ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανταλλαγή Ευαίσθητων Προσωπικών Δεδομένων στον Τομέα Υγείας με Χρήση του Πλαισίου Αλυσίδας-Κορμού Hyperledger Fabric

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιωάννης Χ. Σφοντούρης

Επιβλέπων : Θεοδώρα Βαρβαρίγου

Καθηγήτρια Ε.Μ.Π

Αθήνα, Ιούλιος 2024

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανταλλαγή Ευαίσθητων Προσωπικών Δεδομένων στον Τομέα Υγείας με Χρήση του Πλαισίου Αλυσίδας-Κορμού Hyperledger Fabric

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιωάννης Χ. Σφοντούρης

Επιβλέπων : Θεοδώρα Βαρβαρίγου

Καθηγήτρια Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Ιουλίου 2024.

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

Αθήνα Ιούλιος 2024

.....

Ιωάννης Χ. Σφοντούρης

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

Περίληψη

Στην σημερινή εποχή της τεχνολογικής ανάπτυξης, η χρήση ψηφιακών μέσων αποτελεί αναπόσπαστο κομμάτι της ζωής μας, επηρεάζοντας σχεδόν κάθε πτυχή της καθημερινότητάς μας. Η προστασία των προσωπικών δεδομένων είναι απαραίτητη για να διασφαλιστεί η ιδιωτικότητα των πληροφοριών που διαμοιράζονται μέσω πληροφοριακών συστημάτων, δικτύων και εφαρμογών. Στο πλαίσιο αυτό, η τεχνολογία blockchain αναδύεται ως μια αξιόπιστη λύση για την ενίσχυση της ασφάλειας των δεδομένων. Με τη δυνατότητά της να προσφέρει αποκεντρωμένη διαχείριση και υψηλό επίπεδο κρυπτογράφησης, μπορεί να προστατεύσει αποτελεσματικά τα προσωπικά δεδομένα από μη εξουσιοδοτημένη πρόσβαση και κακόβουλες επιθέσεις. Συνεπώς, η τεχνολογία αυτή μπορεί να έχει ηγετικό ρόλο στην προστασία των ευαίσθητων προσωπικών δεδομένων ασθενών στον Τομέα της Υγείας. Το προτεινόμενο μοντέλο διαμοιρασμού δεδομένων υγείας της παρούσας εργασίας αντιμετωπίζει αδυναμίες των ιατρικών πρακτικών που ισχύουν στην χώρα μας και αφορούν στην αποστολή και παραλαβή των αποτελεσμάτων διαγνωστικών εξετάσεων ασθενών. Αποτελεί εναλλακτική λύση στις υφιστάμενες πρακτικές, που επί του παρόντος περιορίζονται στην ανταλλαγή των δεδομένων μέσω email και χρήση κωδικών, ενώ στο άμεσο μέλλον προγραμματίζεται η αξιοποίηση του ασφαλούς περιβάλλοντος που προσφέρει το gov.gr για την καλύτερη διαχείριση τους. Η προτεινόμενη λύση βασίζεται στη χρήση του Hyperledger Fabric, τεχνολογία blockchain η οποία μπορεί να προσφέρει μια καινοτόμο προσέγγιση για την επίλυση των παραπάνω ζητημάτων. Μέσω της αξιοποίησης ιδιωτικών συλλογών δεδομένων δίνεται η δυνατότητα στους ασθενείς να έχουν τον έλεγχο των προσωπικών τους πληροφοριών, εξασφαλίζοντας ταυτόχρονα την ιδιωτικότητα και την ασφάλεια αυτών των δεδομένων σε ένα ενοποιημένο δίκτυο που στοχεύει να συνδέσει όλες τις Δομές Υγείας.

Λέξεις κλειδιά: Blockchain, Hyperledger Fabric, Αποτελέσματα Διαγνωστικών Εξετάσεων, Τομέας Υγείας, Ιδιωτικές Συλλογές Δεδομένων, Έξυπνα συμβόλαια

Abstract

In today's age of technological development, the use of digital media is an integral part of our lives, affecting almost every aspect of our daily lives. Data protection is necessary to ensure the privacy of personal information shared through information systems, networks and applications. In this context, blockchain technology is emerging as a reliable solution to enhance data security. With its ability to offer decentralized management and a high level of encryption, it can effectively protect personal data from unauthorized access and malicious attacks. Therefore, this technology can have a leading role in protecting sensitive patient data in the Healthcare Sector. The proposed health data sharing model of this work addresses weaknesses of the medical practices that apply in our country and concern the sending and receiving of patient diagnostic test results. It is an alternative solution to the existing management models of the relevant data which are currently limited to their exchange via email and the use of codes, while in the near future it is planned to utilize the secure environment offered by gov.gr for better management. The proposed solution is based on the use of Hyperledger Fabric, a blockchain technology that can offer an innovative approach to protecting sensitive medical data. Through the utilization of private data collections, patients are given the opportunity to have control over their personal information, while ensuring the privacy and security of this data in a unified network that aims to connect all Healthcare Structures.

