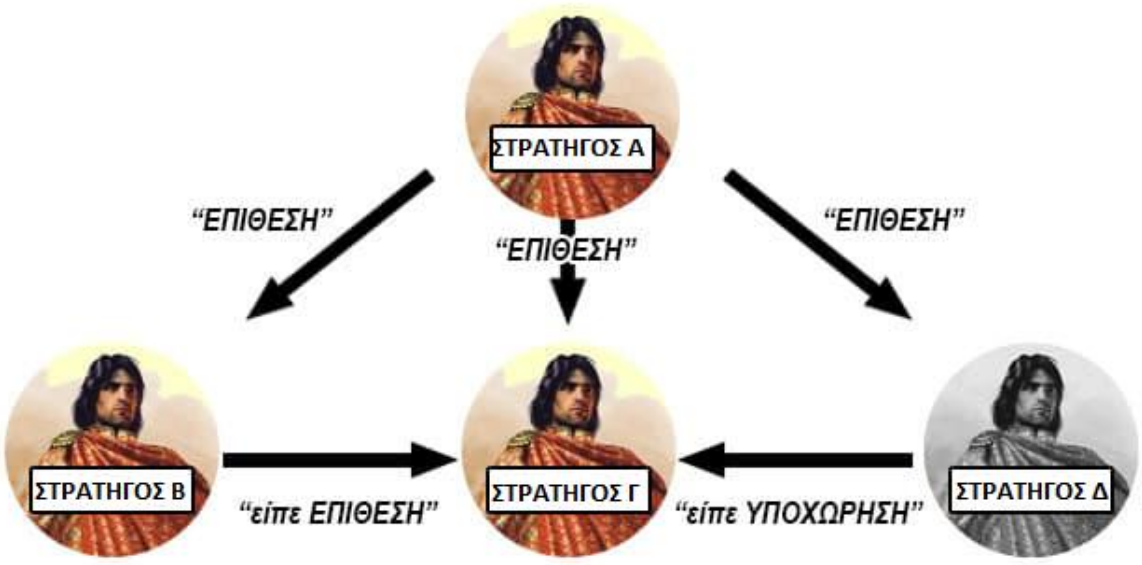
Αν και το blockchain προέκυψε ως εργαλείο για κρυπτονομίσματα, δεν είναι απαραίτητη η ύπαρξη τους για τη δημιουργία αποκεντρωμένων εφαρμογών που κάνουν χρήση του blockchain.

3.1.2 Το πρόβλημα των Βυζαντινών Στρατηγών

Ιστορικά, το πρόβλημα αυτό παρουσιάζεται από την ύπαρξη δύο ή περισσότερων Στρατηγών κατά τη διάρκεια της πολιορκίας μίας πόλης, από αντίθετες πλευρές και την

προσπάθεια τους να συνεργαστούν προκειμένου να πραγματοποιήσουν επίθεση και να καταλάβουν την εχθρική πόλη.

Συγκεκριμένα, αν ο Στρατηγός Α στείλει ένα μήνυμα που αναφέρει "Επίθεση αύριο το απόγευμα" δε μπορεί να γνωρίζει εάν ο Στρατηγός Β το έλαβε, συνεπώς υπάρχει περίπτωση να αποτύχει η επίθεση σε περίπτωση μη σωστής συνεννόησης. Από την άλλη πλευρά, ο Στρατηγός Β δε μπορεί να γνωρίζει εάν το μήνυμα που έλαβε είναι αυθεντικό ή αν έχει σταλεί από τον εχθρό προκειμένου να τον οδηγήσει σε παγίδα. Παρόλα αυτά, θα υποθέσει την αυθεντικότητα του μηνύματος και θα στείλει απάντηση στον Στρατηγό Α, δίχως όμως και πάλι να γνωρίζει εάν ο στρατηγός Α θα λάβει την απάντηση και κινδυνεύει να πραγματοποιήσει μόνος του την επίθεση. Ο Στρατηγός Α βρίσκεται τώρα στη θέση που βρισκόταν πριν ο Στρατηγός Β, καθώς δε γνωρίζει εάν το μήνυμα που θα λάβει είναι αυθεντικό ή αν έχει παραποιηθεί από τον εχθρό. Μπορεί να συνεχίσει την επικοινωνία στέλνοντας μήνυμα επιβεβαίωσης στο Στρατηγό Β, αλλά το πρόβλημα συνεχίζει να υπάρχει σε ολόκληρη τη διάρκεια της επικοινωνίας. Στο σχήμα 3.2 βλέπουμε ένα παράδειγμα του προβλήματος των Βυζαντινών Στρατηγών, όπου ο Στρατηγός Δ είναι προδότης και οδηγεί στην αποτυχία της επίθεσης.



Σχήμα 3.2 - Παράδειγμα του προβλήματος των Βυζαντινών Στρατηγών, όπου ο Στρατηγός Δ είναι προδότης

Στην περίπτωση των κρυπτονομισμάτων, όπως αναφέρθηκε και ανωτέρω, το πρόβλημα ανάγεται στο πρόβλημα της διπλής σπατάλης, το οποίο αντιμετωπίζει το πως μπορεί να επιβεβαιωθεί ότι ένα ποσό ψηφιακών χρημάτων δεν έχει ήδη δαπανηθεί δίχως να υπάρχει

ένας αξιόπιστος ενδιάμεσος, όπως ένα χρηματοπιστωτικό ίδρυμα, το οποίο κρατάει πλήρες ιστορικό των συναλλαγών των μελών.

3.1.3 Βασικός τρόπος λειτουργίας του Blockchain

Το blockchain μοιάζει με ένα ημερολόγιο του οποίου οι καταγραφές ομαδοποιούνται σε blocks με timestamps. Κάθε block αναγνωρίζεται από το hash του και περιέχει το hash του προηγούμενου block. Με τον τρόπο αυτό δημιουργείται μία αλυσίδα από block, ένα blockchain.

Προκειμένου να χρησιμοποιήσει κανείς το blockchain είναι απαραίτητο να δημιουργήσει αρχικά ένα p2p δίκτυο, με όλους τους κόμβους-χρήστες που επιθυμούν να κάνουν χρήση του συγκεκριμένου blockchain. Κάθε κόμβος του δικτύου λαμβάνει δύο κλειδιά:

- Ένα δημόσιο κλειδί (public), το οποίο χρησιμοποιείται από τους χρήστες για τη κρυπτογράφηση των μηνυμάτων προς ανταλλαγή.
- Ένα ιδιωτικό κλειδί (private), το οποίο επιτρέπει σε έναν κόμβο να διαβάσει τέτοιου είδους μηνύματα.

Συνεπώς χρησιμοποιούνται δύο κλειδιά, ένα για κρυπτογράφηση και ένα για αποκρυπτογράφηση. Πρακτικά, το ιδιωτικό κλειδί χρησιμοποιείται για την υπογραφή συναλλαγών του blockchain, όπως για παράδειγμα την έγκριση μιας συναλλαγής, ενώ το δημόσιο κλειδί χρησιμοποιείται ως δημόσια διεύθυνση. Μόνο ο χρήστης με το κατάλληλο ιδιωτικό κλειδί μπορεί να αποκρυπτογραφήσει μηνύματα που έχουν κρυπτογραφηθεί με το αντίστοιχο δημόσιο κλειδί. Η τεχνική αυτή ονομάζεται ασύμμετρη κρυπτογραφία. Όταν ένας κόμβος πραγματοποιεί μια συναλλαγή, την υπογράφει και τη μεταδίδει στους γειτονικούς κόμβους (one hop). Το γεγονός πως υπογράφει τη συναλλαγή με μοναδικό τρόπο, κάνοντας χρήση του ιδιωτικού κλειδιού του, πιστοποιεί την αυθεντικότητα της συναλλαγής, καθώς μόνο ο συγκεκριμένος χρήστης με το συγκεκριμένο κλειδί μπορεί να την υπογράψει και εγγυάται την ακεραιότητά του, καθώς αν υπάρχει κάποιο σφάλμα κατά τη διάρκεια της μετάδοσης των δεδομένων, δε θα μπορεί να αποκρυπτογραφηθεί το μήνυμα. Οι κόμβοι που λαμβάνουν την υπογεγραμμένη συναλλαγή, επιβεβαιώνουν την ορθότητα της πριν τη μεταδώσουν με τη σειρά τους σε άλλους κόμβους συνεισφέροντας με αυτόν τον τρόπο στην εξάπλωση στο δίκτυο. Στη συνέχεια, ανάλογα με τον consensus μηχανισμό που χρησιμοποιείται, εάν οι κόμβοι επιβεβαιώσουν την ορθότητα των

συναλλαγών και τη συνοχή με ό,τι έχει ήδη καταγραφεί στο δίκτυο, το block προστίθεται στο blockchain. Σε αντίθετη περίπτωση απορρίπτεται.

Οι κόμβοι θα πρέπει να συμφωνήσουν στις συναλλαγές και στη σειρά με την οποία αυτές θα πραγματοποιηθούν στα νέα block. Σε αντίθετη περίπτωση, τα ατομικά αντίγραφα του blockchain θα διαφοροποιηθούν και θα καταλήξουμε σε διακλαδώσεις (forks). Αυτό θα έχει ως αποτέλεσμα οι κόμβοι να έχουν διαφορετική οπτική της γενικής κατάστασης (world state) και το δίκτυο δε θα είναι σε θέση να διατηρήσει πλέον μια μοναδική, επίσημη χρονολογία των συμβάντων μέχρι η διακλάδωση να λυθεί. Συνεπώς, ο κατανεμημένος consensus μηχανισμός είναι απαραίτητος , ενώ ο τύπος του εξαρτάται από το blockchain. Σε ένα ιδανικό σενάριο, όλοι οι κόμβοι θα ψήφιζαν και θα επικρατούσε η άποψη της πλειοψηφίας. Σε ένα ανοιχτό (public) δίκτυο όμως, ο καθένας μπορεί να συμμετέχει, κάτι που θεωρητικά θα μπορούσε να έχει καταστροφικές συνέπειες σε περίπτωση που κάποιος προσπαθούσε να εκμεταλλευτεί το γεγονός αυτό και να επηρεάσει το τελικό αποτέλεσμα. Αυτό θα μπορούσε να γίνει εάν για παράδειγμα ο κακόβουλος χρήστης είχε πολλαπλές ψήφους στο σύστημα μέσω πολλαπλών εικονικών ταυτοτήτων και με αυτόν τον τρόπο, αν και μειοψηφία, θα μπορούσε να πάρει τον έλεγχο του δικτύου. Το κάθε blockchain όμως, έχει το δικό του τρόπο να αντιμετωπίζει τέτοιου είδους προβλήματα. Χαρακτηριστικό παράδειγμα είναι αυτό του Bitcoin όπου η διαδικασία επαλήθευσης είναι πολύ δαπανηρή για τους χρήστες. Με αυτόν τον τρόπο, η προσπάθεια να δημιουργήσει κάποιος πολλαπλές ταυτότητες δε θα είχε ουσιαστικό αποτέλεσμα, καθώς οι υπολογιστικές ανάγκες θα ήταν εξαιρετικά μεγάλες, ενώ οι διαθέσιμοι πόροι του χρήστη περιορισμένοι (οικονομικά ασύμφορο). Ο τρόπος αυτός ονομάζεται απόδειξη μόχθου (Proof of Work, PoW).

3.2 Κατηγορίες του Blockchain

Τα διάφορα blockchain μπορούν να ομαδοποιηθούν σε δύο κατηγορίες, τα δημόσια (public) και τα ιδιωτικά (private).

3.2.1 Public Blockchain

Σε ένα public blockchain, όπως το Bitcoin και το Ethereum, οποιοσδήποτε μπορεί να συμμετέχει και κάθε ένας που συμμετέχει μπορεί να διατηρεί την ανωνυμία του. Ο κάθε συμμετέχοντας έχει τη δυνατότητα πρόσβασης στις πληροφορίες του δικτύου και μπορεί να προσθέσει πληροφορίες σε αυτό μέσω της πραγματοποίησης συναλλαγών. Επίσης, μπορεί

να λάβει μέρος στη διαδικασία του consensus προκειμένου να καθοριστεί η έγκυρη κατάσταση του δικτύου.

Σε ένα public blockchain δεν υπάρχει εμπιστοσύνη ανάμεσα στους χρήστες και ο consensus μηχανισμός βασίζεται σε πρωτόκολλα που διασφαλίζουν τη συνοχή στην προσθήκη νέων block στο blockchain. Χαρακτηριστικά παραδείγματα τέτοιων consensus μηχανισμών είναι η απόδειξη μόχθου (Proof of Work, PoW) και η απόδειξη μερίσματος (Proof of Stake, PoS). Οι συναλλαγές των χρηστών είναι ανώνυμες και βασίζονται στη κρυπτογραφία.

3.2.2 Private Blockchain

Σε ένα private blockchain, δε μπορεί να συμμετέχει οποιοσδήποτε. Για να λάβει μέρος κάποιος θα πρέπει να προσκληθεί σε αυτό και να επικυρωθεί από αυτόν που δημιούργησε το δίκτυο ή από ένα σύνολο κανόνων που έχουν τεθεί στο δίκτυο. Ο επιχειρήσεις που θα χρησιμοποιήσουν ένα private blockchain, συνήθως θα δημιουργήσουν ένα δίκτυο με επιτρεπτότητα (permissioned). Ο μηχανισμός πρόσβασης μπορεί να διαφέρει ανάλογα το blockchain, για παράδειγμα οι υπάρχοντες χρήστες θα μπορούσαν να αποφασίζουν για τους μελλοντικούς συμμετέχοντες ή μια ρυθμιστική αρχή θα μπορούσε να εκδώσει άδειες συμμετοχής. Λόγω της διαφορετικής δομής των private blockchain, ο consensus μηχανισμός είναι συνήθως διαφορετικός σε σχέση με τα public, συνεπώς οι απαιτήσεις είναι μειωμένες σε θέμα χρόνου και ενέργειας.

Παράδειγμα private, permissioned blockchain, το οποίο χρησιμοποιήθηκε και στα πλαίσια της συγκεκριμένης εργασίας είναι το Hyperledger Fabric.

3.3 Smart Contracts

Τα έξυπνα συμβόλαια (smart contracts) είναι αποθηκευμένα scripts στο blockchain. Κάθε smart contract έχει μια μοναδική διεύθυνση και προκειμένου να το χρησιμοποιήσουμε καλούμε μία συναλλαγή προς αυτό. Στη συνέχεια, αυτό εκτελείται αυτόνομα με κάποιον συγκεκριμένο, προδιαγεγραμμένο τρόπο σε κάθε κόμβο του δικτύου με βάση τα δεδομένα που περιλαμβάνονται στη συναλλαγή που ενεργοποιεί το smart contract.

Πιο συγκεκριμένα, τα smart contracts μας επιτρέπουν να αυτοματοποιήσουμε και να πραγματοποιήσουμε συναλλαγές μεταξύ των οντοτήτων του δικτύου. Η συμπεριφορά τους

μπορεί να παρομοιαστεί με αυτή ενός αυτόνομου, πλήρως προβλέψιμου χρήστη. Μερικά από τα κύρια χαρακτηριστικά των smart contracts είναι τα ακόλουθα:

- Είναι ντετερμινιστικά. Επειδή ο κώδικας των smart contracts τρέχει σε πολλούς κόμβους ταυτόχρονα, είναι απαραίτητο το αποτέλεσμα να είναι ντετερμινιστικό και να μη περιέχει κανένα στοιχείο τυχαιότητας. Με την ίδια είσοδο όλοι οι κόμβοι δίνουν το ίδιο αποτέλεσμα.
- Είναι αμετάβλητα. Ο κώδικας τους δε μπορεί να τροποποιηθεί από τη στιγμή που θα αναπτυχθούν (deploy) στο blockchain. Σε συνδυασμό με το γεγονός πως κανείς δεν ελέγχει πλήρως τον κώδικα, και ότι όλοι μπορούν να δουν και να εξακριβώσουν τη λειτουργία του, δημιουργεί μία εμπιστοσύνη στα μέλη για την εκτέλεση του και στο να αποφασίσουν εάν επιθυμούν να τα χρησιμοποιήσουν.

3.4 Hyperledger Fabric (HLF)

Το Hyperledger Fabric είναι μια κατανεμημένη, ανοιχτού κώδικα, private πλατφόρμα, σχεδιασμένη για επιχειρηματικό περιεχόμενο που προσφέρει μερικές βασικές διαφοροποιήσεις σε σχέση με άλλες δημοφιλής blockchain πλατφόρμες. Στο κέντρο του δικτύου υπάρχει ένα κατανεμημένο ledger, το οποίο καταγράφει όλες τις συναλλαγές που συμβαίνουν στο δίκτυο. Δημιουργήθηκε υπό την αιγίδα του Linux Foundation και έχει γνωρίσει μεγάλη ανάπτυξη έκτοτε.

3.4.1 Ανάγκη Δημιουργίας του Hyperledger Fabric

Όπως αναφέρθηκε και νωρίτερα, το blockchain μοιάζει με ένα αμετάβλητο ημερολόγιο συναλλαγών που διατηρείται μεταξύ ενός κατανεμημένου δικτύου κόμβων χρηστών. Οι κόμβοι αυτοί διατηρούν ένα αντίγραφο του ημερολογίου, το οποίο έχει επαληθευτεί από ένα μηχανισμό συναίνεσης και οι συναλλαγές ομαδοποιούνται σε block που περιλαμβάνουν ένα hash το οποίο συνδέει το κάθε block με το προηγούμενό του.

Καθώς blockchain όπως το Bitcoin και το Ethereum αποκτούσαν ολοένα και μεγαλύτερο κοινό, δημιουργήθηκαν ανάγκες για χρήση κατανεμημένης τεχνολογίας και σε πιο επιχειρηματικό περιβάλλον. Σε αυτές τις περιπτώσεις όμως, απαιτήσεις σε συγκεκριμένα χαρακτηριστικά, όπως η ταχύτητα και η γνώση της ταυτότητας των συμμετεχόντων του δικτύου κατέστησε τα υπάρχοντα public blockchain μη ικανά να ανταποκριθούν πλήρως

στις νέες αυτές προκλήσεις. Οι απαιτήσεις που τέθηκαν για το επιχειρηματικό περιβάλλον είναι οι ακόλουθες:

- Οι συμμετέχοντες θα πρέπει να μπορούν να αναγνωριστούν, η ταυτότητα τους δηλαδή δεν είναι πλέον μυστική.
- Το δίκτυο είναι πλέον ιδιωτικό, δε μπορεί να λάβει μέρος σε αυτό οποιοσδήποτε επιθυμεί.
- Ο υψηλός ρυθμός εκτέλεσης των συναλλαγών έγινε απαραίτητος.
- Η κατά το δυνατό ελάχιστη καθυστέρηση της επαλήθευσης των συναλλαγών.
- Η ιδιωτικότητα και η εμπιστευτικότητα των συναλλαγών και των δεδομένων μεταξύ των επιχειρηματικών συναλλαγών.

Ενώ πολλά από τα υπάρχοντα blockchain τροποποιήθηκαν προκειμένου να μπορούν να ανταποκριθούν στις νέες αυτές απαιτήσεις, το Hyperledger Fabric σχεδιάστηκε εξ αρχής με γνώμονα τις αρχές αυτές, γεγονός που επηρέασε τις αρχιτεκτονικές επιλογές σχεδίασης του.

3.4.2 Η Δομή του Hyperledger Fabric

Η δομή του Hyperledger Fabric είναι υπεύθυνη για τη μείωση του κόστους, του χρόνου και του ρίσκου σχετικά με τον διαμοιρασμό και την επεξεργασία των πληροφοριών των χρηστών σε σχέση με τα παλαιότερα πρότυπα που χρησιμοποιούνταν. Μερικά από τα βασικά στοιχεία του HLF είναι τα ακόλουθα:

- Η αρχιτεκτονική του Hyperledger Fabric είναι αρθρωτή (modular) και παραμετροποιήσιμη (configurable). Το γεγονός αυτό ενθαρρύνει τη χρήση του σε ένα ευρύ σύνολο εφαρμογών, όπως ο τομέας της υγείας, ο τραπεζικός τομέας, τα οικονομικά κ.α. καθώς μπορεί να τροποποιηθεί σύμφωνα με τις ανάγκες και τις προτεραιότητες που προκύπτουν σε κάθε περίπτωση χωριστά.
- Private/Permissioned blockchain. Δε μπορεί να συμμετέχει οποιοσδήποτε στο blockchain, ενώ οι συμμετέχοντες έχουν κάποιον κοινό σκοπό αν και δεν εμπιστεύονται απαραίτητα ο ένας τον άλλο. Ορίζονται περιορισμοί στους συμμετέχοντες σχετικοί με τα δικαιώματα πρόσβασης και γραφής.

- Δεν απαιτείται η χρήση χρονοβόρων και ενεργοβόρων διαδικασιών για την επίτευξη του consensus, όπως του PoW. Με αυτόν τον τρόπο το κόστος χρήσης του blockchain μειώνεται σε μεγάλο βαθμό.
- Pluggable consensus protocols. Η διαφοροποίηση αυτή επιτρέπει στο Hyperledger Fabric να παραμετροποιείται προκειμένου να είναι αποδοτικό ανεξαρτήτως των αναγκών που προκύπτουν από την εκάστοτε περίπτωση χρήσης. Χαρακτηριστικά, μπορεί να χρησιμοποιεί πρωτόκολλα που να μην αντιμετωπίζουν το πρόβλημα των Βυζαντινών Στρατηγών (Byzantine Fault Tolerant, BFT), προκειμένου να αυξήσει την αποδοτικότητα του, με γνώμονα όμως την ύπαρξη εμπιστοσύνης ανάμεσα στους χρήστες. Παράδειγμα τέτοιας χρήσης μπορεί να είναι ένα blockchain το οποίο χρησιμοποιείται στο εσωτερικό μια εταιρείας. Σε μία τέτοια περίπτωση είναι σημαντικό το consensus protocol να είναι ανθεκτικό σε σφάλματα χρηστών (Crash Fault Tolerant, CFT).
- Το ledger του Hyperledger Fabric αποτελείται από δύο κομμάτια:
    - Την Παγκόσμια Κατάσταση (World State), όπου περιγράφει τη κατάσταση του ledger για κάποια χρονική στιγμή.
    - Το Ημερολόγιο Συναλλαγών (Transaction Log) όπου καταγράφει όλες τις συναλλαγές που οδηγούν σε κάθε κατάσταση στο World State.
- Ιδιωτικότητα και Εμπιστευτικότητα. Το Fabric, ως permissioned blockchain, επιτρέπει τη προσέγγιση της εμπιστευτικότητας μέσω:
    - Της αρχιτεκτονικής των καναλιών επικοινωνίας.
    - Δυνατότητα επιλογής ορισμού δεδομένων ως ιδιωτικά, όπου μπορεί να καθοριστεί το υποσύνολο των χρηστών που θα έχουν πρόσβαση σε αυτά.

Αναλυτικά, στα κανάλια, τα οποία μπορεί κανείς να σκεφτεί ως διαύλους επικοινωνίας μεταξύ δύο ή περισσότερων χρηστών, οι συμμετέχοντες δημιουργούν ένα υποδίκτυο όπου το κάθε μέλος έχει πρόσβαση σε ένα συγκεκριμένο σύνολο συναλλαγών. Με αυτό τον τρόπο μόνο οι συμμετέχοντες στο συγκεκριμένο κανάλι έχουν πρόσβαση στα smart contracts που θα χρησιμοποιηθούν και τα δεδομένα που θα γίνουν αντικείμενο συναλλαγών στο υποδίκτυο.

Από την άλλη πλευρά, τα ιδιωτικά δεδομένα επιτρέπουν σε ορισμένα μέλη του καναλιού να έχουν πρόσβαση δίχως την ανάγκη δημιουργίας ξεχωριστού καναλιού, δημιουργώντας με αυτό τον τρόπο τα ίδια επίπεδα ιδιωτικότητας και εμπιστευτικότητας με πριν δίχως όμως την ανάγκη δημιουργίας ξεχωριστού καναλιού επικοινωνίας και τα κόστη που αυτό επιφέρει.

3.4.3 Smart Contracts στο Hyperledger Fabric

Τα smart contracts στο Fabric ονομάζονται chaincode. Χαρακτηριστικά τους στοιχεία είναι η ικανότητα τους να τρέχουν παράλληλα περισσότερα του ενός στο δίκτυο καθώς και η δυνατότητα να υποβάλλονται στο δίκτυο δυναμικά. Με τα chaincodes το blockchain αποκτά πρόσβαση στο ledger.

Το κύριο χαρακτηριστικό όμως που ξεχωρίζει τη προσέγγιση του Fabric σε σχέση με τις περισσότερες προσεγγίσεις στα smart contracts είναι η διαφοροποίηση του τρόπου που λειτουργούν. Συγκεκριμένα, τα περισσότερα blockchain, όπως το Ethereum, χρησιμοποιούν μια order-execute λογική, στην οποία το consensus protocol επαληθεύει τις συναλλαγές και τις μεταδίδει στους χρήστες οι οποίοι με τη σειρά τους τις εκτελούν σειριακά, ενώ διέπονται από ντετερμινιστικό χαρακτήρα. Για την επίτευξη του ντετερμινισμού όμως, χρησιμοποιούνται γλώσσες ειδικού σκοπού προκειμένου να εξαλειφθούν τα μη ντετερμινιστικά γλωσσικά στοιχεία. Η σειριακή εκτέλεση όμως, δημιουργεί προβλήματα απόδοσης αλλά και κλιμάκωσης του συστήματος.

Αντίθετα, στο Hyperledger Fabric για τις συναλλαγές ακολουθείται μια execute-order-validate λογική. Ο διαχωρισμός σε 3 στάδια της ροής της συναλλαγής αντιμετωπίζει προβλήματα όπως αυτά της απόδοσης του συστήματος, της κλιμάκωσης, της εμπιστευτικότητας κ.α. Η ροή της συναλλαγής μπορεί να αναλυθεί ως εξής:

- Execute: Στο στάδιο αυτό εκτελείται μία συναλλαγή και ελέγχεται ως προς την ορθότητα της.
- Order: Η λειτουργία διάταξης των συναλλαγών εκτελείται από την Ordering Service του συστήματος. Οι συναλλαγές που έχουν υποβληθεί από το στάδιο του execute στη συνέχεια διατάσσονται και κατανέμονται σε block. Μπορούν να ληφθούν παράλληλα πολλές συναλλαγές από διαφορετικούς χρήστες. Οι κόμβοι του ordering service λειτουργούν συλλογικά και συντονισμένα προκειμένου να δημιουργήσουν καλά καθορισμένες ακολουθίες συναλλαγών και να τις οργανώσουν σε block. Τα

block αυτά αποθηκεύονται στο ledger του orderer και στη κατανέμονται στους peers που ανήκουν στο συγκεκριμένο κανάλι επικοινωνίας. Στο Fabric η υπηρεσία αυτή είναι κεντρική, αλλά μπορεί να αποτελείται από πολλούς κόμβους που δουλεύουν παράλληλα με σκοπό την ολοκλήρωση της διαδικασίας. Αυτός είναι και ο λόγος όπου το ordering service έχει σχεδιαστεί προκειμένου να μπορεί να είναι αποκεντρωμένο. Το Fabric υποστηρίζει 3 διαφορετικές λειτουργίες στο κομμάτι αυτό, το Solo, το Kafka καθώς και το Raft.

- Validate: Στο στάδιο αυτό επαληθεύονται οι συναλλαγές με βάση κάποια συγκεκριμένη πολιτική πριν τις υποβάλλουμε στο blockchain. Η πολιτική που εφαρμόζεται σε κάθε εφαρμογή καθορίζει πόσοι κόμβοι ή ποιοι από αυτούς πρέπει να εγγυηθούν για την ορθή λειτουργία του chaincode. Συνεπώς κάθε συναλλαγή χρειάζεται να εκτελεστεί από ένα υποσύνολο των κόμβων.

Η διαδικασία αυτή εξαλείφει τα μη ντετερμινιστικά στοιχεία πριν καν φτάσουν στη διαδικασία του Ordering, με αποτέλεσμα να καθίσταται εφικτός ο προγραμματισμός του chaincode σε γλώσσες προγραμματισμού γενικού σκοπού, όπως η Go, Java, Node.js.

4. Σχεδιασμός και υλοποίηση του Συστήματος

Στο κεφάλαιο αυτό θα παρουσιαστούν λεπτομέρειες που αφορούν τον σχεδιασμό και την υλοποίηση της εφαρμογής μας που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας. Στη συνέχεια θα γίνει περιγραφή της δομής του chaincode καθώς και των στοιχείων (assets) που περιέχονται στον κώδικα μας.

4.1 Σχεδιασμός και Αρχιτεκτονική

Η εφαρμογή μας βασίζεται στο Hyperledger Fabric, το οποίο είναι ένα private, permissioned blockchain, συνεπώς πρόκειται για μία κατανεμημένη εφαρμογή (Decentralized Application, DApp). Για την υλοποίηση και τη διαχείριση της έχουμε εκμεταλλευτεί τη χρήση chaincode, όπου εκεί υπάρχουν όλες οι συναρτήσεις-λειτουργίες της, ενώ η συγγραφή έγινε σε γλώσσα προγραμματισμού Go. Τα δεδομένα και οι συναλλαγές είναι αποθηκευμένα στο κατανεμημένο ledger, ενώ ο χρήστης, ο οποίος πρέπει να είναι εγκεκριμένος για να έχει πρόσβαση στο δίκτυο, προκειμένου να εκτελέσει τις λειτουργίες που επιθυμεί χρησιμοποιεί την γραμμή εντολών (Command Line Interface, CLI). Για την ανάπτυξη του chaincode χρησιμοποιήθηκε το fabric-contract-api-go [18] μέσω του οποίου η εφαρμογή επικοινωνεί με το δίκτυο. Πιο αναλυτικά, το δίκτυο μας αποτελείται από τα εξής χαρακτηριστικά:

- Έναν Orderer (Organization), ο οποίος είναι υπεύθυνος για το ordering service (Raft)
- Δύο Οργανισμούς (Organizations), όπου ο κάθε ένας από αυτούς έχει έναν χρήστη (peer).
- Ένα κανάλι επικοινωνίας (channel), στο οποίο συνδέονται οι οργανισμοί, ενώ παράλληλα εκεί γίνεται deploy το chaincode.

Για τη δημιουργία του δικτύου, αρχικά έγινε εγκατάσταση όλων των προ-απαιτούμενων [19] που χρειάζεται το Hyperledger Fabric καθώς και των αρχείων εγκατάστασης [20] που περιγράφονται στην επίσημη σελίδα. Μετά την ολοκλήρωση των διαδικασιών αυτών, είμαστε σε θέση να ξεκινήσουμε να υλοποιούμε το δίκτυο μας. Για τις ανάγκες της εργασίες το δίκτυο τρέχει τοπικά σε προσωπικό υπολογιστή και γίνεται προσομοίωση χρήσης από τους δύο χρήστες (peers) που αναφέρουμε ανωτέρω.

Δημιουργία δικτύου

Για τη δημιουργία των συστατικών (components) του δικτύου δημιουργήθηκε ένα βασικό αυτοματοποιημένο σενάριο (script). Το script αυτό κάνει χρήση των απαραίτητων εργαλείων που εγκαταστάθηκαν κατά την ανωτέρω διαδικασία και των αντίστοιχων αρχείων εισόδου τα οποία δημιουργήθηκαν για τις ανάγκες της προσομοίωσης. Στο τέλος της εργασίας παρατίθεται σύνδεσμος που οδηγεί σε όλα τα αρχεία από τα οποία αποτελείται το πρόγραμμα μας.

Αναλυτική διαδικασία

Για τους organizations του δικτύου πρέπει να δημιουργηθούν όλα τα απαραίτητα ψηφιακά πιστοποιητικά και τα private key, τα οποία ονομάζουμε crypto material. Τα στοιχεία αυτά είναι απαραίτητα καθώς με αυτόν τον τρόπο ορίζεται κάθε organization στο δίκτυο, και στο HLF το οποίο είναι permissioned blockchain κάθε κόμβος πρέπει να χρησιμοποιεί τα πιστοποιητικά του προκειμένου να υπογράφει και να επικυρώνει τις πράξεις του.

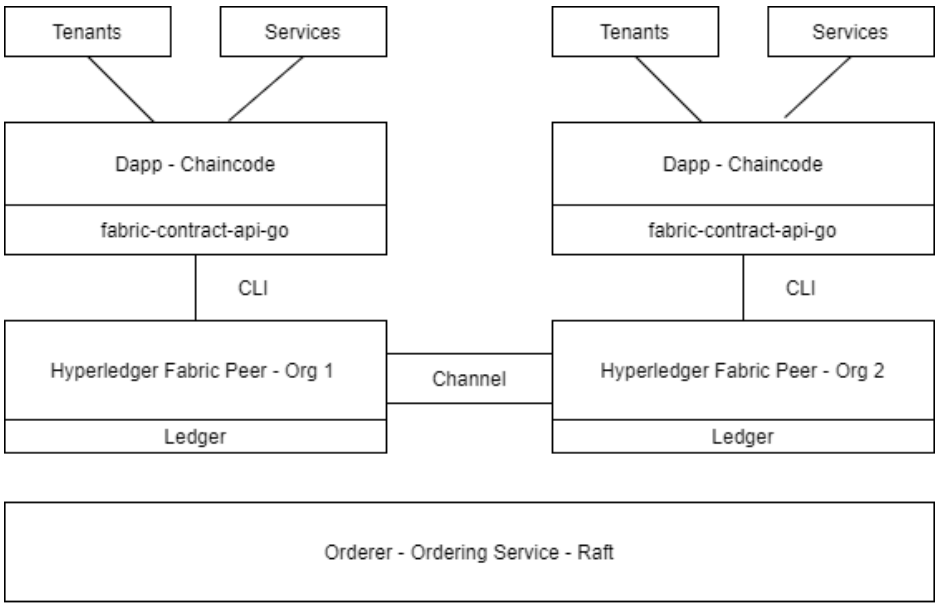
Μπορούμε να χρησιμοποιήσουμε δύο τρόπους για να το πραγματοποιήσουμε αυτό. Είτε με το Cryptogen Tool [21], το οποίο δέχεται ως είσοδο ένα σύνολο αρχείων διαμόρφωσης και παράγει τα απαραίτητα crypto material, είτε μέσω της αρχής έκδοσης πιστοποιητικών του Fabric (Certificate Authority, CA) [22]. Τα πιστοποιητικά και τα κλειδιά που δημιουργούνται, υπογράφονται από τα CAs δημιουργώντας ένα ιστορικό εμπιστοσύνης (root of trust) για κάθε organization. Το σενάριο δημιουργίας (script) που χρησιμοποιούμε προκειμένου να δημιουργήσουμε το δίκτυο μας, χρησιμοποιεί το Docker Compose [23] προκειμένου να δημιουργηθούν τρεις CAs, μία για κάθε organization. Μόλις δημιουργηθούν τα crypto material για τους organizations, πρέπει να δημιουργηθεί το αρχικό block (genesis block) του orderer στο κανάλι του συστήματος (system channel). Το block αυτό είναι απαραίτητο προκειμένου να λειτουργήσουν οι orderer κόμβοι, στην περίπτωση μας ένας, και να δημιουργηθούν κανάλια εφαρμογών (application channels).

Για τη δημιουργία του genesis block, χρησιμοποιείται το εργαλείο Configtxgen [24]. Αυτό έχει ως είσοδο ένα αρχείο με τη μορφή "configtx.yaml" το οποίο περιέχει τους απαραίτητους ορισμούς για το δίκτυο. Το genesis block ορίζεται με βάση το προφίλ που περιέχεται στο αρχείο αυτό. Επίσης, ορίζει και μία κοινοπραξία (consortium) η οποία αναφέρει ποιοι organizations αναγνωρίζονται από το δίκτυο. Κάθε μέλος του δικτύου αποκτά δικό του αρχείο στο Membership Service Provider (MSP) [25], το οποίο

χρησιμοποιείται για να επιτρέπει στους κόμβους και στους χρήστες να αναγνωρίζονται ως μέλη του δικτύου.

Αφού δημιουργηθούν τα crypto material και το genesis block μπορούμε να δημιουργήσουμε το κανάλι της εφαρμογής. Για τη δημιουργία του καναλιού χρησιμοποιείται το Configtxgen εργαλείο ξανά, προκειμένου να δημιουργήσει τις απαραίτητες συναλλαγές δημιουργίας και τις ενημερώσεις στους peers. Στη συνέχεια, μπορούμε να εγκαταστήσουμε και να εκκινήσουμε το chaincode μας στο κανάλι. Αρχικά το chaincode συσκευάζεται (packaged) και εγκαθίσταται στους επιλεγμένους peers. Στη συνέχεια κάθε peer organization πρέπει να εγκρίνει τον κώδικα. Αφού ο κώδικας εγκριθεί από όλους τους peer organizations, υποβάλλεται (committed) στο κανάλι και είναι πλέον έτοιμος να χρησιμοποιηθεί. Για να εκκινήσει καλούμε την συνάρτηση εκκίνησης.

Τέλος, η αλληλεπίδραση με το chaincode γίνεται μέσω του περιβάλλοντος της γραμμής εντολών (Command Line Interface, CLI) ενώ μέσω μεταβλητών περιβάλλοντος μπορούμε να επιλέξουμε αν θέλουμε να χρησιμοποιούμε το CLI ως ο peer από τον Organization 1 ή από το Organization 2, αφού το πρόγραμμα μας τρέχει τοπικά.



Σχήμα 4.1 - Αρχιτεκτονική Εφαρμογής

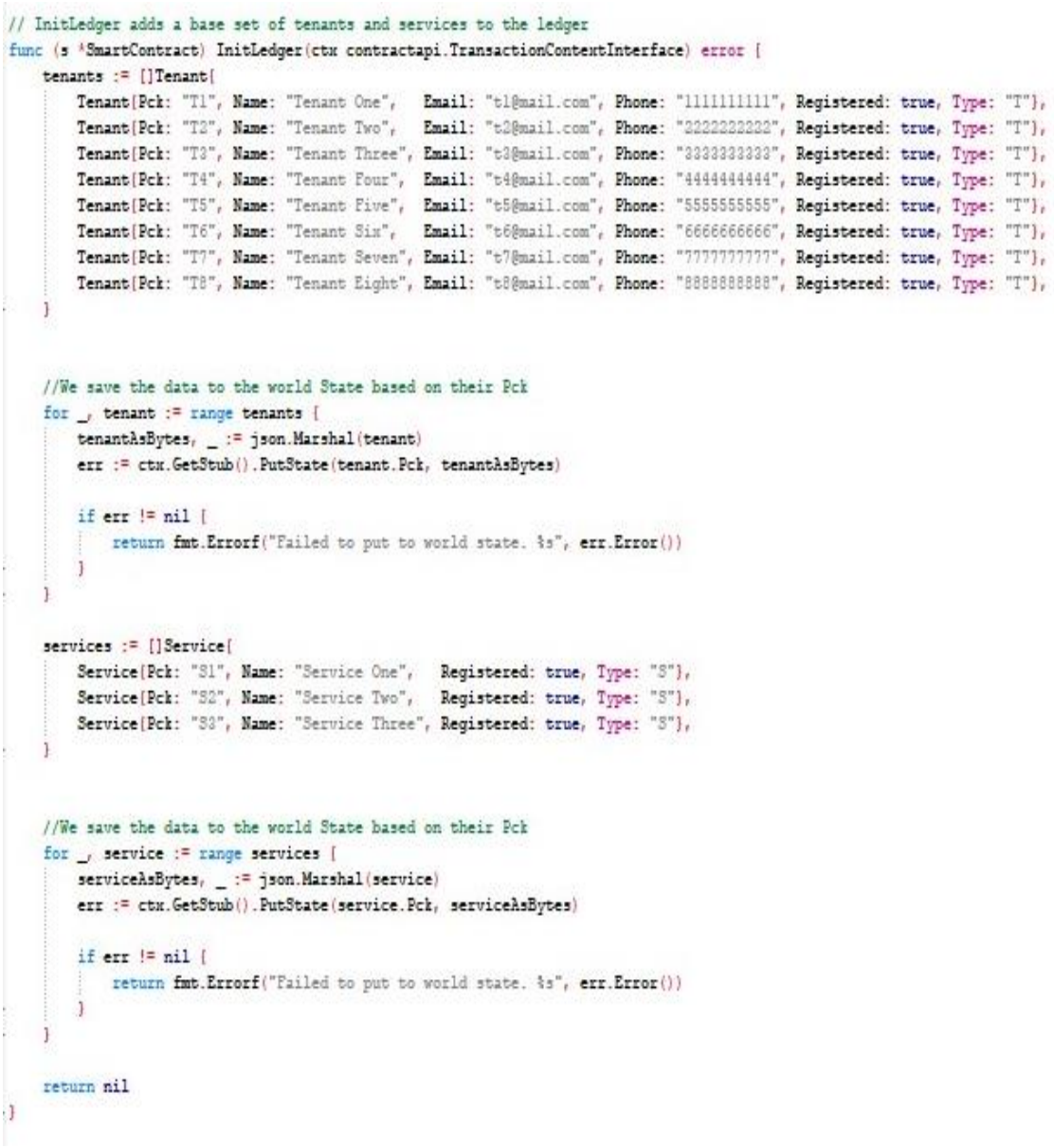
Στο σχήμα 4.1 μπορούμε να δούμε την αναπαράσταση της αρχιτεκτονικής της εφαρμογής μας, τον τρόπο που συνδέονται μεταξύ τους οι δύο peers, μέσω του καναλιού επικοινωνίας, αλλά και πως, μέσω του CLI και κάνοντας χρήση του fabric-contract-api-go, αποκτούν πρόσβαση στο chaincode προκειμένου να διαχειριστούν τους tenants και τα services. Σε

πραγματικές συνθήκες, τη θέση των organizations και των peers αυτών, θα μπορούσε να χρησιμοποιεί ο εκάστοτε cloud provider, ο οποίος μέσω του συστήματος αυτού μπορεί να διαχειριστεί τις υπηρεσίες που παρέχει και τους χρήστες που κάνουν χρήση των παρεχόμενων υπηρεσιών. Συγκεκριμένα, ο cloud provider θα αναλάμβανε το ρόλο του organization, ενώ οι peers του θα ήταν τα πολλαπλά instances την εφαρμογής, για παράδειγμα ένα σε κάθε server σε κάποιο σύμπλεγμα διακομιστών (server cluster), τα οποία θα εκτελούνταν παράλληλα και θα σχημάτιζαν το blockchain δίνοντας με αυτόν τον τρόπο τη δυνατότητα να εκμεταλλευτούν τα πλεονεκτήματα του όπως έχει περιγραφεί ανωτέρω.

4.2 Δομή του Chaincode

Στο σημείο αυτό θα παρουσιάσουμε τη γενική δομή του chaincode μας. Συγκεκριμένα, ο κώδικας μας αποτελείται από τριών ειδών συναρτήσεις, όπου κάθε συνάρτηση αποτελεί μία λειτουργία του συστήματος. Αναλυτική περιγραφή των επιμέρους συναρτήσεων θα γίνει στο κεφάλαιο 5, όπου θα παρουσιαστούν με τη μορφή διαγραμμάτων ακολουθίας μηνυμάτων και θα εξηγηθεί ο τρόπος λειτουργίας τους. Οι συναρτήσεις χωρίζονται στις εξής κατηγορίες:

- Συναρτήσεις Εκκίνησης (Initiation). Εδώ έχουμε μια συνάρτηση την οποία καλούμε προκειμένου να εκκινήσουμε την εκτέλεση του chaincode. Παράλληλα, κατά την εκκίνηση γίνεται εγγραφή κάποιων χρηστών και υπηρεσιών στο ledger για σκοπούς ευκολίας επίδειξης της λειτουργίας του συστήματος. Μετά την πρώτη κλήση της συνάρτησης αυτής, δε προσφέρει κάποια άλλη λειτουργικότητα στο πρόγραμμα μας και δε χρησιμοποιείται ξανά. Στο σχήμα 4.2 μπορούμε να δούμε ένα απόσπασμα από το chaincode όπου παρουσιάζει μία ενδεικτική συνάρτηση εκκίνησης. Στη συνέχεια της εργασίας δε θα ασχοληθούμε περαιτέρω με τη συνάρτηση εκκίνησης.



```go
// InitLedger adds a base set of tenants and services to the ledger
func (s *SmartContract) InitLedger(ctx contractapi.TransactionContextInterface) error {
    tenants := []Tenant{
        Tenant{Pck: "T1", Name: "Tenant One",   Email: "t1@mail.com", Phone: "1111111111", Registered: true, Type: "T"},
        Tenant{Pck: "T2", Name: "Tenant Two",   Email: "t2@mail.com", Phone: "2222222222", Registered: true, Type: "T"},
        Tenant{Pck: "T3", Name: "Tenant Three", Email: "t3@mail.com", Phone: "3333333333", Registered: true, Type: "T"},
        Tenant{Pck: "T4", Name: "Tenant Four",  Email: "t4@mail.com", Phone: "4444444444", Registered: true, Type: "T"},
        Tenant{Pck: "T5", Name: "Tenant Five",  Email: "t5@mail.com", Phone: "5555555555", Registered: true, Type: "T"},
        Tenant{Pck: "T6", Name: "Tenant Six",   Email: "t6@mail.com", Phone: "6666666666", Registered: true, Type: "T"},
        Tenant{Pck: "T7", Name: "Tenant Seven", Email: "t7@mail.com", Phone: "7777777777", Registered: true, Type: "T"},
        Tenant{Pck: "T8", Name: "Tenant Eight", Email: "t8@mail.com", Phone: "8888888888", Registered: true, Type: "T"},
    }

    //We save the data to the world State based on their Pck
    for _, tenant := range tenants {
        tenantAsBytes, _ := json.Marshal(tenant)
        err := ctx.GetStub().PutState(tenant.Pck, tenantAsBytes)

        if err != nil {
            return fmt.Errorf("Failed to put to world state. %s", err.Error())
        }
    }

    services := []Service{
        Service{Pck: "S1", Name: "Service One",   Registered: true, Type: "S"},
        Service{Pck: "S2", Name: "Service Two",   Registered: true, Type: "S"},
        Service{Pck: "S3", Name: "Service Three", Registered: true, Type: "S"},
    }


    //We save the data to the world State based on their Pck
    for _, service := range services {
        serviceAsBytes, _ := json.Marshal(service)
        err := ctx.GetStub().PutState(service.Pck, serviceAsBytes)

        if err != nil {
            return fmt.Errorf("Failed to put to world state. %s", err.Error())
        }
    }

    return nil
}
```

Σχήμα 4.2 - Απόσπασμα κώδικα που παρουσιάζει τη συνάρτηση εκκίνησης.