Key Words: Blockchain, Hyperledger Fabric, Diagnostic Test Results, Healthcare Sector, Private Data Collections, Smart Contracts

Ευχαριστίες

Ευχαριστώ την καθηγήτριά μου κα. Θεοδώρα Βαρβαρίγου του τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής του Εθνικού Μετσόβιου Πολυτεχνείου, για την υποστήριξη και την καθοδήγηση σε όλη την διάρκεια των πανεπιστημιακών μου σπουδών, όπως και για την εμπιστοσύνη και την ελευθερία που μου παρείχε να ερευνήσω και να εξειδικεύσω το αντικείμενό της διπλωματικής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω όλα τα μέλη του εργαστηρίου και ιδιαίτερα τους Αντώνη Λίτκε και Ευθύμιο Χονδρογιάννη για τον χρόνο που αφιέρωσαν και την καθοδήγηση που μου παρείχαν.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και όλους τους αγαπημένους μου ανθρώπους για την υποστήριξη και την έμπνευση που μου έδωσαν καθ' όλη την διάρκεια της πανεπιστημιακής μου πορείας.

Αθήνα, Ιούλιος 2024

Ιωάννης Σφοντούρης

Πίνακας Περιεχομένων

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Η ταχύτατη εξέλιξη της τεχνολογίας και η αυξανόμενη ανάγκη για ασφαλή διαχείριση δεδομένων έχουν δημιουργήσει ένα δυναμικό περιβάλλον που απαιτεί νέες προσεγγίσεις και καινοτομίες. Η τεχνολογία blockchain αποτελεί μια από τις πιο υποσχόμενες εναλλακτικές στις παραδοσιακές πρακτικές διαχείρισης δεδομένων, προσφέροντας ανώτερη ασφάλεια, διαφάνεια και ακεραιότητα. Στην εποχή του Internet-of-Things (IoT) και του Cloud Computing αναδεικνύεται η σημασία της δημιουργίας ευέλικτων και αποτελεσματικών πρακτικών, θέτοντας προκλήσεις όσον αφορά την προστασία των ευαίσθητων προσωπικών δεδομένων και την ταυτοποίηση των οντοτήτων πίσω από την ανωνυμία του διαδικτύου.

Το υπολογιστικό νέφος (Cloud Computing) αποτελεί την πρακτική της αξιοποίησης ενός δικτύου από απομακρυσμένους διακομιστές, οι οποίοι φιλοξενούνται στο διαδίκτυο για την αποθήκευση, διαχείριση και επεξεργασία δεδομένων έναντι της παραδοσιακής προσέγγισης σε τοπικούς διακομιστές ή προσωπικούς υπολογιστές. Επιτρέπει την δημιουργία εφαρμογών και υπηρεσιών οι οποίες παρέχονται στον τελικό χρήστη μέσω του διαδικτύου [1]. Συνεπώς, με την βελτίωση της ποιότητας των υποδομών των δικτύων η χρήση υπηρεσιών που βασίζονται στο cloud φέρνει επανάσταση στον τρόπο που οι επιχειρήσεις και οι οργανισμοί μπορούν να διαχειρίζονται και να επεξεργάζονται δεδομένα. Παρ ’όλα αυτά, θέτει νέους προβληματισμούς για την ασφαλή διαχείριση αυτών των δεδομένων, καθώς αν και αποτελεί μια ευέλικτη λύση είναι επιρρεπής σε κακόβουλες επιθέσεις. Επιπλέον, πρακτικές όπως το KYC (Know-your-Customer) που αποσκοπεί στην ταυτοποίηση του φυσικού προσώπου που βρίσκεται πίσω από έναν υπολογιστή, έχουν αρχίσει να είναι ευρέως διαδεδομένες ιδίως σε τραπεζικά συστήματα και υπηρεσίες που είναι σημαντική αυτή η ταυτοποίηση [2].

Στο πλαίσιο αυτό, το blockchain δίνει την δυνατότητα εύρεσης λύσεων και βελτίωσης των ανωτέρω πρακτικών. Οι Nikolas Kapsoulis κ.α. στο ερευνητικό άρθρο με τίτλο «*Know Your Customer (KYC) Implementation with Smart Contracts on a Privacy-Oriented Decentralized Architecture*» το 2020 [2], επισημαίνουν ότι η αξιοποίηση της αποκεντρωμένης αρχιτεκτονικής που προσφέρει το blockchain συμβάλλει σημαντικά στην αποδοτικότητα και την χρονική βελτιστοποίηση των λειτουργιών του KYC, παρέχοντας ένα ασφαλές περιβάλλον διαχείρισης αυτών των πληροφοριών. Οι C. V. N. U. B. Murthy κ.α. στο ερευνητικό άρθρο με τίτλο «*Blockchain Based Cloud Computing: Architecture and Research Challenges*» το 2020 [1] καταλήγουν πως η αξιοποίηση του blockchain σε συνδυασμό με το Cloud Computing συμβάλλει σημαντικά στην βελτίωση της ασφάλειας και διαχείρισης των δεδομένων, ενώ παράλληλα επιτρέπει μεγαλύτερη χρηστικότητα, εμπιστοσύνη και επεκτασιμότητα.