- Συναρτήσεις Ερώτησης (Query). Εδώ έχουμε συναρτήσεις που έχουν ως σκοπό να διαβάσουν δεδομένα από το ledger. Για παράδειγμα, η συνάρτηση αναζήτησης ενός tenant είναι συνάρτηση ερώτησης. Στο σχήμα 4.3 παρουσιάζεται μία ενδεικτική συνάρτηση ερώτησης. Δέχεται ως όρισμα το κλειδί με το οποίο αποθηκεύουμε τους tenants στο ledger και επιστρέφει τα δεδομένα του, αν υπάρχει, σε αντίθετη περίπτωση επιστρέφει μήνυμα σφάλματος.

```go
//IsTenant returns the tenant stored in the world state with given Pck (Key)
func (s *SmartContract)IsTenant(ctx contractapi.TransactionContextInterface, pck string) (*Tenant, error) {
    //getting the data from world state based on the pck
    tenantAsBytes, err := ctx.GetStub().GetState(pck)

    if err != nil {
        return nil, fmt.Errorf("Failed to read from world state. %s", err.Error())
    }

    if tenantAsBytes == nil {
        return nil, fmt.Errorf("%s does not exist", pck)
    }

    tenant := new(Tenant)
    _ = json.Unmarshal(tenantAsBytes, tenant)

    return tenant, nil
}
```

Σχήμα 4.3 - Απόσπασμα κώδικα που παρουσιάζει μία συνάρτηση ερώτησης.

- Συναρτήσεις Ενημέρωσης (Update). Οι συναρτήσεις αυτές έχουν ως σκοπό την εγγραφή δεδομένων στο ledger. Παράδειγμα μίας τέτοιας συνάρτησης είναι η συνάρτηση εγγραφής ενός tenant στο ledger. Η ονομασία επιλέχθηκε καθώς οι συναρτήσεις αυτές ενημερώνουν την παγκόσμια κατάσταση (world state) εγγράφοντας νέα δεδομένα στο ledger. Παράδειγμα μίας τέτοιας συνάρτησης είναι η συνάρτηση εγγραφής ενός tenant όπως μπορούμε να δούμε και στο σχήμα 4.4.

```go
//Enroll adds a new tenant to the world state with given details
func (s *SmartContract) Enroll(ctx contractapi.TransactionContextInterface, pck string, name string, email string, phone string) error {
    //matching the given data to the tenant fields
    tenant := Tenant{
        Pck:        pck,
        Name:       name,
        Email:      email,
        Phone:      phone,
        Registered: true,
        Type:       "T",
    }

    //storing to the world state based on the pck
    tenantAsBytes, _ := json.Marshal(tenant)
    return ctx.GetStub().PutState(pck, tenantAsBytes)
}
```

Σχήμα 4.4 - Απόσπασμα κώδικα που παρουσιάζει μία συνάρτηση ενημέρωσης.

Επιπρόσθετα, υπάρχουν πολλές συναρτήσεις που είναι συνδυαστικές, όπως η συνάρτηση ενημέρωσης των στοιχείων ενός tenant, όπου στην αρχή λειτουργεί ως query αναζητώντας τον tenant στο ledger ενώ στη συνέχεια περνάει τα νέα στοιχεία, τα οποία όμως δεν εγγράφονται πάνω στα παλιά, αλλά δημιουργούν μία νέα εγγραφή στο ledger, κρατώντας με αυτό τον τρόπο ένα αδιάλλακτο ιστορικό όλων των συναλλαγών. Αν και συνδυαστικές, τις συναρτήσεις αυτές θα τις ομαδοποιήσουμε στην κατηγορία των συναρτήσεων ενημέρωσης, καθώς σκοπός

τους είναι η ενημέρωση του world state, ενώ η αρχική αναζήτηση λειτουργεί βοηθητικά. Μπορούμε να δούμε ένα τέτοιο παράδειγμα στο σχήμα 4.5.

```
//Update function updates the info of a tenant with new info in world state
func (s *SmartContract) Update(ctx contractapi.TransactionContextInterface, tenantNumber string, newName string, newEmail string, newPhone string) error {
    //getting the data from the world state
    tenant, err := s.IsTenant(ctx, tenantNumber)

    if err != nil {
        return err
    }
    //updating tenant info, all fields
    tenant.Name = newName
    tenant.Email = newEmail
    tenant.Phone = newPhone

    //storing back to the world state the updated info
    tenantAsBytes, _ := json.Marshal(tenant)

    return ctx.GetStub().PutState(tenantNumber, tenantAsBytes)
}
```

Σχήμα 4.5 - Απόσπασμα κώδικα που παρουσιάζει μία συνδυαστική συνάρτηση ενημέρωσης.

Από την πλευρά του χρήστη, προκειμένου να αξιοποιήσει τις συναρτήσεις αυτές, πρέπει να χρησιμοποιήσει το CLI ως peer, σε οποιοδήποτε από τα δύο organizations, για τους σκοπούς της εργασίας δε μας επηρεάζει αυτό. Προκειμένου να αλληλοεπιδράσει όμως με το δίκτυο πρέπει να ακολουθήσει πρώτα μερικά βήματα. Αρχικά θα πρέπει να δώσει τις εντολές που φαίνονται στο σχήμα 4.6 προκειμένου να συσχετιστεί με κάποια από τα αρχεία εγκατάστασης που αναφέρουμε ανωτέρω.

```
File  Edit  View  Search  Terminal  Help
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma# export PATH=${PWD}/../bin:${PWD}:$PATH
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma# export FABRIC_CFG_PATH=$PWD/../config/
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma#
```

Σχήμα 4.6 - Αρχικές εντολές

Στη συνέχεια μπορούμε να επιλέξουμε αν επιθυμούμε να χρησιμοποιούμε το CLI ως peer από τον πρώτο οργανισμό ή τον δεύτερο. Αυτό γίνεται μέσω της της ανάθεσης κάποιων μεταβλητών περιβάλλοντος στο CLI. Συγκεκριμένα, προκειμένου να λειτουργούμε ως peer από τον πρώτο οργανισμό πρέπει να θέσουμε τις ακόλουθες μεταβλητές που παρουσιάζονται στο σχήμα 4.7.

```
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma# export CORE_PEER_TLS_ENABLED=true
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma# export CORE_PEER_LOCALMSPID="Org1MSP"
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma# export CORE_PEER_TLS_ROOTCERT_FILE=${PWD}/organizations/peerOrganizations/
org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma# export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1
.example.com/users/Admin@org1.example.com/msp
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma# export CORE_PEER_ADDRESS=localhost:7051
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma#
```

Σχήμα 4.7 - Ορισμός μεταβλητών για χρήση ως peer από τον πρώτο οργανισμό.

Στη συνέχεια, ο peer μπορεί να πραγματοποιήσει δύο ειδών ενέργειες από το CLI, τις ακόλουθες:

- Invoke
- Query

Όταν επιθυμούμε να χρησιμοποιήσουμε συναρτήσεις οι οποίες θα κάνουν νέες εγγραφές στο ledger, όπως στην εγγραφή νέου tenant θα πρέπει να χρησιμοποιήσουμε το invoke. Παράδειγμα μίας τέτοιας ενέργειας είναι αυτή που φαίνεται στο σχήμα 4.8 και οδηγεί στην εγγραφή ενός ακόμη tenant στο ledger.

```
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma# peer chaincode invoke -o localhost:7050 --ordererTLSHostnameOverride order
er.example.com --tls true --cafile ${PWD}/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsc
a.example.com-cert.pem -C mychannel -n fabcar --peerAddresses localhost:7051 --tlsRootCertFiles ${PWD}/organizations/peerOrganizations/or
g1.example.com/peers/peer0.org1.example.com/tls/ca.crt --peerAddresses localhost:9051 --tlsRootCertFiles ${PWD}/organizations/peerOrganiz
ations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt -c '{"function":"Enroll","Args":["T10","Tenant Ten","10@mail.com","101010
1010"]}'
2020-07-18 19:40:18.345 EEST [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200
```

Σχήμα 4.8 - Chaincode Invoke

Αντίστοιχα, όταν θέλουμε να διαβάσουμε δεδομένα από το ledger, χωρίς να γράψουμε σε αυτό, χρησιμοποιούμε query όπως στο σχήμα 4.9, στο οποίο αναζητούμε στο ledger τον tenant T10 τον οποίο εγγράψαμε στο σχήμα 4.8. Η συνάρτηση IsTenant, για την οποία θα εξηγηθεί ο τρόπος λειτουργίας της στο κεφάλαιο 5, όπως και της Enroll του σχήματος 4.8, πηγαίνει στο ledger και αναζητά αν υπάρχει ο tenant αυτός, και επιστρέφει τα δεδομένα που βρήκε στο ledger. Αν δεν υπήρχε ο tenant θα επέστρεφε το κατάλληλο μήνυμα.

```
root@ThanosPC:/home/thanos/Samples/new-fabric-samples/diploma# peer chaincode query -C mychannel -n fabcar -c '{"Args":["IsTenant","T10"]
}'
{"pck":"T10","name":"Tenant Ten","email":"10@mail.com","phone":"1010101010","registered":true,"type":"T"}
```

Σχήμα 4.9 - Chaincode Query

4.3 Παρουσίαση των Assets της εφαρμογής

Αφού παρουσιάσαμε τη δομή του chaincode, στη συνέχεια θα ασχοληθούμε με τα assets και τη δομή τους. Πριν μπούμε όμως σε αυτό το κομμάτι, είναι σημαντικό να ασχοληθούμε με τη λογική πίσω από τα assets και το πώς συνδέονται με το cloud computing. Συγκεκριμένα, το πρόγραμμα μας, πραγματοποιεί τέσσερις λειτουργίες διαχείρισης των cloud υπηρεσιών (services) και των χρηστών (tenants). Αρχικά θα ασχοληθούμε με τη διαχείριση της ταυτότητας των services και των tenants, η οποία πραγματοποιείται μέσω της ανάθεσης ιδιωτικών κλειδιών. Κάθε υπηρεσία και κάθε χρήστης έχει ένα μοναδικό κλειδί, με το οποίο ξεχωρίζει από όλα τα υπόλοιπα στοιχεία της εφαρμογής. Αυτό μας οδηγεί στη δεύτερη λειτουργία, την πιστοποίηση των υπηρεσιών και των χρηστών, η οποία βασίζεται στην ίδια λογική με πριν, όπου το μοναδικό κλειδί χρησιμοποιείται προκειμένου να γίνει επαλήθευση και πιστοποίηση της ταυτότητας. Παράδειγμα χρήσης της πιστοποίησης είναι όταν επιθυμεί ένας χρήστης να πραγματοποιήσει μια ενέργεια η οποία όμως επιτρέπεται μόνο σε

συγκεκριμένους χρήστες, οπότε πρέπει να ελεγχθεί αν ανήκει στη λίστα με τους εγκεκριμένους χρήστες. Στη συνέχεια, έχουμε την εξουσιοδότηση. Η εξουσιοδότηση είναι στην ουσία ένα συμβόλαιο μεταξύ υπηρεσιών ή υπηρεσιών και χρηστών. Έχουν, αντίστοιχα με πριν, ένα μοναδικό κλειδί για να τα χαρακτηρίσει, αλλά περιέχουν και τα κλειδιά αυτών που μετέχουν, δηλαδή αυτού που εξουσιοδοτεί και αυτού που εξουσιοδοτείται. Τα συμβόλαια αυτά τα ονομάζουμε Delegations ενώ στη συνέχεια μπορούν να δημιουργηθούν και επιμέρους συμβόλαια, τα οποία ονομάζουμε Subdelegations και τα οποία είναι μεταξύ χρηστών και περιγράφουν τη χρονική διάρκεια για την οποία δεσμεύονται οι υπηρεσίες αλλά και πόσοι από τους διαθέσιμους πόρους, προκειμένου να δημιουργηθούν περαιτέρω συμβόλαια, κάθε υπηρεσίας δεσμεύονται. Με τη λήξη της χρονικής διάρκειας ή τον πρόωρο τερματισμό του συμβολαίου, οι πόροι αυτοί ελευθερώνονται και πάλι και είναι διαθέσιμοι να παραχωρηθούν εκ νέου, με τη δημιουργία νέου συμβολαίου. Τέλος, υπάρχει η λειτουργία της χρέωσης, όπου το κάθε συμβόλαιο κοστολογείται με βάση τη χρονική του διάρκεια και τη χρήση συγκεκριμένων πόρων. Η λειτουργία αυτή είναι ενδεικτική και εκτός από τη χρονική διάρκεια ασχολούμαστε με τον αριθμό των πυρήνων (cpu cores) που χρησιμοποιεί κάθε συμβόλαιο προκειμένου να πραγματοποιηθεί.

Στη συνέχεια, το κομμάτι που ακολουθεί θα χωριστεί σε τρία τμήματα, όπου το πρώτο θα παρουσιάσει τη δομή των tenants στον κώδικα μας, το δεύτερο τη δομή των υπηρεσιών (services) ενώ το τρίτο θα ασχοληθεί με την παρουσίαση των εξουσιοδοτήσεων (delegations) και των υπό-εξουσιοδοτήσεων (sub-delegations).

4.3.1 Tenants

Για τους tenants, κάθε εγγραφή στο σύστημα περνάει τα στοιχεία τους σε μία δομή (struct) που περιλαμβάνει κάποια βασικά στοιχεία, όπως το δημόσιο κλειδί (Pck) με το οποίο τον αναγνωρίζουμε καθώς και στοιχεία όπως το όνομα, το e-mail και το τηλέφωνο του tenant. Επιπρόσθετα, χρησιμοποιούνται κάποιες βοηθητικές μεταβλητές που εξυπηρετούν στη διαχείριση του tenant. Η μεταβλητή Registered είναι υπεύθυνη για να ενημερώνει το σύστημα αν ο tenant είναι ακόμη εγγεγραμμένος ή αν έχει διαγραφεί για οποιονδήποτε λόγο, μιας και δε μπορεί πρακτικά να διαγραφεί από το ledger, καθώς δεν επιτρέπεται από το blockchain αυτό. Τέλος, η μεταβλητή Type υπάρχει σε όλα τα assets και είναι βοηθητική ως προς τη διαχείριση τους από τις διάφορες συναρτήσεις. Στο σχήμα 4.10 παρατηρούμε

τη δομή από την οποία αποτελείται ο κάθε tenant. Τα στοιχεία αυτά αποθηκεύονται σε JSON μορφή.

```
// Tenant describes basic details of what makes up a tenant
type Tenant struct {
    Pck         string   `json:"pck"`
    Name        string   `json:"name"`
    Email       string   `json:"email"`
    Phone       string   `json:"phone"`
    Registered  bool     `json:"registered"`    //false if not, true if Registered
    Type        string   `json:"type"`          //T for tenants
}
```

Σχήμα 4.10 – Δομή ενός Tenant

4.3.2 Services

Για τις υπηρεσίες, κάθε εγγραφή στο σύστημα, αντίστοιχα με τους tenants, περνάει τα στοιχεία τους σε μία δομή (struct) που περιλαμβάνει το κλειδί (Pck) με το οποίο αναγνωρίζουμε την υπηρεσία και το όνομα της υπηρεσίας. Επιπρόσθετα, χρησιμοποιείται η βοηθητική μεταβλητή Registered η οποία έχει αντίστοιχη χρήση με αυτή στους tenants, ενώ παρατηρούμε πως και εδώ υπάρχει η μεταβλητή Type. Τα στοιχεία, αντίστοιχα με πριν, αποθηκεύονται σε JSON μορφή. Στο σχήμα 4.11 μπορούμε να δούμε τη δομή μιας υπηρεσίας.

```
//Service describes basic details of what makes up a service
type Service struct {
    Pck                 string  `json:"pck"`
    Name                string  `json:"name"`
    Registered          bool    `json:"registered"` //true if registered false if not
    Type                string  `json:"Type"`       //S for Services
}
```

Σχήμα 4.11 – Δομή ενός Service

Στο σημείο αυτό, είναι σημαντικό να τονίσουμε πως τα πεδία των δομών των υπηρεσιών και των ενοίκων είναι ενδεικτικά, καθώς μπορούν να τροποποιηθούν προκειμένου να περιλαμβάνουν τα ανά περίπτωση απαραίτητα στοιχεία της εκάστοτε εφαρμογής. Πρακτικά αυτό σημαίνει πως εάν επιθυμούσαμε θα μπορούσαμε να προσθέσουμε για παράδειγμα ένα πεδίο διεύθυνσης για τους ενοίκους όπου θα καταχωρούσαμε τη διεύθυνση κατοικίας τους ή ένα πεδίο e-mail στα services, όπου θα καταχωρούσαμε το e-mail της κάθε υπηρεσίας, δίχως να αλλάξει η λογική με την οποία λειτουργεί το πρόγραμμα μας. Για τους σκοπούς όμως της εν λόγω εφαρμογής δε κρίθηκε απαραίτητο να προστεθούν επιπλέον πεδία από αυτά που αναφέρθηκαν.

4.3.3 Delegations και Subdelegations

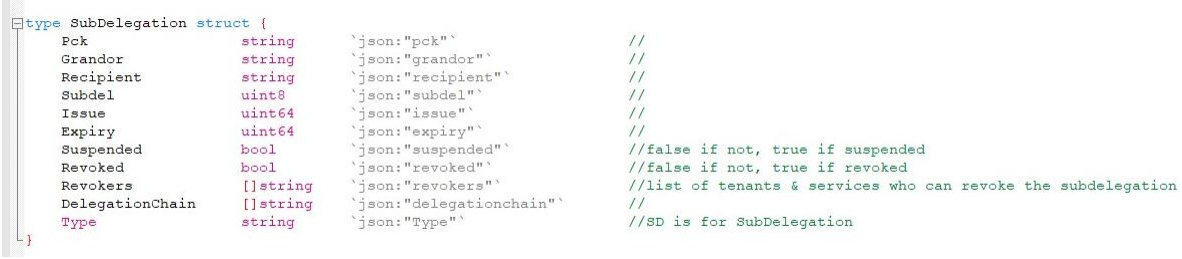
Στο κεφάλαιο αυτό θα παρουσιάσουμε τις εξουσιοδοτήσεις (delegations) αλλά και τις υπο-εξουσιοδοτήσεις (sub-delegations). Οι εξουσιοδοτήσεις μπορούν να γίνουν είτε από service σε service είτε από service σε tenant. Αντίστοιχα, οι υπό-εξουσιοδοτήσεις μπορούν να γίνουν μόνο από tenant σε tenant. Οι εξουσιοδοτήσεις στην ουσία είναι τα δικαιώματα χρήσης, για παράδειγμα ενός tenant σε μια υπηρεσία.

```
//Delegation describes basic details of what makes up a Delegation
type Delegation struct {

    Pck              string   `json:"pck"`              //
    Grandor          string   `json:"grandor"`          //
    Recipient        string   `json:"recipient"`        //
    Subdel           uint8    `json:"subdel"`           //
    Issue            uint64   `json:"issue"`            //
    Expiry           uint64   `json:"expiry"`           //
    Suspended        bool     `json:"suspended"`        //false if not, true if suspended
    Revoked          bool     `json:"revoked"`          //false if not, true if revoked
    Revokers         []string `json:"revokers"`         //list of tenants & services who can revoke the delegation
    DelegationChain  []string `json:"delegationchain"`  //
    Type             string   `json:"Type"`             //D is for Delegation
}
```

Σχήμα 4.12 – Δομή ενός Delegation

Στο σχήμα 4.12 παρατηρούμε τη δομή του delegation. Αυτό αποτελείται, όπως όλα, από το μοναδικό δημόσιο κλειδί του (Pck) το οποίο λαμβάνει κατά την εγγραφή του στο ledger από τον χρήστη, το πεδίο Grantor και Recipient, τα οποία αναφέρονται στον εντολέα και τον αποδέκτη της εξουσιοδότησης. Ο όρος Subdel ορίζει πόσες υπό-εξουσιοδοτήσεις μπορεί να δημιουργήσει ο αποδέκτης της εξουσιοδότησης. Στη συνέχεια έχουμε τους όρους Issue και Expiry τα οποία ορίζουν τη χρονική έναρξη και τη λήξη του εκάστοτε delegation. Ακολούθως, έχουμε τα Revoked και Suspended τα οποία λειτουργούν ως βοηθητικές μεταβλητές, προκειμένου να μπορούμε να σημειώνουμε εύκολα και γρήγορα αν κάποιο delegation έχει ανακληθεί ή ανασταλεί. Μετά, έχουμε μία λίστα η οποία ενημερώνεται με τα στοιχεία όσων μπορούν να κάνουν ανάκληση σε ένα delegation. Τέλος, έχουμε μία βοηθητική λίστα με τα delegation και subdelegation που συνδέονται μεταξύ τους (μέσω του subdel που αναφέραμε προηγουμένως) και τη βοηθητική μεταβλητή Type που υπάρχει σε όλες τις δομές και χρησιμοποιείται από τις συναρτήσεις για να καθορίσει το είδος της δομής που επεξεργάζεται η εκάστοτε συνάρτηση του προγράμματος.



```
type SubDelegation struct {
    Pck                 string      `json:"pck"`                  //
    Grandor             string      `json:"grandor"`              //
    Recipient           string      `json:"recipient"`            //
    Subdel              uint8       `json:"subdel"`               //
    Issue               uint64      `json:"issue"`                //
    Expiry              uint64      `json:"expiry"`               //
    Suspended           bool        `json:"suspended"`            //false if not, true if suspended
    Revoked             bool        `json:"revoked"`              //false if not, true if revoked
    Revokers            []string    `json:"revokers"`             //list of tenants & services who can revoke the subdelegation
    DelegationChain     []string    `json:"delegationchain"`      //
    Type                string      `json:"Type"`                 //SD is for SubDelegation
}
```

Σχήμα 4.13 – Δομή ενός Subdelegation

Στο σχήμα 4.13 βλέπουμε και τη δομή των subdelegations. Όπως παρατηρούμε, η δομή είναι αντίστοιχη με αυτή των delegations, με μόνη διαφοροποίηση αυτή στο πεδίο Type. Τα στοιχεία, όπως και πριν, αποθηκεύονται στο ledger σε JSON μορφή.

5. Λειτουργίες εφαρμογής

Το κεφάλαιο αυτό θα χωριστεί σε τρία τμήματα, όπου στο πρώτο θα γίνει παρουσίαση του τρόπου διαχείρισης των χρηστών-ενοίκων, στο δεύτερο των υπηρεσιών και στο τρίτο των εξουσιοδοτήσεων. Σε κάθε λειτουργία και κλήση που υποστηρίζεται θα παρουσιαστούν και αντίστοιχα διαγράμματα ακολουθίας (message sequence diagrams) με όλα τα πιθανά σενάρια εξέλιξής του. Επίσης, θα θεωρούμε, πως οι κλήσεις γίνονται από τον peer του πρώτου organization, χωρίς όμως αυτό να επηρεάζει τη λειτουργικότητα του προγράμματος με κάποιο τρόπο. Ο κώδικας της εφαρμογής παρατίθεται εδώ:

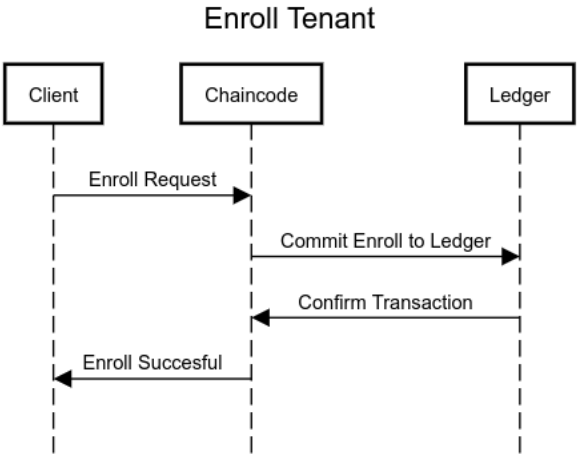
https://github.com/Kthanasis/Tenant-Service-Cloud-Management-using-Blockchain

5.1 Tenant Management

Εγγραφή

Η πρώτη λειτουργία διαχείρισης των tenants που θα παρουσιάσουμε είναι η εγγραφή (enroll) τους στο ledger. Για την εγγραφή ο χρήστης δίνει στο σύστημα το μοναδικό κλειδί που θα χαρακτηρίζει τον ένοικο, το όνομα του, το e-mail καθώς και το τηλέφωνό του. Τα υπόλοιπα στοιχεία που υπάρχουν στο struct συμπληρώνονται αυτόματα και στη συνέχεια αποθηκεύονται στο ledger σε JSON μορφή. Από τη στιγμή που θα δημιουργηθεί μία εγγραφή στο ledger, εξαιτίας της λογικής του blockchain δεν είναι δυνατό να αφαιρεθεί ή να τροποποιηθεί, συνεπώς οποιαδήποτε αλλαγή γίνει από τη στιγμή της εγγραφής και μετά θα αποθηκευτεί εκ νέου στο ledger, δημιουργώντας με αυτόν τον τρόπο ένα πλήρες ιστορικό. Μετά την εγγραφή του tenant, είναι άμεσα διαθέσιμος για να πραγματοποιήσει συναλλαγές με το σύστημα.

Αναλυτικά, όπως βλέπουμε στο σχήμα 5.1, ο peer κάνει invoke το chaincode προκειμένου να χρησιμοποιήσει την συνάρτηση Enroll ενώ παράλληλα αποστέλλει τα στοιχεία του νέου tenant. Η συνάρτηση Enroll του chaincode στη συνέχεια, καταχωρεί την εγγραφή στο ledger. Στη συνέχεια το ledger επιβεβαιώνει τη συναλλαγή στο chaincode το οποίο με τη σειρά του επιστρέφει στον peer μήνυμα για την επιτυχή εγγραφή του νέου tenant.

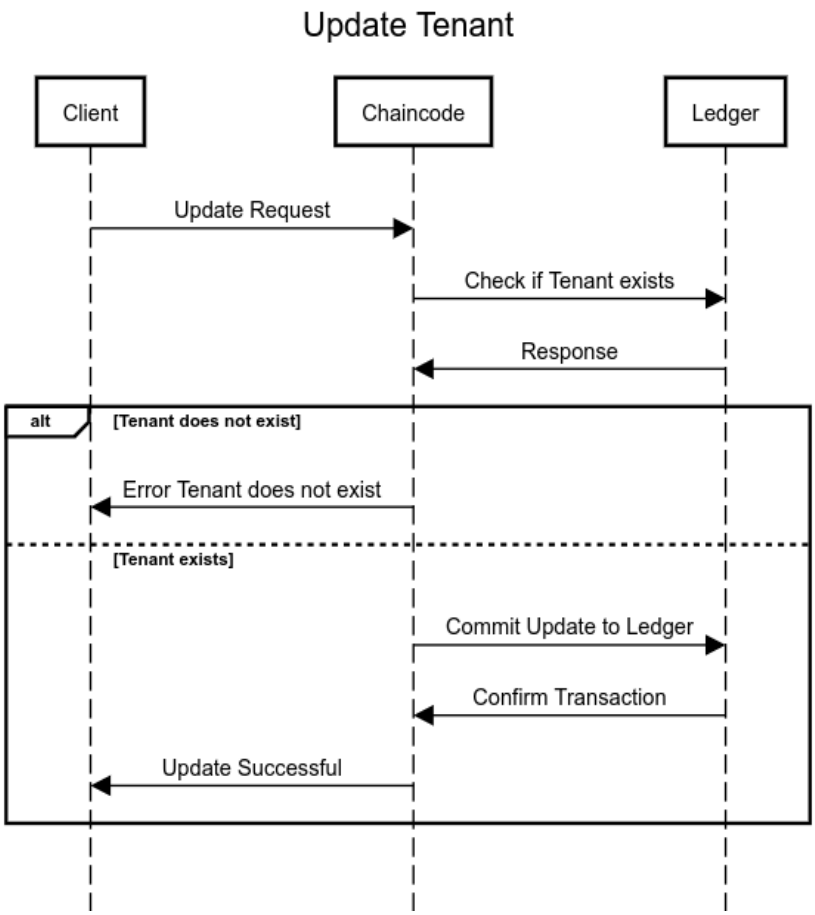


Σχήμα 5.1 - Enroll tenant

Ενημέρωση

Η δεύτερη λειτουργία διαχείρισης των tenants είναι όταν ο peer επιθυμεί να ενημερώσει τα στοιχεία ενός tenant. Όπως αναφέρθηκε και ανωτέρω, σε αυτή τη περίπτωση δεν ενημερώνεται η δομή που δημιουργήθηκε εξ αρχής, καθώς αυτό δεν είναι δυνατό εξαιτίας των βασικών αρχών του blockchain. Αντίθετα, δημιουργείται μία νέα δομή, η οποία βασίζεται στο ίδιο κλειδί που είχε και πριν ο tenant και χρησιμοποιείται ως αναγνωριστικό του, ενώ τα υπόλοιπα στοιχεία (όνομα, e-mail και τηλέφωνο) μπορούν να μεταβληθούν

Όπως παρατηρούμε και στο σχήμα 5.2, αρχικά ο peer κάνει invoke τη συνάρτηση update αποστέλλοντας και τα αντίστοιχα στοιχεία. Η συνάρτηση αρχικά περιλαμβάνει μία ερώτηση προς το ledger όπου ελέγχει την ύπαρξη ή όχι του tenant. Σε περίπτωση που δεν υπάρχει ήδη καταχωρημένος ο tenant επιστρέφεται ένα μήνυμα σφάλματος στον peer που τον ενημερώνει σχετικά. Σε αντίθετη περίπτωση, αποστέλλονται τα νέα δεδομένα στο ledger όπου δημιουργείται μία νέα εγγραφή με τα ενημερωμένα στοιχεία. Στη συνέχεια η εγγραφή επιβεβαιώνεται και εμφανίζεται το μήνυμα επιτυχούς πραγματοποίησης της συναλλαγής στον peer.



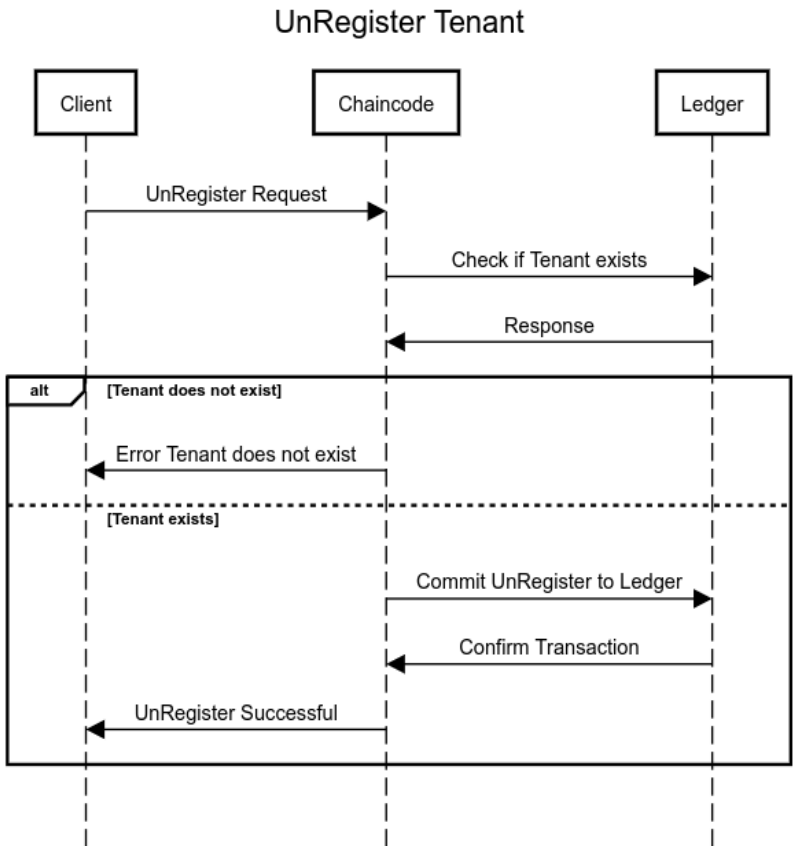
Σχήμα 5.2 - Update tenant

Διαγραφή

Στη λειτουργία αυτή, ο χρήστης αναζητά στο ledger με τη χρήση του κλειδιού τον ένοικο που θέλει να διαγράψει. Φυσικά, δε πρόκειται να αφαιρεθούν τα στοιχεία από το ledger καθώς αυτό δεν είναι δυνατό, αλλά αυτό που συμβαίνει στην προκειμένη περίπτωση είναι μία νέα εγγραφή με το ίδιο κλειδί, όπου τα στοιχεία του ενοίκου παραμένουν αμετάβλητα, αλλά τροποποιείται η βοηθητική μεταβλητή Registered σε false. Στη συνέχεια αποθηκεύουμε εκ νέου στο ledger. Με αυτόν τον τρόπο, από τη στιγμή της διαγραφής και μετά, ο ένοικος αυτός δεν μπορεί να πραγματοποιήσει άλλες συναλλαγές στο σύστημα καθώς η μεταβλητή Registered θα ελέγχεται συνεχώς, και έχοντας την τιμή false θα απορρίπτεται οποιοδήποτε αίτημα.

Η διαδικασία που ακολουθείται εδώ είναι όμοια με πριν. Αρχικά, σύμφωνα και με το σχήμα 5.3, ο peer κάνει invoke το chaincode χρησιμοποιώντας τη συνάρτηση διαγραφής των tenant και δίνει το Pck του. Στη συνέχεια η συνάρτηση ψάχνει στο ledger να δει αν είναι εγγεγραμμένος ο tenant. Σε περίπτωση που δεν είναι, ο peer παίρνει ως απάντηση ένα

μήνυμα που αναφέρει πως δεν υπάρχει tenant με το Pck αυτό αποθηκευμένος στο ledger. Αν όμως ο tenant βρεθεί, τότε στη συνέχεια η συνάρτηση θα ξαναστείλει στο ledger αίτημα, αυτή τη φορά για καταχώρηση όμως των νέων δεδομένων. Θα δημιουργηθεί νέα εγγραφή στο ledger όπου η μεταβλητή Registered θα είναι πλέον false. Στη συνέχεια το ledger θα επιβεβαιώσει τη συναλλαγή και ο peer θα λάβει το αντίστοιχο μήνυμα επιτυχούς ενημέρωσης του ledger.



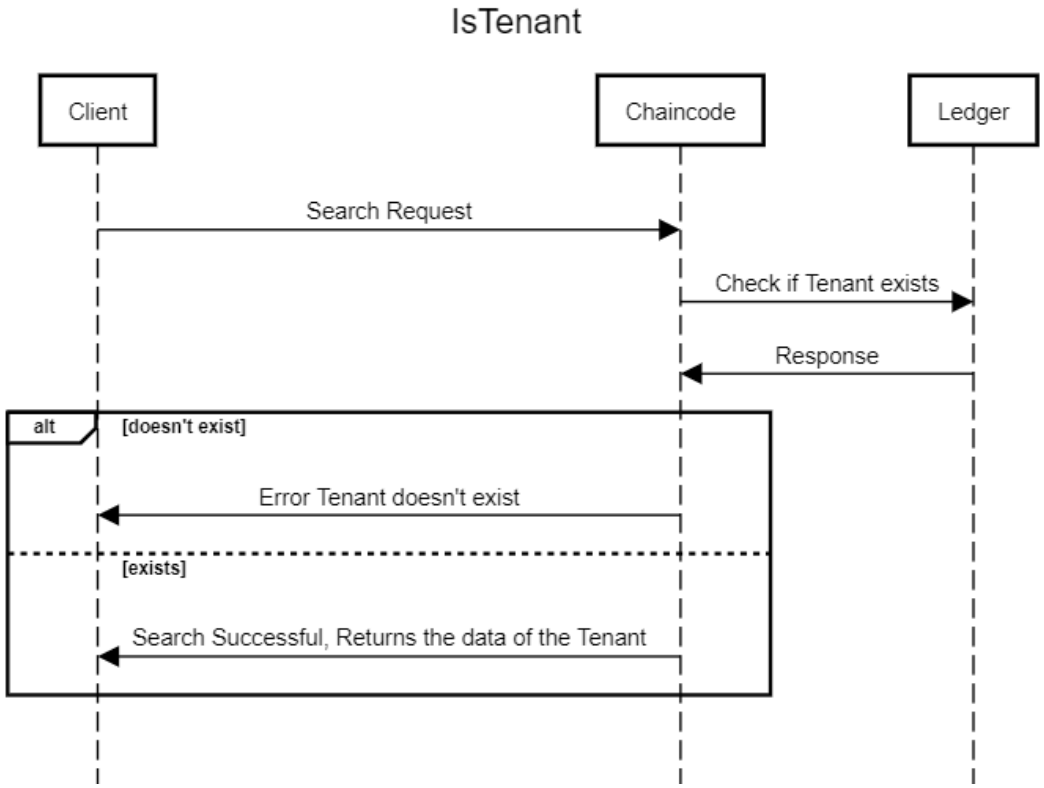
Σχήμα 5.3 - Διαγραφή tenant

Αναζήτηση

Η λειτουργία της αναζήτησης των ενοίκων είναι από τις βασικότερες, καθώς εκτός από την αυτόνομη λειτουργία της, την αναζήτηση δηλαδή των ενοίκων στο ledger με βάση το κλειδί με το οποίο έχουν αποθηκευτεί, χρησιμοποιείται και ως βοηθητική λειτουργία για τις περισσότερες λειτουργίες διαχείρισης των ενοίκων, και όχι μόνο, καθώς μέσω αυτής ανακτούμε τα δεδομένα από το ledger.

Αναλυτικά, όπως παρουσιάζεται και στο σχήμα 5.4, έχουμε ένα query από τον peer, καθώς θέλει απλά να ανακτήσει δεδομένα από το ledger. Στη συνέχεια, το chaincode ζητά

πρόσβαση στο ledger προκειμένου να δει αν ο tenant υπάρχει. Το ledger απαντά, ενώ στη συνέχεια, εάν ο tenant δεν υπάρχει στο ledger (αν είχε εγγραφεί και στη συνέχεια διαγράφηκε, θα παραμένει στο ledger, καθώς υπενθυμίζουμε, το ledger δεν επιτρέπει την διαγραφή ή επεξεργασία δεδομένων του, παρά μόνο την αναζήτηση ή την προσθήκη, θα επιστραφεί στον peer ένα μήνυμα που θα τον ενημερώνει σχετικά. Σε αντίθετη περίπτωση, θα επιστραφούν στο CLI του peer τα δεδομένα του tenant, όπως παρατηρούμε και στο σχήμα 4.9.



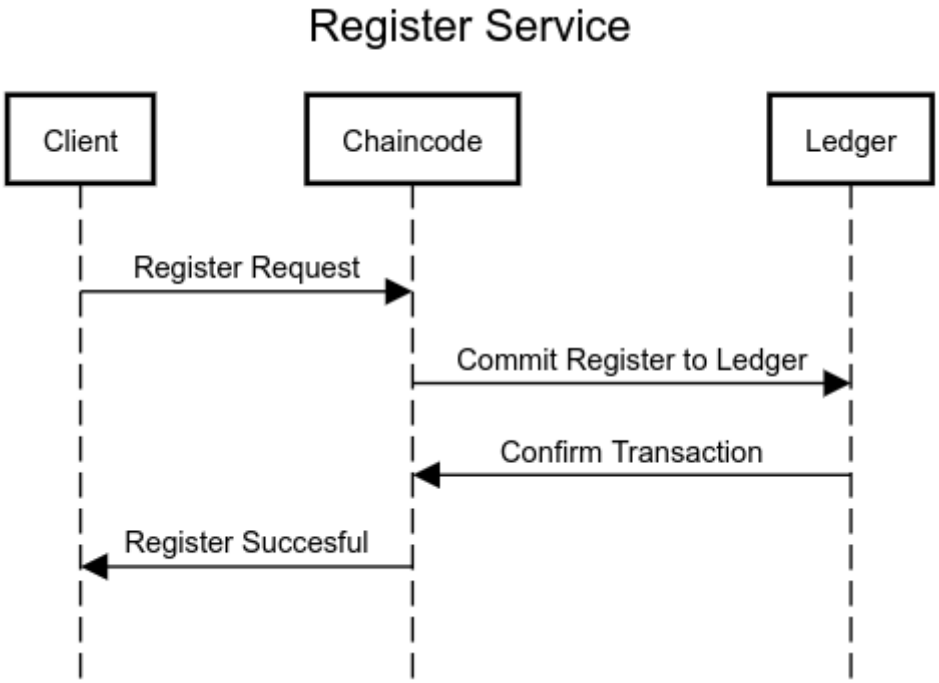
Σχήμα 5.4 - Αναζήτηση tenant

5.2 Service Management

Εγγραφή

Ο χρήστης εδώ θέλει να εγγράψει μία νέα υπηρεσία στο ledger. Για την υλοποίηση χρειάζεται να δώσει ένα μοναδικό δημόσιο κλειδί (Pck) και το όνομα της υπηρεσίας, ενώ οι τιμές των βοηθητικών μεταβλητών δημιουργούνται αυτόματα. Στη συνέχεια, η δομή

αποθηκεύεται στο ledger με βάση το κλειδί της υπηρεσίας σε JSON μορφή. Η υπηρεσία, μετά την εγγραφή της στο ledger είναι άμεσα διαθέσιμη.