Παρατηρούμε επομένως πως το blockchain μπορεί να συμβάλει θετικά στο να βελτιώσει υπάρχουσες πρακτικές, όπως και να καινοτομήσει για την δημιουργία νέων μεθόδων και πρακτικών σε μια καθημερινότητα η οποία μετασχηματίζεται συνεχώς με νέες τεχνολογίες οι οποίες ενσωματώνονται σε όλο και περισσότερες πτυχές της ζωής μας.

Ειδικότερα στον τομέα Υγείας όπου η προστασία των προσωπικών δεδομένων αποτελεί ζήτημα ύψιστης σημασίας, η ανεύρεση νέων προσεγγίσεων και τεχνικών αποτελεί αντικείμενο εντατικής έρευνας. Το blockchain μπορεί να παίξει καθοριστικό ρόλο στην βελτίωση των μέχρι σήμερα εφαρμοζόμενων πρακτικών προσφέροντας καινοτόμες λύσεις για την ασφαλή διαχείριση των ιατρικών δεδομένων, την εξασφάλιση της αυθεντικότητας των δεδομένων και την ενίσχυση της εμπιστοσύνης των εμπλεκόμενων μερών. Το ερευνητικό άρθρο των Anton Hasselgren κ.α. με τίτλο «*Blockchain in healthcare and health sciences—A scoping review*» το 2020 [3] παρουσιάζει μια συνολική προσέγγιση με βελτιώσεις που θα μπορούσε να θέσει η αξιοποίηση του blockchain στις ήδη υπάρχουσες δομές των συστημάτων υγείας. Καταλήγουν πως υπάρχουν πολυάριθμοι τομείς της ιατρικής που θα μπορούσαν να επωφεληθούν από την ένταξη του blockchain στον χώρο της υγείας, επιτρέποντας την ασφαλή ψηφιοποίηση και διαχείριση του φακέλου ασθενών και συμβάλλοντας στην δημιουργία αυτοματοποιημένων υπηρεσιών υγείας, όπως διαγνωστικών υπηρεσιών για ασθενείς, διοικητικών συστημάτων κ.ο.κ.

1.2 Αντικείμενο Διπλωματικής

Η παρούσα διπλωματική εστιάζει στην αξιοποίηση της τεχνολογίας του blockchain με στόχο την αντιμετώπιση αδυναμιών ιατρικών πρακτικών που ισχύουν μέχρι σήμερα στην χώρα μας και οι οποίες αφορούν στην αποστολή και παραλαβή αποτελεσμάτων διαγνωστικών εξετάσεων ασθενών. Αποτελεί εναλλακτική λύση στα υφιστάμενα μοντέλα διαχείρισης των σχετικών δεδομένων που επί του παρόντος περιορίζονται στην ανταλλαγή τους μέσω email και χρήση κωδικών. Πιο συγκεκριμένα, μέσω της χρήσης του πλαισίου αλυσίδας-κορμού Hyperledger Fabric δίνεται η δυνατότητα δημιουργίας ενός ενοποιημένου δικτύου μεταξύ των Δομών Υγείας μέσα στο οποίο ο ασθενής θα μπορεί να ανταλλάσσει τα ευαίσθητα προσωπικά του δεδομένα με ασφάλεια και ιδιωτικότητα. Η προσέγγιση που προτείνεται εστιάζει στην αξιοποίηση εργαλείων του Hyperledger Fabric και πιο συγκεκριμένα των ιδιωτικών συλλογών δεδομένων (Private Data Collections) σε συνδυασμό με τα κατάλληλα έξυπνα συμβόλαια τα οποία μπορούν να χειριστούν αυτές τις συλλογές. Με αυτό τον τρόπο μπορούμε να δημιουργήσουμε ένα επιπρόσθετο επίπεδο ιδιωτικότητας εντός του δικτύου μας χωρίς όμως να περιορίζουμε τις δυνατότητες επικοινωνίας μεταξύ των επιμέρους οντοτήτων.