Σχήμα 5.5 - Εγγραφή Υπηρεσίας

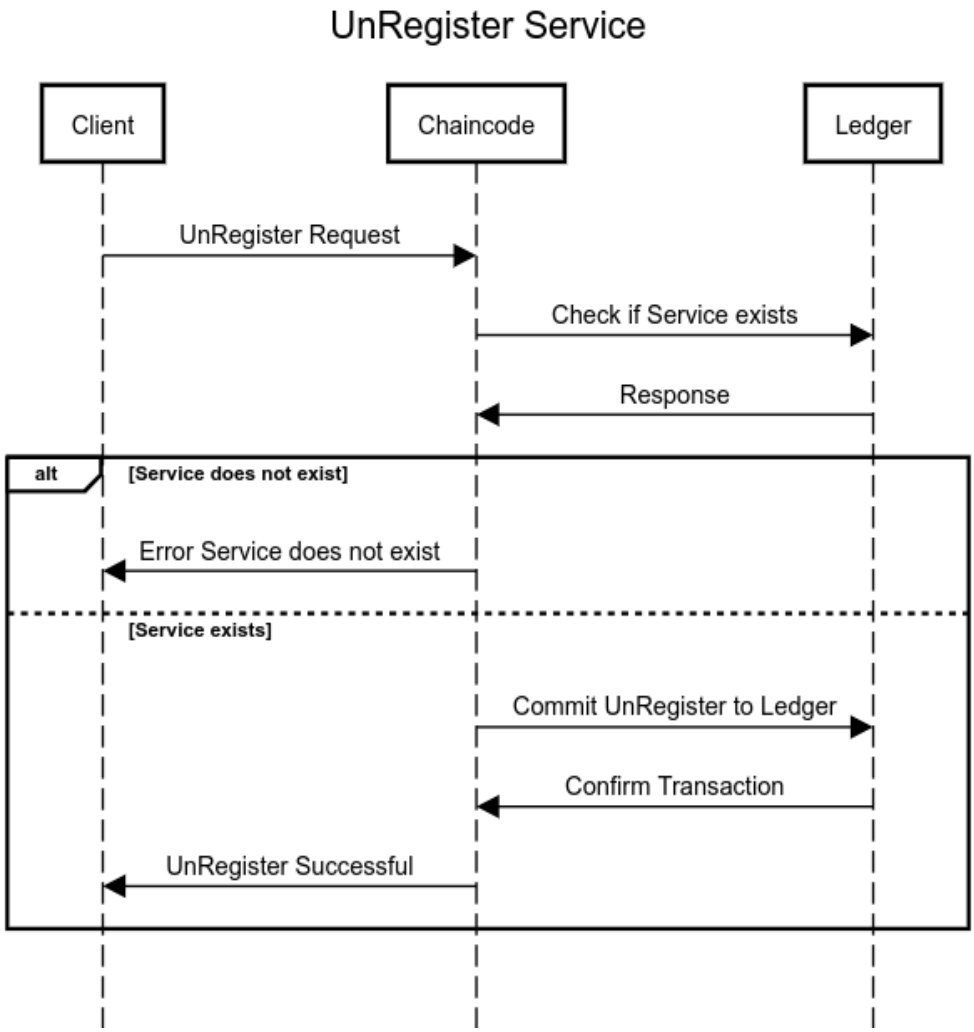
Εδώ η αναλυτική διαδικασία που ακολουθείται και παρουσιάζεται στο σχήμα 5.5, είναι όμοια με τη διαδικασία εγγραφής ενός tenant. Ο peer κάνει invoke το chaincode, δίνοντας τα κατάλληλα στοιχεία ως παραμέτρους της συνάρτησης εγγραφής. Στη συνέχεια το chaincode υποβάλει τα δεδομένα στο ledger, όπου δημιουργείται η εγγραφή και λαμβάνει την επιβεβαίωση της συναλλαγής. Στη συνέχεια, ο peer ενημερώνεται πως η διαδικασία εγγραφής ήταν επιτυχής.

Διαγραφή

Στη λειτουργία αυτή, ο χρήστης επιθυμεί να διαγράψει μία υπηρεσία από τις διαθέσιμες πως αξιοποίηση υπηρεσίες. Όπως και πριν, η ουσιαστική διαγραφή από το ledger δεν είναι δυνατή, καθώς αυτό θα ήταν αντίθετο στις αρχές του blockchain. Η υπηρεσία μόλις διαγραφεί παύει άμεσα να είναι διαθέσιμη.

Αντίστοιχα και εδώ, ο peer κάνει invoke το chaincode, δίνοντας ως όρισμα στη συνάρτηση διαγραφής μόνο το Pck της υπηρεσίας. Στη συνέχεια, το chaincode ζητάει από το ledger τα στοιχεία της υπηρεσίας. Εάν τα στοιχεία αυτά δε βρεθούν αποθηκευμένα στο ledger, τότε

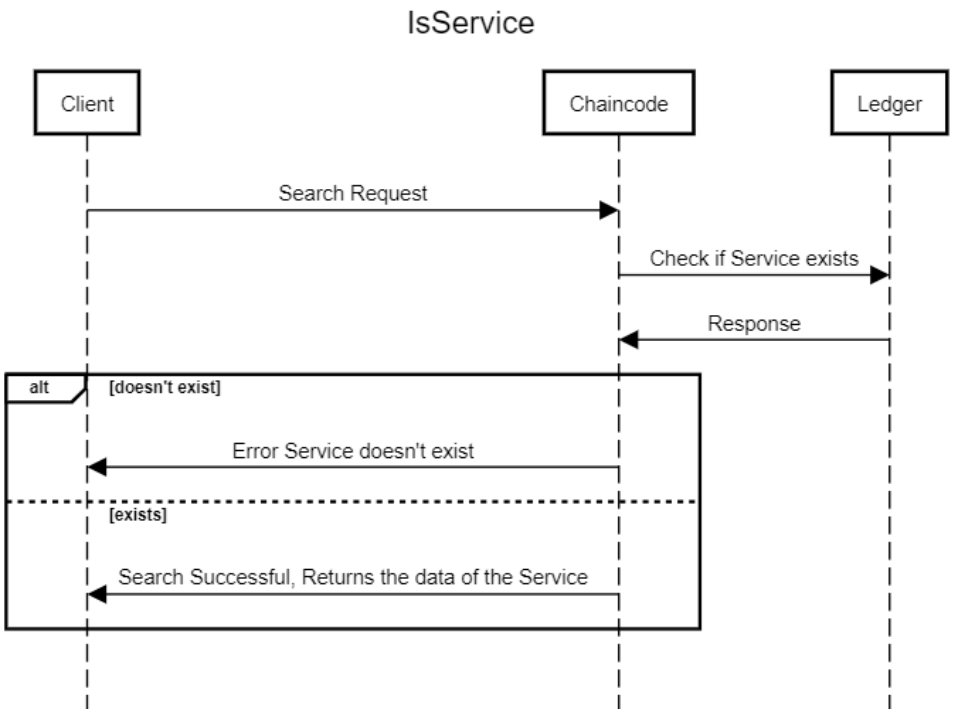
στον peer επιστρέφεται ένα μήνυμα σφάλματος, το οποίο αναφέρει πως δεν υπάρχει η συνάρτηση αυτή. Αντίθετα, σε περίπτωση που τα στοιχεία βρεθούν στο ledger, το chaincode θα αλλάξει τη μεταβλητή Registered σε false και θα υποβάλλει εκ νέου τα στοιχεία του service στο ledger. Αυτό με τη σειρά του θα επιβεβαιώσει τη πραγματοποίηση της συναλλαγής και τέλος στον peer θα εμφανιστεί μήνυμα πως πραγματοποιήθηκε η συναλλαγή του. Τη διαδικασία μπορούμε να τη δούμε και στο σχήμα 5.6.



Σχήμα 5.6 - Διαγραφή Υπηρεσίας

Αναζήτηση

Η λειτουργία αυτή βοηθάει τον χρήστη να αναζητήσει τις υπηρεσίες που έχουν εγγραφεί στο σύστημα. Εάν η υπηρεσία εντοπιστεί στο ledger θα επιστρέψει τα δεδομένα της σε JSON μορφή. Μία από τις βασικότερες λειτουργίες διαχείρισης των υπηρεσιών, είτε αυτόνομα είτε συμπληρωματικά με άλλες λειτουργίες και μπορούμε να τη δούμε στο σχήμα 5.7



Σχήμα 5.7 - Αναζήτηση Υπηρεσίας

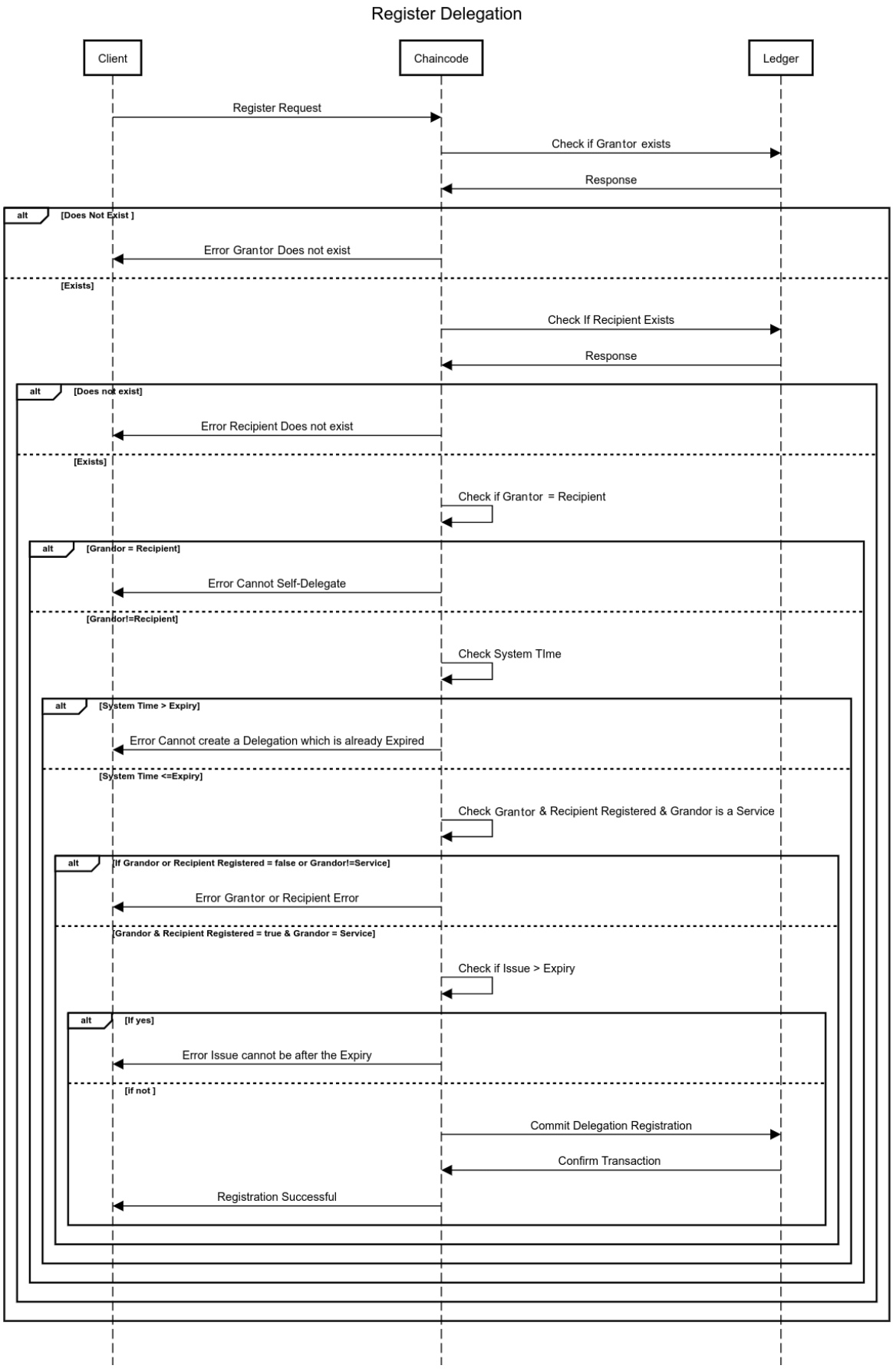
Ο peer εδώ πραγματοποιεί ένα query προς το chaincode, όπου δίνει ως όρισμα στη συνάρτηση αναζήτησης το Pck της υπηρεσίας που αναζητά. Στη συνέχεια το chaincode θα ζητήσει τα δεδομένα από το ledger, το οποίο με τη σειρά του θα απαντήσει στο chaincode. Σε περίπτωση που δεν βρεθούν τα δεδομένα στο ledger τότε η απάντηση που παίρνει ο peer από το chaincode είναι ένα μήνυμα σφάλματος το οποίο αναφέρει πως δεν υπάρχει η υπηρεσία που αναζητεί. Αντίθετα, αν βρεθεί, τότε επιστρέφονται τα στοιχεία της υπηρεσίας.

5.3 Delegation και Subdelegation Management

Εγγραφή delegation

Ο χρήστης εδώ επιθυμεί να δημιουργήσει ένα delegation μεταξύ ενός Service και ενός ακόμη Service, ή μεταξύ ενός Service και ενός Tenant. Η συνάρτηση αυτή παίρνει ως ορίσματα το δημόσιο κλειδί Pck του delegation, με το οποίο θα αναφερόμαστε σε αυτό, τα Pcks των Grantor και Recipient, δηλαδή από ποιον θα είναι το delegation και σε ποιον, τον αριθμό των επιτρεπόμενων subdelegations και τη χρονική έναρξη και λήξη του. Τα υπόλοιπα πεδία συμπληρώνονται αυτόματα μετά από τους κατάλληλους ελέγχους. Στη συνέχεια αποθηκεύει στο ledger σε μορφή JSON το εκάστοτε delegation. Απαγορεύεται να δημιουργηθεί delegation από από Tenant σε Tenant και από Tenant σε Service.

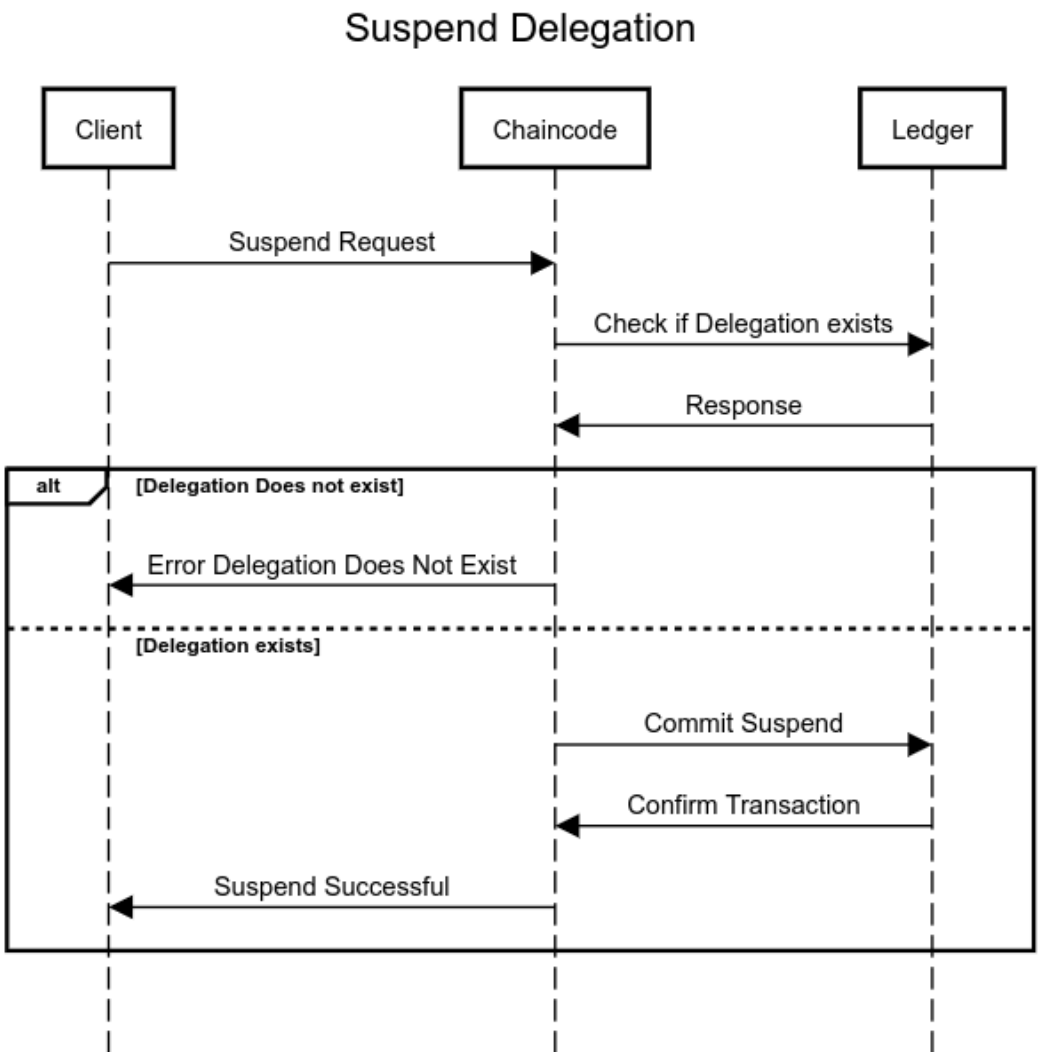
Αναλυτικά, σύμφωνα και με το σχήμα 5.8, ο peer κάνει invoke το chaincode και δίνει τα αντίστοιχα ορίσματα της συνάρτησης εγγραφής. Αρχικά το chaincode πηγαίνει στο ledger και κοιτάει αν ο grantor υπάρχει στο ledger. Εάν δεν υπάρχει εγγεγραμμένο το service τότε το chaincode επιστρέφει στον peer μήνυμα σφάλματος, όπου τον ενημερώνει σχετικά. Εάν υπάρχει τότε το chaincode ξανακάνει αίτημα προς το ledger προκειμένου να δει εάν υπάρχει ο recipient αυτή τη φορά. Η διαδικασία που ακολουθεί είναι όμοια. Το ledger επιστρέφει τα δεδομένα στο chaincode, εάν δεν υπάρχει επιστρέφεται μήνυμα σφάλματος στον peer. Σε αντίθετη περίπτωση το chaincode συνεχίζει και κάνει ελέγχους προκείμενου να δει εάν υπάρχει κάποιο θέμα με τις υπόλοιπες παραμέτρους. Αναλυτικά ελέγχει εάν ο grantor είναι ο ίδιος με τον recipient, κάτι που δεν επιτρέπεται. Αν ναι, η διαδικασία διακόπτεται και επιστρέφεται στον peer το κατάλληλο μήνυμα. Εάν όχι, ελέγχεται εάν το delegation έχει ήδη λήξει κατά τη διάρκεια της εγγραφής, κάτι που εάν συμβαίνει θα διακοπεί η διαδικασία και θα επιστραφεί το κατάλληλο μήνυμα στον peer. Εάν ως εδώ δεν έχει προκύψει κάτι από τα παραπάνω, ελέγχεται εάν ένας από τους δύο, grantor και recipient, έχει διαγραφεί. Σε περίπτωση που κανείς δεν έχει διαγραφεί η διαδικασία συνεχίζει και εξετάζει εάν ο grantor είναι service ενώ στη συνέχεια εάν η ώρα λήξης του delegation είναι μικρότερη από την ώρα έκδοσης. Εάν κάποιος από τους τελευταίους ελέγχους εντοπίσει κάποιο πρόβλημα, θα σταματήσει τη διαδικασία και θα επιστρέψει το αντίστοιχο μήνυμα. Σε αντίθετη περίπτωση, το chaincode υποβάλει τα δεδομένα στο ledger, το οποίο επιβεβαιώνει τη συναλλαγή και το chaincode επιστρέφει μήνυμα επιτυχούς εγγραφής του delegation στον peer.



Σχήμα 5.8 - Εγγραφή Delegation

Αναστολή Delegation

Στη περίπτωση αυτή επιθυμούμε να αναστείλουμε κάποιο delegation. Η αναζήτηση γίνεται με βάση το Pck και η εφαρμογή της αναστολής είναι άμεση.



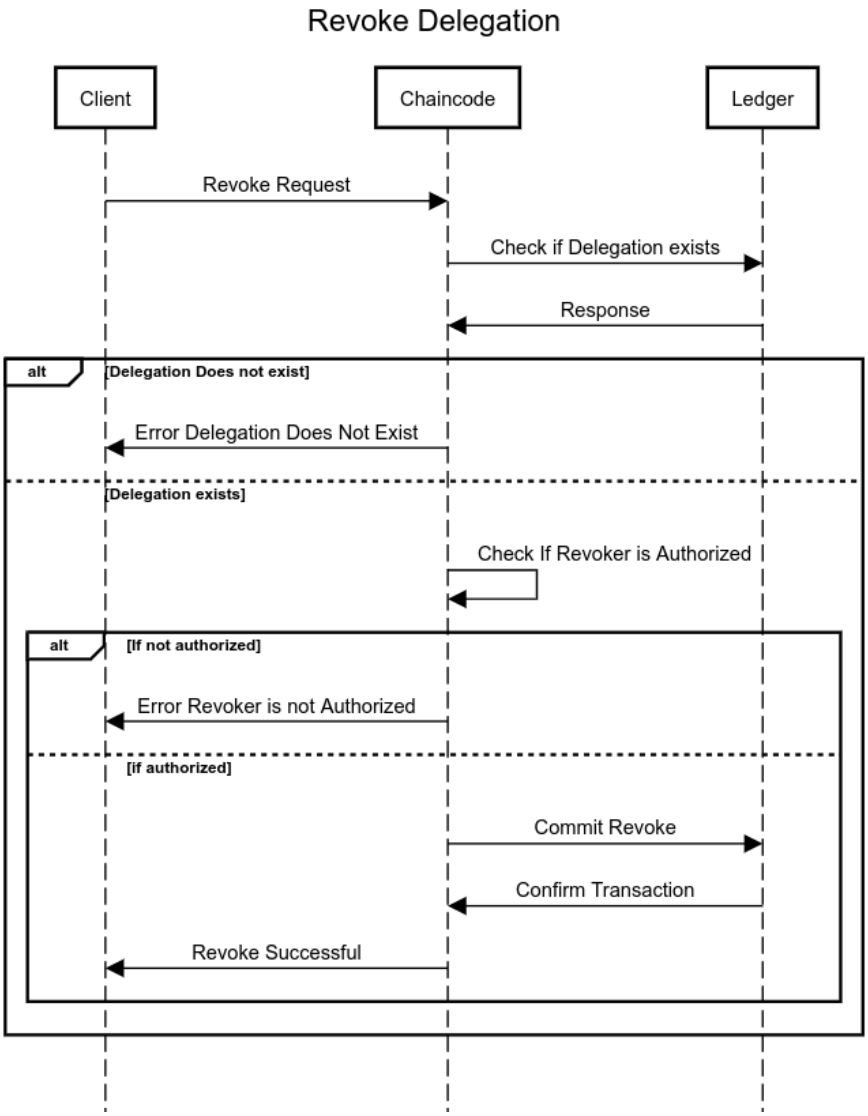
Σχήμα 5.9 - Αναστολή Delegation

Ακόμη μία περίπτωση invoke του chaincode από τον peer. Ως μοναδικό όρισμα της συνάρτησης αναστολής δίνεται το Pck του, με το οποίο στη συνέχεια το chaincode πάει στο ledger και ελέγχει την ύπαρξη του delegation. Σε περίπτωση που το delegation δεν υπάρχει, το ledger επιστρέφει την ανάλογη απάντηση στο chaincode, το οποίο με τη σειρά του ενημερώνει τον peer πως δεν υπάρχει το delegation το οποίο επιθυμεί να αναστείλει. Εάν υπάρχει όμως, τότε το chaincode στέλνει στο ledger το ενημερωμένο delegation, όπου η μεταβλητή Suspended πλέον θα είναι true. Το ledger επιβεβαιώνει τη συναλλαγή στο

chaincode και αυτό με τη σειρά του ενημερώνει τον peer για την επιτυχή αναστολή του delegation. Η διαδικασία περιγράφεται στο σχήμα 5.9.

Ανάκληση Delegation

Η λογική της συνάρτησης αυτής είναι όμοια με την προηγούμενη, εδώ όμως εκτός από το Pck, ως όρισμα δίνεται και το Pck αυτού που θέλει να κάνει την ανάκληση. Αυτοί που επιτρέπεται να κάνουν ανάκληση ενός delegation είναι εγγεγραμμένοι στη λίστα Revokers, στη δομή του delegation.



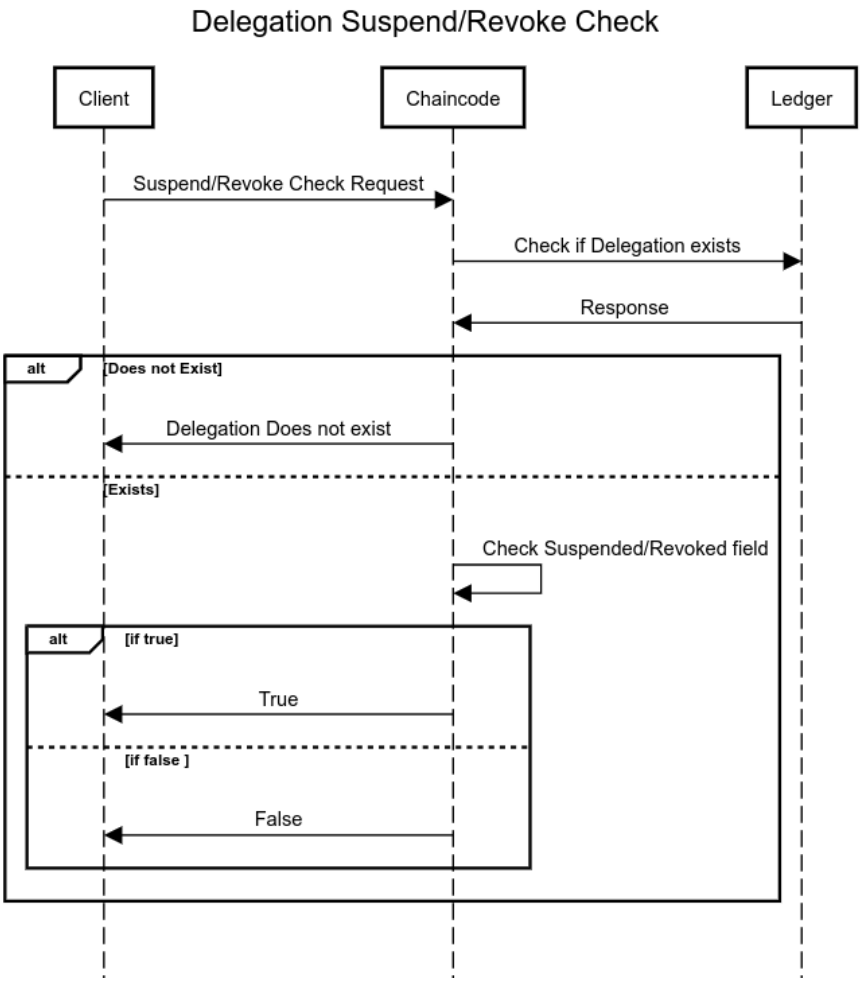
Σχήμα 5.10 - Ανάκληση Delegation

Αναλυτικά, ο peer κάνει invoke το chaincode και δίνει τα κατάλληλα ορίσματα της συνάρτησης ανάκλησης. Όπως βλέπουμε και στο σχήμα 5.10, στη συνέχεια το chaincode ζητάει από το ledger τα στοιχεία του delegation με το αντίστοιχο Pck. Το ledger επιστρέφει στο chaincode με την απάντηση στο αίτημα. Εάν το delegation δεν υπάρχει, τότε το chaincode επιστρέφει σχετικό μήνυμα στον peer και η διαδικασία σταματά εκεί. Αντίθετα, αν υπάρχει, τότε το chaincode ελέγχει τα στοιχεία του delegation, και βρίσκει αν αυτός που επιθυμεί να κάνει ανάκληση στο συγκεκριμένο delegation είναι εγκεκριμένος. Εάν όχι, τότε επιστρέφει στον peer σχετικό μήνυμα που τον ενημερώνει πως δεν ολοκληρώθηκε η ανάκληση καθώς δεν ήταν εγκεκριμένος ο revoker. Αν όμως, είναι εγκεκριμένος, τότε το chaincode ενημερώνει το πεδίο Revoked σε true και καταχωρεί την ανάκληση στο ledger. Αυτό με τη σειρά του επιβεβαιώνει την ανάκληση πίσω στο chaincode, το οποίο ενημερώνει τον peer πως η ανάκληση ολοκληρώθηκε.

Έλεγχος Αναστολής και Ανάκλησης Delegation

Αν και πρόκειται για δύο διαφορετικές συναρτήσεις, η λογική και στις δύο είναι ίδια, συνεπώς αποφασίσαμε να τις ομαδοποιήσουμε στην παρουσίαση τους. Ο χρήστης δίνει ως όρισμα το Pck που επιθυμεί να ελέγξει και παίρνει ως απάντηση true ή false ανάλογα με τον αν έχει ανακληθεί ή ανασταλεί το delegation.

Στο σχήμα 5.11 μπορούμε να δούμε την αναλυτική πορεία του ελέγχου. Εδώ έχουμε ένα query από τον peer προς το chaincode. Στη συνέχεια το chaincode ζητά από το ledger τα δεδομένα για το delegation. Όταν επιστρέφονται τα δεδομένα, αν το delegation δεν υπάρχει, τότε εμφανίζεται σχετικό, ενημερωτικό μήνυμα στον peer. Σε αντίθετη περίπτωση το chaincode εξετάζει τις μεταβλητές Suspended και Revoked αντίστοιχα. Εάν είναι true τότε το μήνυμα αυτό εμφανίζεται στον peer και τον ενημερώνει σχετικά. Εάν δεν είναι, τότε επιστρέφεται το μήνυμα false από το chaincode προς τον peer.

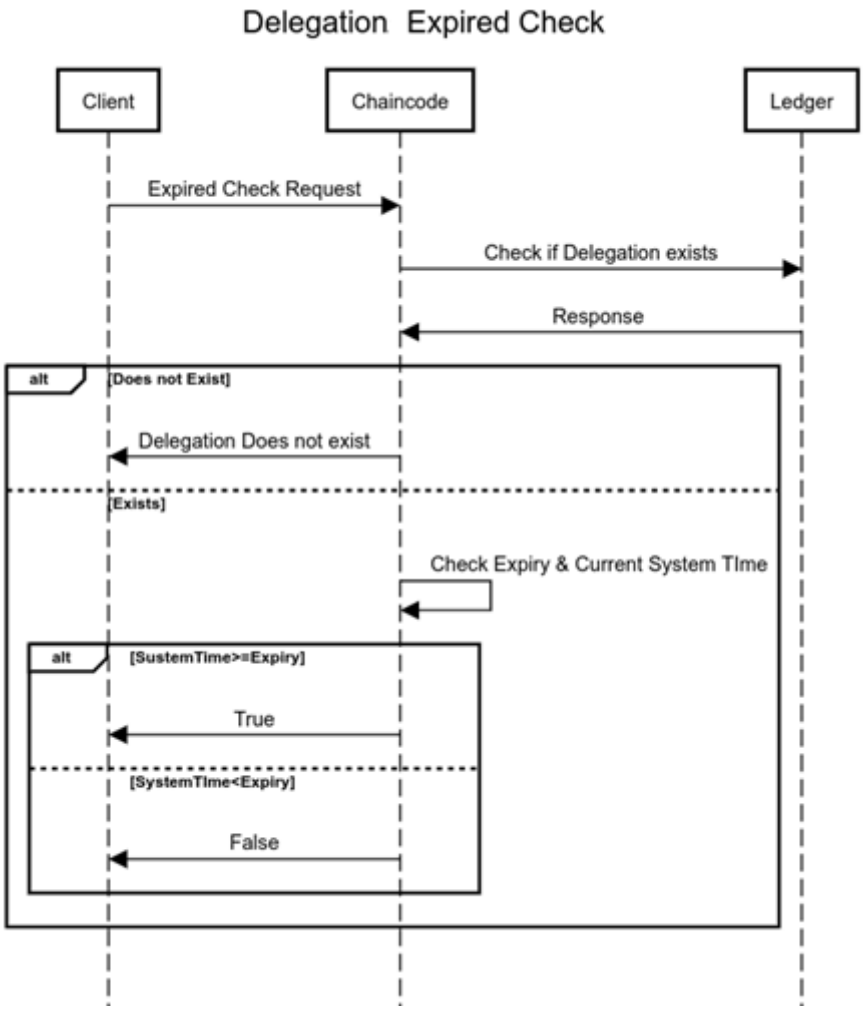


Σχήμα 5.11 - Έλεγχος Αναστολής/Ανάκλησης Delegation

Έλεγχος χρονικής λήξης Delegation

Αντίστοιχη λογική με πριν. Ο χρήστης δίνει ως όρισμα το Pck του delegation που θέλουμε να ελέγξουμε εάν έχει λήξη η χρονική ισχύς του.

Άλλο ένα query από τον peer προς το chaincode. Όπως βλέπουμε και στο σχήμα 5.12, μετά τον καθορισμένο έλεγχο προκειμένου να δούμε αν το delegation υπάρχει, όπου αν δεν υπάρχει επιστρέφεται ένα μήνυμα σφάλματος στον peer που τον ενημερώνει κατάλληλα, το chaincode αποκτά την τοπική ώρα του συστήματος και τη συγκρίνει με τη μεταβλητή Expiry. Με αυτό τον τρόπο αποφασίζει εάν ένα delegation έχει λήξει, οπότε και επιστρέφει την τιμή true στον peer, αλλιώς το delegation είναι ακόμη χρονικά ενεργό, συνεπώς το chaincode επιστρέφει την τιμή false στον peer.



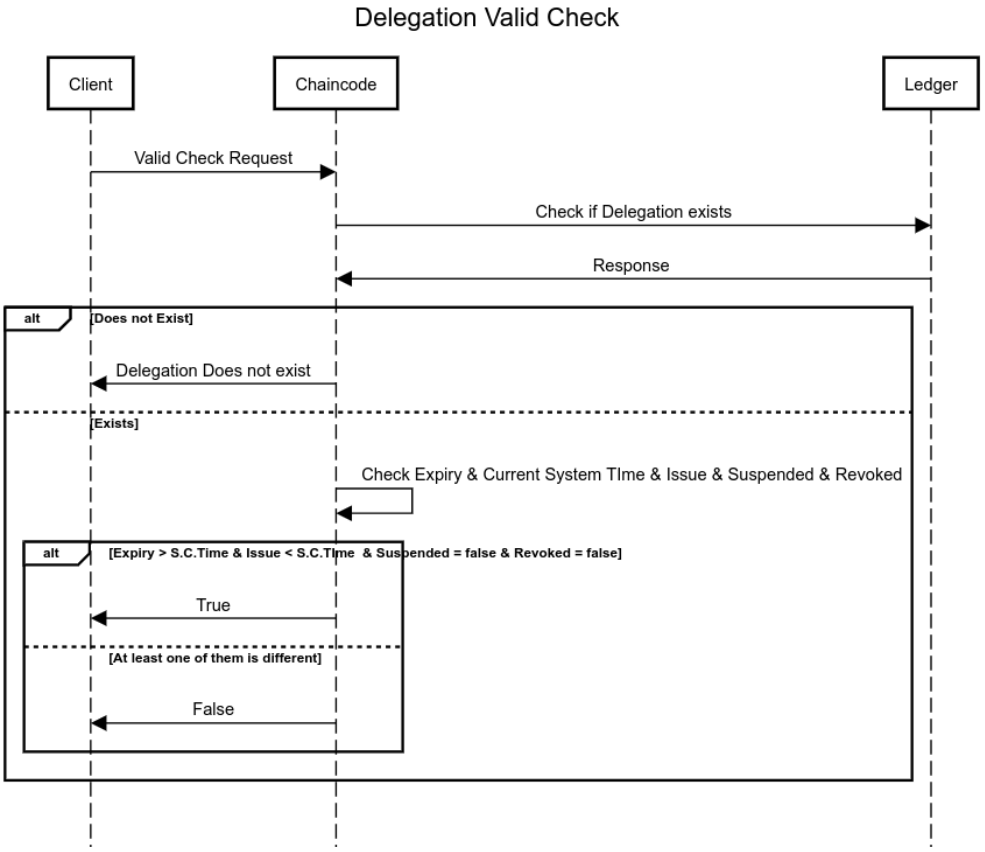
Σχήμα 5.12 - Έλεγχος χρονικής λήξης Delegation

Έλεγχος εγκυρότητας Delegation

Η συνάρτηση αυτή αποτελείται από την συνδυαστική λογική των προηγούμενων δύο ελέγχων, ελέγχει εάν έχει ανακληθεί, ανασταλεί και λήξει, με τη προσθήκη του ελέγχου εκκίνησης του delegation, καθώς εάν δεν έχει εκκινήσει ακόμη, τότε δε το θεωρούμε σε ισχύ.

Το query αυτό, εκκινεί με τον peer να δίνει ως όρισμα το Pck του delegation που θέλουμε να ελέγξουμε. Στη συνέχεια το chaincode πηγαίνει στο ledger και αναζητεί τα δεδομένα του. Το ledger στέλνει την απάντηση του, και αν δεν βρει δεδομένα που να αντιστοιχούν στο Pck που δόθηκε, τότε ο peer ενημερώνεται με αντίστοιχο μήνυμα από το chaincode και η διαδικασία σταματάει. Αντίθετα, εάν βρεθούν τα στοιχεία γίνεται έλεγχος των μεταβλητών αναστολής και ανάκλησης, ενώ οι χρόνοι έναρξης και λήξης συγκρίνονται με τον τοπικό χρόνο του συστήματος που τρέχει η εφαρμογή μας. Σε περίπτωση που έστω και ένα από αυτά δεν είναι ενεργό, το chaincode επιστρέφει στον peer τη τιμή false,

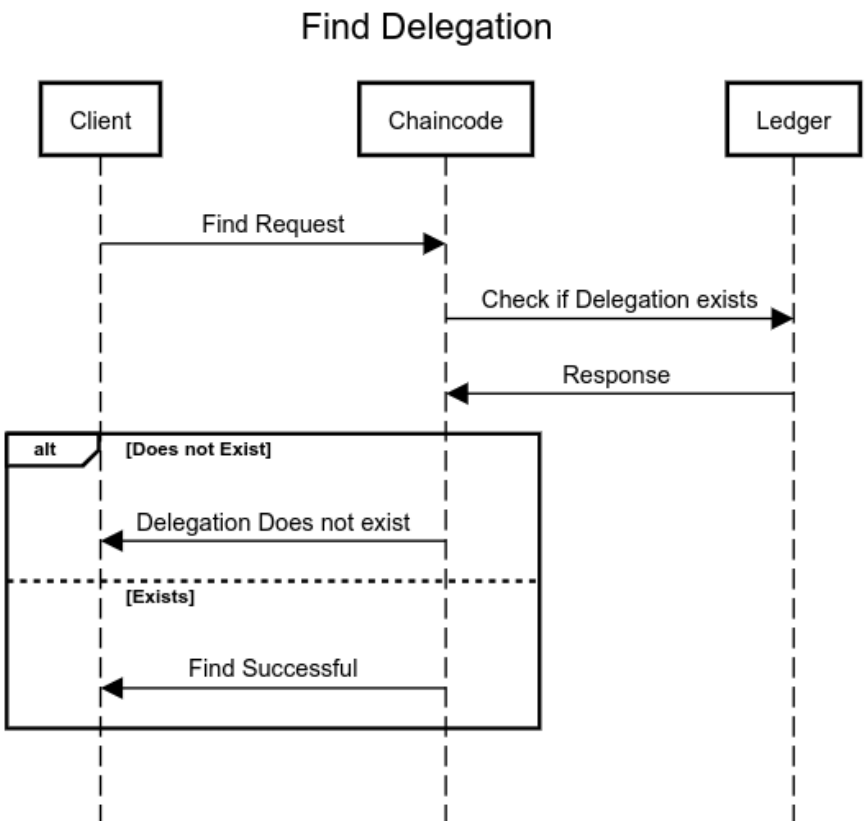
ενημερώνοντας τον πως το delegation αυτό δεν είναι ενεργό. Σε αντίθετη περίπτωση, του επιστρέφει την τιμή true, όπως παρατηρούμε και στο σχήμα 5.13.



Σχήμα 5.13 - Έλεγχος εγκυρότητας Delegation

Αναζήτηση Delegation

Εδώ έχουμε την αντίστοιχη συνάρτηση με τους tenants και τα services. Έχουμε λοιπόν ένα query από τον peer, ο οποίος δίνει ως όρισμα το Pck του delegation που αναζητεί. Στη συνέχεια, το chaincode πηγαίνει στο ledger και με βάση το Pck ζητά τα δεδομένα για το delegation. Το ledger απαντά στο chaincode, ενώ στη συνέχεια, αν το delegation δε βρεθεί θα επιστραφεί στον peer αντίστοιχο μήνυμα που θα τον ενημερώνει, ενώ αν βρεθεί, το chaincode θα επιστρέψει στον peer τα δεδομένα που πήρε από το ledger. Η διαδικασία περιγράφεται στο σχήμα 5.14.



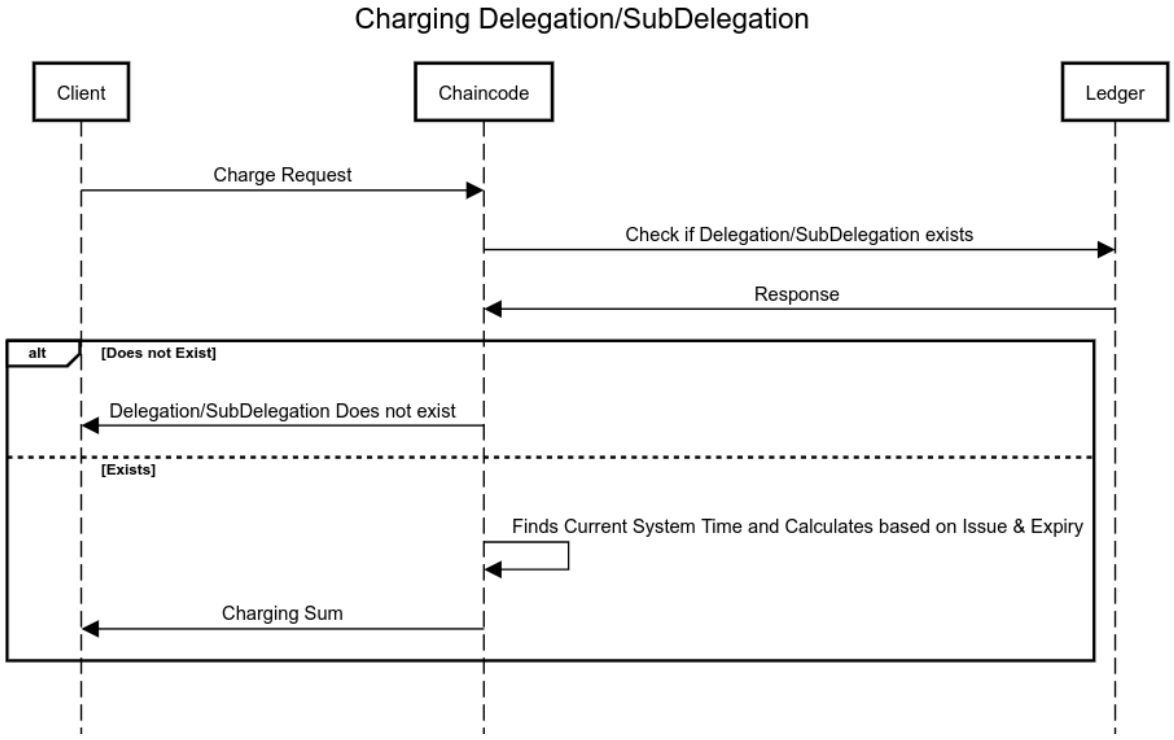
Σχήμα 5.14 - Αναζήτηση Delegation

Συνάρτηση χρέωσης

Για τη διαδικασία της χρέωσης έχουμε κάνει κάποιες συμβάσεις. Έχουμε ορίσει πως η χρέωση θα γίνεται με βάση τον αριθμό των πυρήνων (cores) που χρησιμοποιούνται για το delegation και τη χρονική διάρκεια. Ο τρόπος χρέωσης είναι ενδεικτικός και σκοπός του είναι να παρουσιάσει τη δυνατότητα χρέωσης, ενώ μπορεί πολύ εύκολα να τροποποιηθεί προκειμένου να ανταποκρίνεται στις ιδιαίτερες ανάγκες της κάθε εφαρμογής. Η συνάρτηση χρέωσης είναι κοινή για τα delegation και τα subdelegation και βλέπουμε τη πορεία της στο σχήμα 5.15.

Στο query αυτό, ο peer δίνει ως όρισμα στη συνάρτηση του chaincode το Pck του delegation ή του subdelegation για το οποίο επιθυμεί να ενημερωθεί για τη χρέωση ως τη στιγμή της ερώτησης. Η χρέωση ενημερώνεται δυναμικά, και εξαρτάται και από τη χρονική στιγμή που κάνει την ερώτηση ο peer σε σχέση με την έναρξη και λήξη. Το chaincode μόλις λάβει τα δεδομένα θα αναζητήσει στο ledger τα δεδομένα. Το ledger θα επιστρέψει στο chaincode με τα δεδομένα. Εάν δεν βρεθούν στο ledger, τότε το chaincode θα απαντήσει στον peer με ένα μήνυμα σφάλματος, το οποίο θα τον ενημερώσει σχετικά. Σε αντίθετη περίπτωση, το chaincode θα πραγματοποιήσει τους εσωτερικούς του υπολογισμούς με βάση τον αριθμών

των πυρήνων που δόθηκε αλλά και την ώρα του συστήματος σε σχέση με τις μεταβλητές Issue και Expiry. Στη συνέχεια θα επιστρέψει στον peer τη χρέωση που υπολογίστηκε.



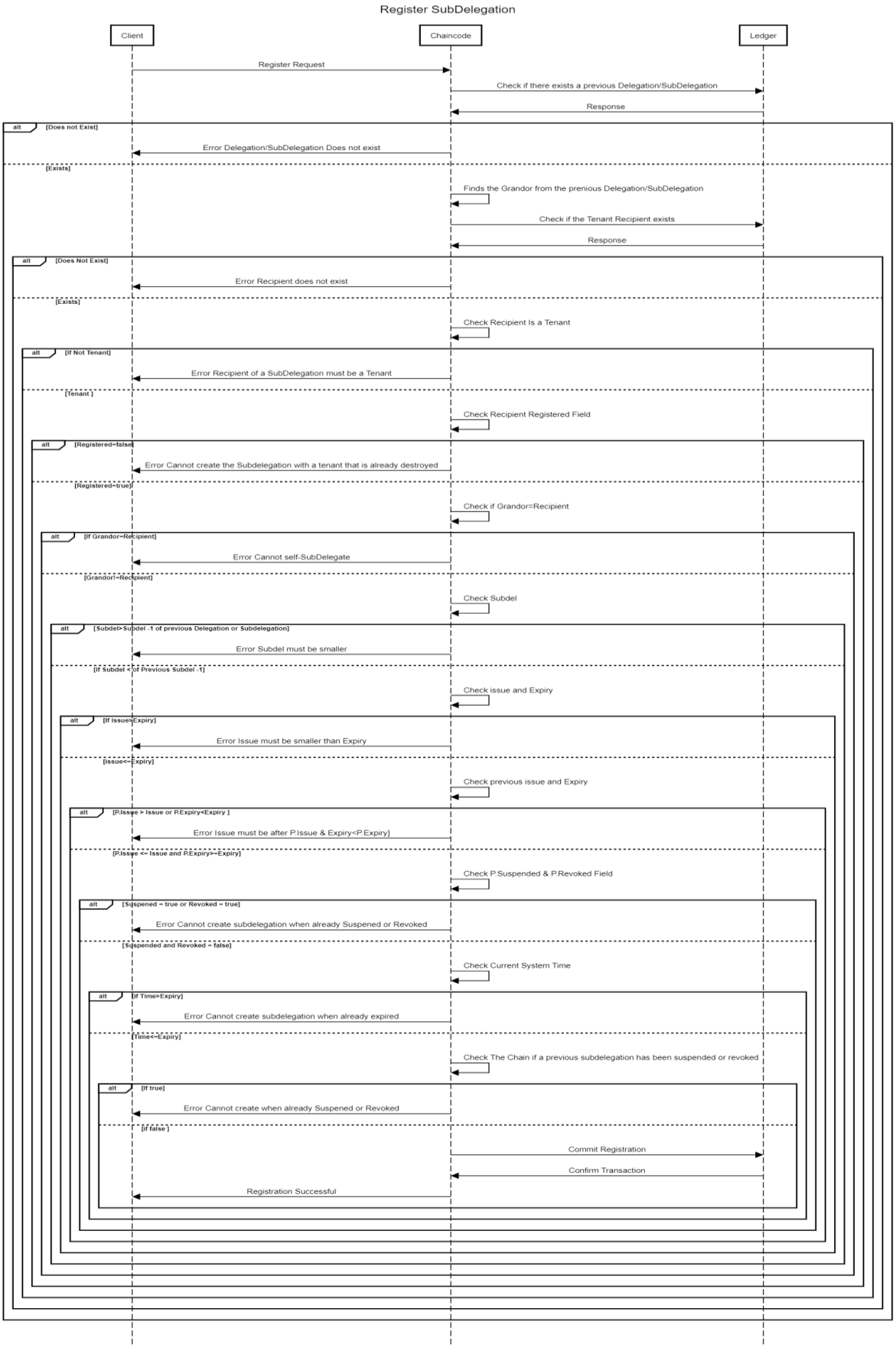
Σχήμα 5.15 - Χρέωση Delegation/Subdelegation

Εγγραφή Subdelegation

Η εγγραφή εδώ έχει μερικές διαφοροποιήσεις σε σχέση με την εγγραφή των delegations. Συγκεκριμένα, ως ορίσματα ορίσματα έχουμε το Pck του subdelegation, το Pck του αποδέκτη, τον αριθμό των διαθέσιμων subdelegations, καθώς και την έναρξη και λήξη του, αλλά αντί για τον Pck του εντολέα, έχουμε το Pck του προηγούμενου delegation ή subdelegation με το οποίο συνδέεται. Τα subdelegations σχηματίζουν ουσιαστικά μια αλυσίδα μεταξύ τους μαζί με το αρχικό delegation από το οποίο προέρχονται τα δικαιώματα που εκχωρούνται. Τα υπόλοιπα πεδία συμπληρώνονται αυτόματα.

Αναλυτικά, όπως βλέπουμε και στο σχήμα 5.16, εδώ ο peer κάνει invoke το chaincode περνώντας τα ορίσματα που αναφέραμε. Στη συνέχεια, το chaincode θα πάει στο ledger και θα ζητήσει τα στοιχεία του προηγούμενο delegation ή subdelegation. Στη συνέχεια το ledger θα επιστρέψει στο chaincode το αποτέλεσμα της αναζήτησης. Εάν δεν υπάρχει, τότε θα ενημερωθεί ο peer από το chaincode και η διαδικασία θα σταματήσει. Εάν υπάρχει, τότε μέσω των δεδομένων αυτών το chaincode θα βρει τον grantor. Στη συνέχεια, θα ελέγξει εάν

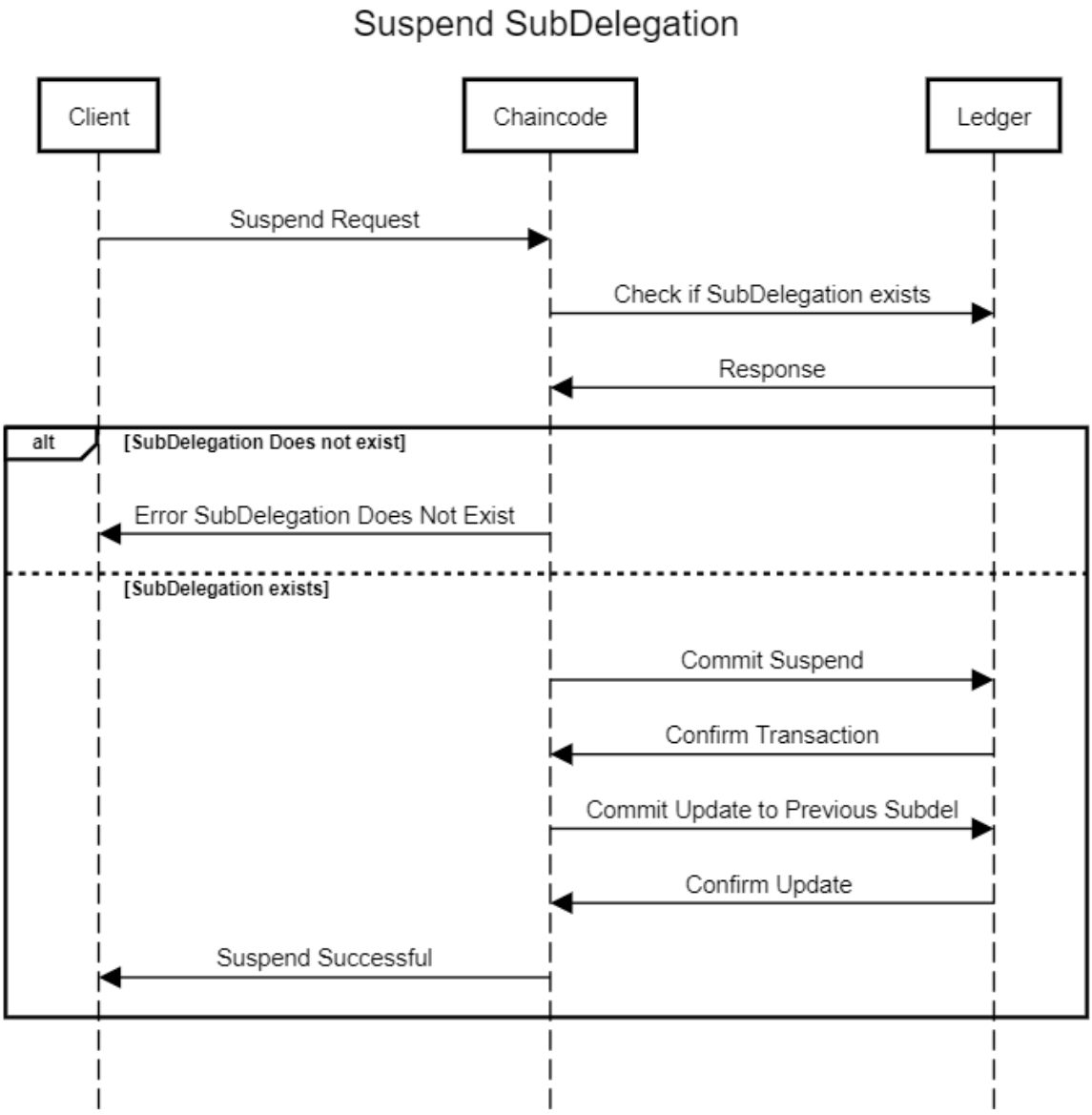
ο recipient υπάρχει μέσω του ledger. Ανάλογα την απάντηση που θα πάρει από το ledger είτε θα τερματίσει τη διαδικασία εάν δεν είναι επιστρέφοντας στον peer το κατάλληλο μήνυμα, είτε θα συνεχίζει κοιτάζοντας από τα δεδομένα του, εάν είναι tenant, καθώς όπως είπαμε, δεν επιτρέπονται subdelegations σε services. Ακολουθεί η ίδια διαδικασία ανάλογα με το αν είναι ή όχι, ενώ στη συνέχεια ελέγχουμε εάν είναι registered ή αν έχει διαγραφεί. Συνεχίζοντας με την ίδια λογική, και εφόσον δεν έχει διαγραφεί ελέγχουμε μήπως ο grantor είναι ο recipient, καθώς αυτό δεν επιτρέπεται. Εάν και εδώ δεν αντιμετωπίσουμε κάποιο πρόβλημα συνεχίζουμε με τον έλεγχο του subdel, όπου ελέγχουμε εάν μπορεί να δοθεί το δικαίωμα τόσων subdelegation, με βάση τα subdelegation που έχει διαθέσιμα το προηγούμενο στην αλυσίδα delegation ή subdelegation. Εάν δεν επαρκούν, η διαδικασία θα σταματήσει και θα επιστραφεί το κατάλληλο μήνυμα σφάλματος στον peer. Εάν επαρκούν, θα συνεχίσουμε με τον έλεγχο των μεταβλητών Issue και Expiry, όπου αν δεν αντιμετωπίσουμε εκεί κάποιο θέμα σε σχέση με τον χρόνο υποβολής του αιτήματος δημιουργίας του subdelegation, θα συνεχίσουμε με τον έλεγχο των ίδιων μεταβλητών σε σχέση όμως με τις μεταβλητές του προηγούμενου στη λίστα delegation/subdelegation, καθώς δεν επιτρέπεται να ξεκινά το επόμενο πριν ξεκινήσει αυτό από το οποίο προέρχεται, αλλά και δεν επιτρέπεται να λήγει μετά τη λήξη του προηγούμενου. Τέλος, κοιτάμε αν κάποιο από τα προηγούμενα στην αλυσίδα έχουν γίνει revoked ή suspended. Αν ναι, η διαδικασία σταματάει με το αντίστοιχο μήνυμα. Αν όχι, τότε η εγγραφή υποβάλλεται στο ledger, αυτό την επιβεβαιώνει πίσω στο chaincode, και το αντίστοιχο μήνυμα επιτυχούς υποβολής εμφανίζεται στον peer.



Σχήμα 5.16 - Εγγραφή Subdelegation

Αναστολή Subdelegation

Η διαδικασία που ακολουθείται εδώ, είναι όμοια με αυτή του delegation. Η μοναδική διαφορά, είναι πως όταν γίνεται η αναστολή, στη συνέχεια ενημερώνεται και το Subdel πεδίο του προηγούμενου, προκειμένου να αυξηθεί κατά τον αριθμό των διαθέσιμων Sudbel που είχε το προς αναστολή subdelegation προσαυξημένο κατά ένα, καθώς και το ίδιο πιάνει μία θέση.



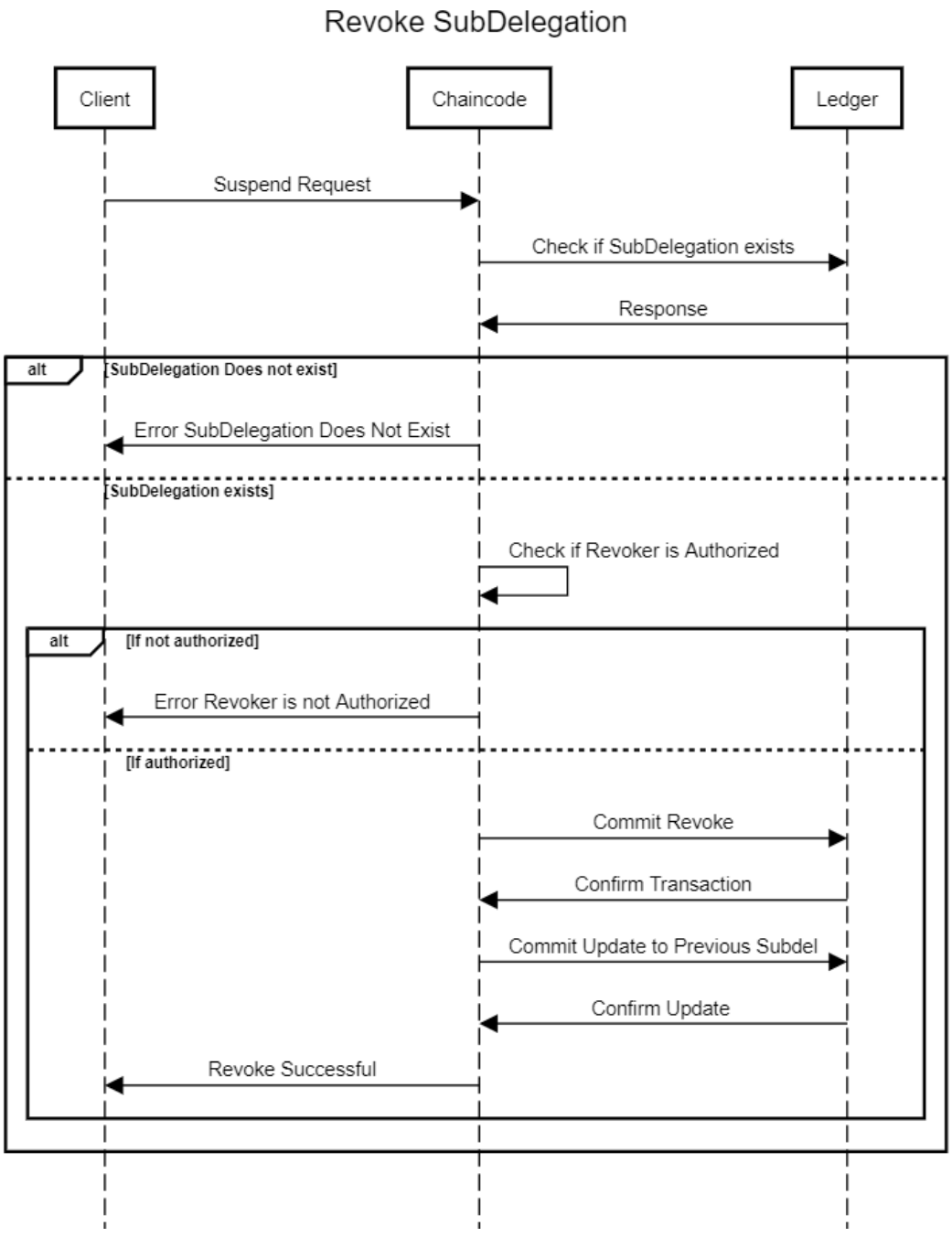
Σχήμα 5.17 - Αναστολή Subdelegation

Συνεπώς, όπως αναφέραμε και πριν, η διαδικασία είναι όμοια με πριν, όμως τώρα, πριν επιστρέψει στον peer το μήνυμα επιτυχούς αναστολής, όπως βλέπουμε και στο σχήμα 5.17, το chaincode υποβάλει την ενημέρωση του Subdel του προηγούμενου στο ledger, ενώ μόλις

πάρει την επιβεβαίωση από αυτό, τότε επιστρέφει στον peer το μήνυμα επιτυχούς αναστολής.

Ανάκληση Subdelegation

Η λογική πίσω από την συνάρτηση αυτή είναι όμοια με αυτή ενός delegation, σε συνδυασμό με την ενημέρωση του Subdel του προηγούμενου όπως πριν.



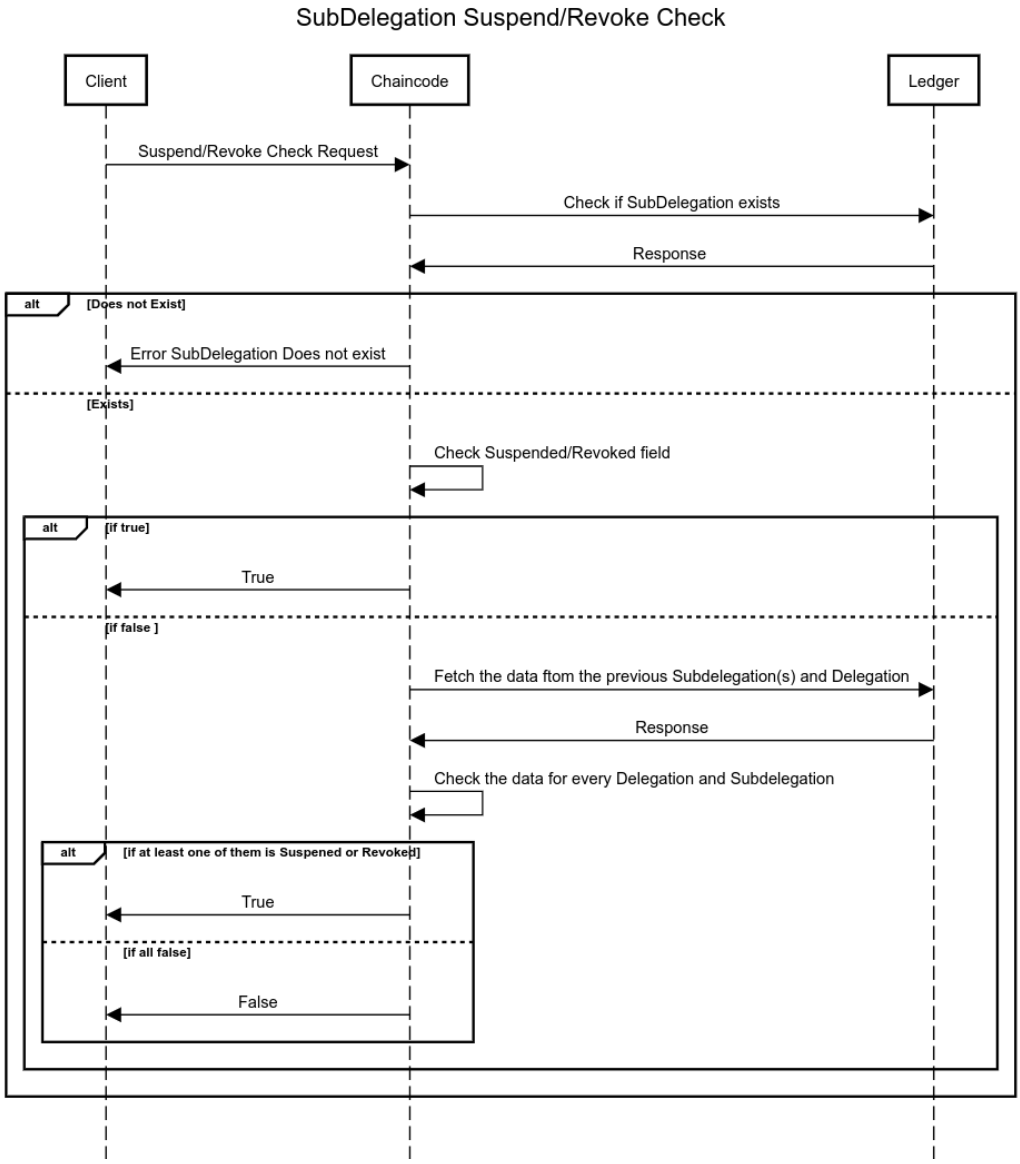
Σχήμα 5.18 - Ανάκληση Subdelegation

Η διαδικασία εδώ ξεκινάει με τον peer να κάνει invoke το chaincode. Στη συνέχεια, όπως βλέπουμε και στο σχήμα 5.18, το chaincode ελέγχει την ύπαρξη του Subdelegation στο ledger. Στη συνέχεια, το ledger επιστρέφει στο chaincode την απάντηση και αν δεν υπάρχει, το κατάλληλο μήνυμα εμφανίζεται στον peer ως απάντηση από το chaincode και η διαδικασία σταματάει. Εάν υπάρχει, τότε το chaincode ελέγχει αν ο revoker που δόθηκε είναι μέσα στη λίστα με τους εξουσιοδοτημένους revokers. Αν όχι, τότε και πάλι επιστρέφεται μήνυμα σφάλματος και η διαδικασία σταματάει, σε αντίθετη περίπτωση, η ανάκληση υποβάλλεται στο ledger. Το ledger την επιβεβαιώνει και στην συνέχεια υποβάλλεται η ενημέρωση του προηγούμενου subdel. Αφού επιβεβαιωθεί και αυτή η υποβολή, το chaincode επιστρέφει στον peer ένα μήνυμα επιτυχούς ανάκλησης.

Έλεγχος αναστολής και ανάκλησης

Εδώ έχουμε και πάλι δύο συναρτήσεις που θα τις ομαδοποιήσουμε για να τις παρουσιάσουμε. Έχουν ως όρισμα το Pck του subdelegation που θέλουμε να ελέγξουμε, και μόλις ανακτηθούν από το ledger τα στοιχεία του, ελέγχουμε τις μεταβλητές “Revoked” και “Suspended” αντίστοιχα. Σε αντίθεση όμως με την διαδικασία που ακολουθήσαμε στα delegations, εδώ δε μπορούμε να απαντήσουμε ακόμη με true ή false καθώς πρέπει να διατρέξουμε όλη την αλυσίδα που σχηματίζεται προκειμένου να ελέγξουμε αν έχει ανακληθεί ή ανασταλεί κάποιο subdelegation ή το delegation σε προηγούμενο επίπεδο της αλυσίδας. Αυτό συμβαίνει διότι αν για παράδειγμα ανακληθεί ένα subdelegation στο επίπεδο N τότε όλα τα subdelegations που προέρχονται από αυτό και είναι στο επίπεδο N+1 και μετά ανακαλούνται και αυτά αυτόματα.

Σύμφωνα με την εικόνα 5.19 αρχικά ο peer κάνει query τη συνάρτηση, και για τις δύο συναρτήσεις η διαδικασία είναι όμοια. Στη συνέχεια γίνεται ο γνωστός έλεγχος αν υπάρχει το subdelegation. Εάν όχι, τότε μόλις λάβει την απάντηση από το ledger, το chaincode στέλνει στον peer το αντίστοιχο μήνυμα και η διαδικασία σταματάει. Αν υπάρχει, τότε το chaincode ελέγχει το πεδίο Suspended/Revoked, αν αυτό είναι ίσο με true τότε επιστρέφει στον peer τη τιμή true και σταματάει και πάλι η διαδικασία. Αν όχι, τότε το chaincode πηγαίνει και πάλι στο ledger και αναζητά τα στοιχεία για τα προηγούμενα σε επίπεδο subdelegations ή delegation. Εάν βρει έστω ένα true τότε επιστρέφει τη τιμή true στον peer, σε αντίθετη περίπτωση επιστρέφει τη τιμή false.



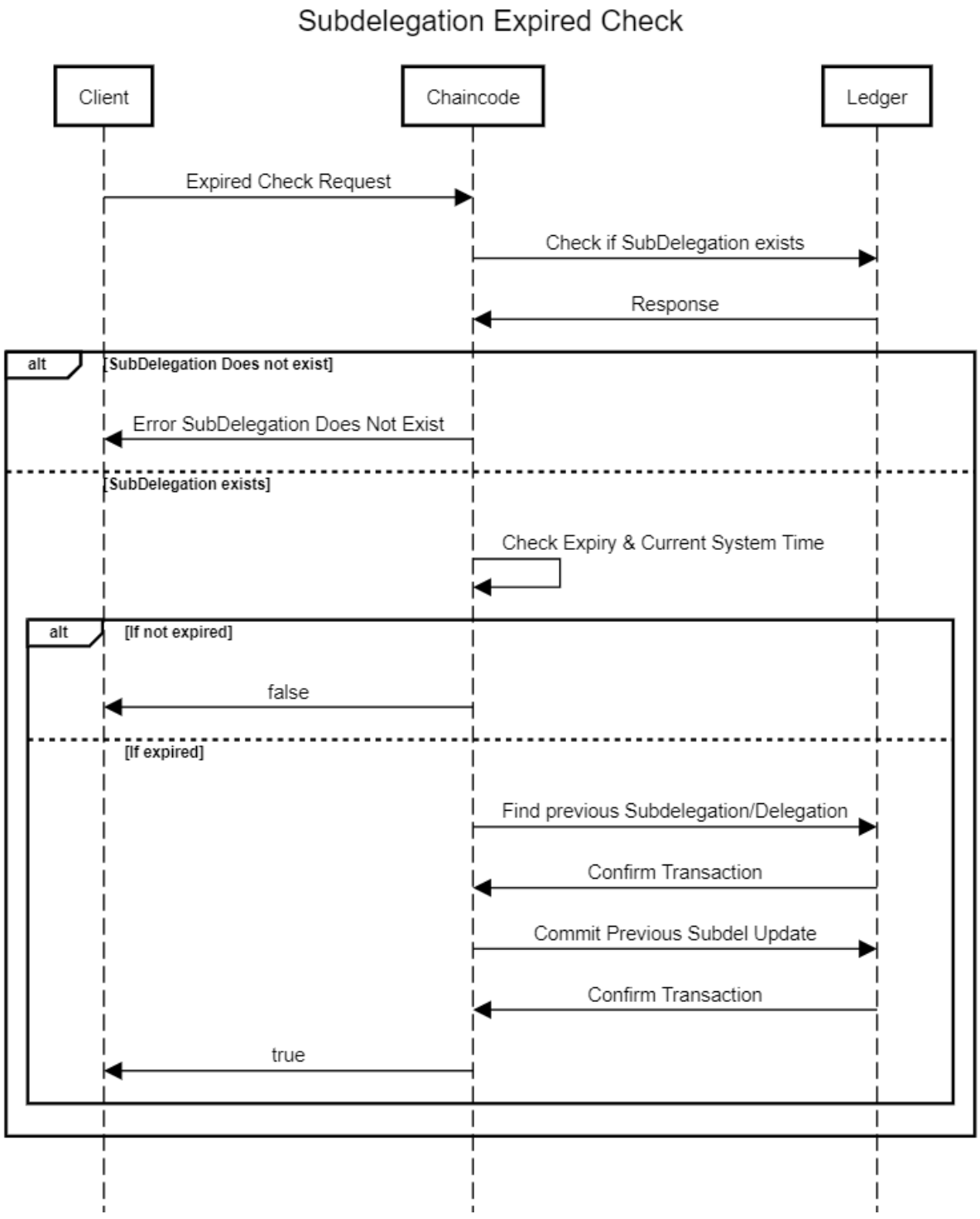
Σχήμα 5.19 - Έλεγχος αναστολής/ανάκλησης Subdelegation

Έλεγχος χρονικής εγκυρότητας Subdelegation

Η διαφοροποίηση σε σχέση με τη συνάρτηση για τα delegation έγκειται στο γεγονός πως, εδώ, αν έχει λήξει ένα subdelegation, αυξάνουμε το subdel του προηγούμενου σύμφωνα με το subdel του ληγμένου επαυξημένο κατά ένα.

Αναλυτικά, όπως βλέπουμε και στο σχήμα 5.20, έχουμε πως εδώ ο peer κάνει invoke το chaincode. Το chaincode με τη σειρά του ελέγχει εάν υπάρχει πράγματι το subdelegation που είναι προς εξέταση. Μόλις πάρει την απάντηση από το ledger, εάν δεν υπάρχει τότε επιστρέφει μήνυμα σφάλματος προς τον peer. Αν όμως υπάρχει, τότε ελέγχει το τοπικό χρόνο του συστήματος και το συγκρίνει με τη μεταβλητή Expiry. Εάν δεν έχει λήξει, τότε επιστρέφει στον peer την ένδειξη false και εκεί σταματάει η διαδικασία. Αντίθετα, αν έχει

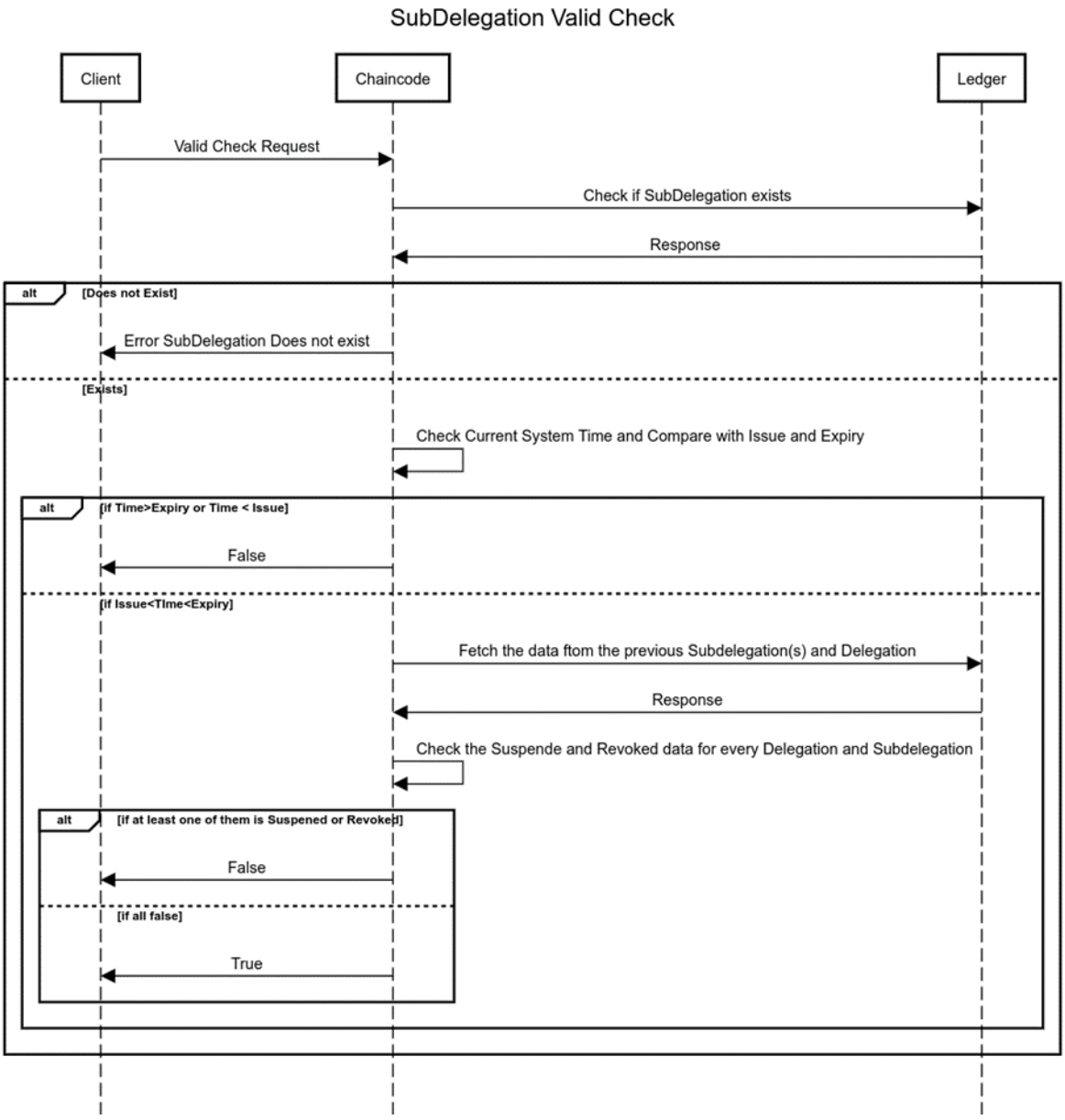
λήξει, τότε το chaincode πηγαίνει στο ledger και ζητάει τα στοιχεία του προηγούμενου delegation/subdelegation. Μόλις το ledger τα επιστρέψει, το chaincode υποβάλει τα ενημερωμένα στοιχεία, με το αυξημένο subdel. Στη συνέχεια η συναλλαγή επιβεβαιώνεται προς το chaincode και αυτό επιστρέφει στον peer την απάντηση true.



Σχήμα 5.20 - Έλεγχος χρονικής εγκυρότητας Subdelegation

Έλεγχος Εγκυρότητας Subdelegation

Η συνάρτηση αυτή παίρνει ως όρισμα το Pck του subdelegation που θέλουμε να ελέγξουμε ως προς την εγκυρότητα του. Η διαδικασία ξεκινά με όμοιο τρόπο με αυτόν που ακολουθούμε για τα delegations, όπως παρατηρούμε και στην εικόνα 5.21. Στη συνέχεια όμως, γίνεται ο αντίστοιχος έλεγχος που περιγράφηκε και πριν σχετικά με την ανάκληση και την αναστολή σε όλα τα προηγούμενα επίπεδα της αλυσίδας. Η επιστροφή είναι true ή false.



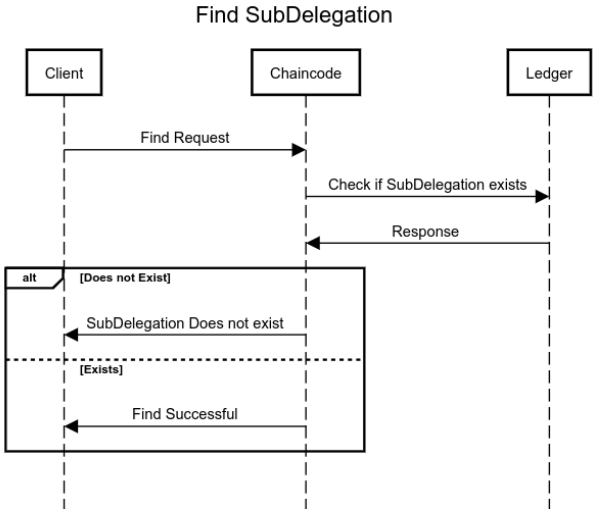
Σχήμα 5.21 – Έλεγχος εγκυρότητας Subdelegation

Η λειτουργία αυτή ξεκινάει με τον peer να πραγματοποιεί ένα query προς το chaincode δίνοντας του τα κατάλληλα ορίσματα. Στη συνέχεια, το chaincode πηγαίνει στο ledger και ζητάει τα στοιχεία του Subdelegation. Εάν το Subdelegation δεν υπάρχει στο ledger, τότε το chaincode επιστρέφει ενημερωτικό μήνυμα σφάλματος στον peer και η διαδικασία τερματίζει. Αντίθετα, εάν υπάρχει τότε γίνεται αρχικά έλεγχος με την ώρα του συστήματος προκειμένου να διαπιστωθεί έχει λήξει. Εάν ναι, τότε η επιστροφή του chaincode στον peer είναι false. Σε περίπτωση που δεν έχει λήξει όμως, το chaincode ζητάει ξανά από το ledger, αναδρομικά τα στοιχεία και των προηγούμενων Subdelegation και delegation που συνδέονται με το προς εξέταση delegation. Η διαδικασία ελέγχου επαναλαμβάνεται, και αν δεν έχει γίνει revoked ή suspended το ίδιο ή κάποιο από τα προηγούμενά του, τότε στον peer επιστρέφεται η τιμή true. Αλλιώς θα επιστραφεί η τιμή false.

Αναζήτηση Subdelegation

Η τελευταία συνάρτηση με την οποία θα ασχοληθούμε είναι αυτή της αναζήτησης ενός Subdelegation. Η διαδικασία είναι ίδια με αυτή του delegation, όπως παρατηρούμε και στο σχήμα 5.22.

Είναι ένα query που πραγματοποιεί ο peer δίνοντας ως όρισμα το Pck του Subdelegation που επιθυμεί να αναζητήσει. Το chaincode στη συνέχεια πηγαίνει στο ledger για να πάρει τα δεδομένα. Μόλις ανακτηθούν τα δεδομένα από το ledger, σε περίπτωση που δε βρεθεί το Subdelegation θα επιστραφεί μήνυμα σφάλματος που θα ενημερώνει σχετικά τον peer, αλλιώς το chaincode θα επιστρέψει τα δεδομένα στον peer.



Σχήμα 5.22 – Αναζήτηση Subdelegation

6. Επίλογος και Μελλοντικές επεκτάσεις

Στο κεφάλαιο αυτό θα παρουσιαστεί μία σύνοψη για το πρόγραμμα μας και μερικές ιδέες για μελλοντική επέκταση του.

6.1 Σύνοψη και συμπεράσματα

Σκοπός της παρούσας διπλωματικής εργασίας ήταν η εξέταση της αποκεντρωμένης αρχιτεκτονικής του blockchain σε συνδυασμό με το cloud management. Στο πλαίσιο αυτό δημιουργήθηκε μία εφαρμογή που βασίστηκε στο Hyperledger Fabric, ένα private, permissioned blockchain, όπου εκμεταλλευόμενοι την ύπαρξη chaincode δημιουργήσαμε ένα πρόγραμμα διαχείρισης για cloud tenants και services. Βασικό χαρακτηριστικό του συστήματος είναι η δυνατότητα τροποποίησης του chaincode προκειμένου να προσαρμόζεται στις ιδιαίτερες απαιτήσεις της εκάστοτε περίπτωσης, το οποίο εκτείνεται στους tenants, τα services αλλά και τα delegations που δημιουργούνται μεταξύ τους. Αποδείχθηκε πως μπορούν να αυτοματοποιηθούν οι διαδικασίες διαχείρισης, ενώ λόγω της δυνατότητας κλιμάκωσης του συστήματος, αυτό μπορεί να επεκταθεί και ως προς το πλήθος αλλά και ως προς την διαφοροποίηση των χαρακτηριστικών των εμπλεκόμενων μελών. Η δυνατότητα γραφής του chaincode σε γλώσσες οι οποίες μπορεί να είναι ήδη γνωστές σε πολλές περιπτώσεις είναι ένα σημαντικό θετικό στοιχείο για τη συγγραφή του κώδικα, αλλά και για την ανάγνωση και κατανόηση του από οποιονδήποτε ενδιαφερόμενο. Ο συνδυασμός του cloud management με το blockchain μας έδωσε σε ένα πρώτο επίπεδο αξιόπιστα και γρήγορα αποτελέσματα και σίγουρα είναι ένα σημείο που χρήζει περαιτέρω διερεύνησης.

6.2 Μελλοντικές επεκτάσεις

Η εφαρμογή μας υλοποιήθηκε ως ένα chaincode. Σε επόμενο επίπεδο θα μπορούσε να χωριστεί και να εμπλουτιστεί σε πολλαπλά chaincodes, ειδικά από τη στιγμή που αυτό υποστηρίζεται από το Hyperledger Fabric, προκειμένου να δημιουργηθεί η δυνατότητα να γίνει αρθρωτή η δομή του προγράμματος και πιο ευέλικτη στο να προσαρμόζεται στις εξελισσόμενες ανάγκες. Επίσης, για τους ερευνητικούς σκοπούς της εργασίας, το σύστημα βασίστηκε σε δύο peer organizations, συνεπώς σε μελλοντικό επίπεδο θα μπορούσαμε να

εξετάσουμε τη συμπεριφορά του όταν έχουμε πολλούς peers οι οποίοι θα συναλλάσσονται ταυτόχρονα με το ledger και κατά πόσο αυτό θα επιφέρει καθυστερήσεις στο σύστημα. Συνεπώς η απόδοση κατά την κλιμάκωση του συστήματος είναι ένας ακόμη τομέας που μπορούμε να εξετάσουμε στη συνέχεια, ιδιαίτερα σε συνδυασμό με μεγαλύτερης κλίμακας αριθμό tenants και services. Τέλος, έμφαση πρέπει να δοθεί στην ακόμη μεγαλύτερη αυτοματοποίηση των διεργασιών η οποία εξαλείφει το ενδεχόμενο ανθρώπινου λάθους και διευκολύνει τη σωστή και απροβλημάτιστη λειτουργία αλλά και στην αύξηση των διαθέσιμων λειτουργιών προκειμένου να μεγιστοποιηθεί η παραγωγικότητα του προγράμματός μας.

7. Βιβλιογραφία:

[1] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

[2] Cachin, C., & Vukolić, M. (2017). Blockchain consensus protocols in the wild. arXiv preprint arXiv:1707.01873.

[3] Thakur, S., & Breslin, J. G. (2017). A robust reputation management mechanism in federated cloud. IEEE Transactions on Cloud Computing.

[4] Pustchi, N., Patwa, F., & Sandhu, R. (2016, March). Multi cloud iaas with domain trust in openstack. In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy (pp. 121-123).

[5] Kalra, S., & Prabhakar, T. V. (2018, February). Towards dynamic tenant management for microservice based multi-tenant saas applications. In Proceedings of the 11th Innovations in Software Engineering Conference (pp. 1-5).

[6] Sukmana, M. I., Torkura, K. A., Graupner, H., Cheng, F., & Meinel, C. (2019, January). Unified Cloud Access Control Model or Cloud Storage Broker. In 2019 International Conference on Information Networking (ICOIN) (pp. 60-65). IEEE.

[7] Kempf, J., Nayak, S., Robert, R., Feng, J., Deshmukh, K. R., Shukla, A., ... & Sjöberg, J. (2019). The Nubo Virtual Services Marketplace. arXiv preprint arXiv:1909.04934.

[8] Nayak, S., Narendra, N. C., Shukla, A., & Kempf, J. (2018, July). Saranyu: Using smart contracts and blockchain for cloud tenant management. In 2018 IEEE 11th International Conference on Cloud Computing (CLOUD) (pp. 857-861). IEEE.

[9] Al-Bassam, M. (2017, April). SCPKI: a smart contract-based PKI and identity system. In Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts (pp. 35-40).

[10] Rosa, R., & Rothenberg, C. E. (2018). Blockchain-based decentralized applications for multiple administrative domain networking. IEEE Communications Standards Magazine, 2(3), 29-37.

[11] Bozic, N., Pujolle, G., & Secci, S. (2017, October). Securing virtual machine orchestration with blockchains. In 2017 1st Cyber Security in Networking Conference (CSNet) (pp. 1-8). IEEE.

[12] El Ioini, N., & Pahl, C. (2018, October). Trustworthy orchestration of container based edge computing using permissioned blockchain. In 2018 Fifth International Conference on Internet of Things: Systems, Management and Security (pp. 147-154). IEEE.

[13] Hammi, M. T., Hammi, B., Bellot, P., & Serhrouchni, A. (2018). Bubbles of Trust: A decentralized blockchain-based authentication system for IoT. Computers & Security, 78, 126-142.

[14] Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. Ieee Access, 4, 2292-2303.

[15] Hewa, T., Braeken, A., Ylianttila, M., & Liyanage, M. Blockchain-based Automated Certificate Revocation for 5G IoT.

[16] Reantongcome, V., Visoottiviseth, V., Sawangphol, W., Khurat, A., Kashihara, S., & Fall, D. (2020, April). Securing and Trustworthy Blockchain-based Multi-Tenant Cloud Computing. In 2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE) (pp. 256-261). IEEE.

[17] Nguyen, D. C., Pathirana, P. N., Ding, M., & Seneviratne, A. (2019). Integration of blockchain and cloud of things: Architecture, applications and challenges. arXiv preprint arXiv:1908.09058.

[18] Fabric-Contract-API-go: https://github.com/hyperledger/fabric-contract-api-go

[19] Prerequisites: https://hyperledger-fabric.readthedocs.io/en/latest/prereqs.html

[20] Αρχεία εγκατάστασης: https://hyperledger-fabric.readthedocs.io/en/latest/install.html

[21] Cryptogen Tool: https://hyperledger-fabric.readthedocs.io/en/release-2.0/commands/cryptogen.html

[22] Hyperledger CA: https://hyperledger-fabric-ca.readthedocs.io/en/latest/

[23] Docker Compose : https://docs.docker.com/compose/

[24] Configtxgen Tool: https://hyperledger-fabric.readthedocs.io/en/release-2.0/commands/configtxgen.html

[25] Membership Service Providers (MSP): https://hyperledger-fabric.readthedocs.io/en/release-2.0/msp.html