1.3 Οργάνωση του τόμου

Η εργασία ακολουθεί την παρακάτω δομή:

- Το πρώτο κεφάλαιο αποτελεί μια εισαγωγή η οποία στοχεύει στην συνοπτική παρουσίαση του αντικειμένου στο οποίο εστιάζει η παρούσα διπλωματική.
- Στο κεφάλαιο 2 παρουσιάζεται η τεχνολογία του blockchain και τα βασικά δομικά χαρακτηριστικά της, τα οποία είναι σημαντικά για την κατανόηση του αντικειμένου.
- Στο κεφάλαιο 3 αναλύεται ενδελεχώς η πλατφόρμα του Hyperledger Fabric παρέχοντας το απαραίτητο θεωρητικό υπόβαθρο που χρειάζεται για την κατανόηση του τρόπου λειτουργίας του.
- Το κεφάλαιο 4 αποτελεί μια εισαγωγή στο πρακτικό μέρος της διπλωματικής και παρουσιάζει την προτεινόμενη λύση όπως και τα εργαλεία τα οποία αξιοποιήθηκαν για την υλοποίηση του δικτύου. Επιπλέον, παρέχει τις κατάλληλες οπτικοποιήσεις όπως και παρουσιάζει το παράδειγμα χρήσης το οποίο υλοποιήθηκε για τους σκοπούς της συγκεκριμένης εργασίας.
- Το κεφάλαιο 5 παρουσιάζει την υλοποίηση των σεναρίων όπως και την τεχνική υλοποίηση του δικτύου που αναπτύχθηκε, αναλύοντας τα επιμέρους σημεία ενδιαφέροντος.
- Το κεφάλαιο 6 είναι ο επίλογος στον οποίο γίνεται μια σύνοψη της ερευνητικής διαδικασίας προτείνοντας μελλοντικές επεκτάσεις.

Μέρος Ι

Θεωρητικό Μέρος

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Σε αυτό το κεφάλαιο θα γίνει μια αναλυτική παρουσίαση της τεχνολογίας του blockchain και των δομικών χαρακτηριστικών της. Θα αναλυθούν τα Ethereum και Ethereum Based δίκτυα, δίνοντας έμφαση στο κατάλληλο θεωρητικό υπόβαθρο που απαιτείται για την κατανόησή τους. Ειδικότερα, θα δοθεί μια εισαγωγή στα projects του Hyperledger με κυριότερο το Hyperledger Fabric, το οποίο αξιοποιήθηκε για την υλοποίηση του δικτύου της παρούσας εργασίας.

2.1 Εισαγωγή στην τεχνολογία Blockchain

2.1.1 Τι είναι το blockchain

Το blockchain είναι ένα αναδυόμενο μοντέλο πληροφορικής το οποίο προσφέρει μια αποκεντρωμένη, με υψηλή αποδοτικότητα και διαφάνεια, προσέγγιση σε πολυάριθμες εφαρμογές. Ουσιαστικά, αποτελεί μια αλυσίδα από μπλοκ που αποθηκεύουν όλες τις διεκπεραιωμένες συναλλαγές χρησιμοποιώντας έναν δημόσιο λογιστικό κατάλογο (ledger). Η αλυσίδα μεγαλώνει συνεχώς όταν νέα μπλοκ προσαρτώνται σε αυτή [4]. Η έννοιά του προήλθε από ένα κατανεμημένο σύστημα κρυπτονομίσματος που ονομάζεται Bitcoin και το οποίο παρουσίασε ο Nakamoto το 2008 [5]. Πρακτικά αποτελεί μια κοινή, συνεχώς αυξανόμενη και κατανεμημένη βάση δεδομένων [6]. Κάθε μπλοκ ταυτοποιείται από ένα κρυπτογραφημένο hash και κάθε επόμενο μπλοκ αναφέρεται στο hash του μπλοκ που προηγήθηκε δημιουργώντας έτσι μια αλυσίδα από μπλοκ ή blockchain [7]. Για την ασφάλεια των χρηστών και την συνέπεια του λογιστικού καταλόγου εφαρμόζονται τεχνικές ασύμμετρης κρυπτογραφίας όπως και αλγόριθμοι κατανεμημένης συναίνεσης. Επομένως, η τεχνολογία blockchain διέπεται από 4 βασικά χαρακτηριστικά [4] [8]:

- **Αποκέντρωση**: Στα συμβατικά κεντρικοποιημένα συστήματα συναλλαγών, κάθε συναλλαγή πρέπει να επικυρωθεί μέσω μίας κεντρικής αξιόπιστης αρχής (π.χ. της κεντρικής τράπεζας). Αυτό έχει ως αναπόφευκτο αποτέλεσμα το κόστος και τα προβλήματα απόδοσης στους κεντρικούς διακομιστές. Εν αντιθέσει, το blockchain δεν χρειάζεται αυτόν τον τρίτο μεσάζοντα για την αξιοπιστία των συναλλαγών καθώς υπεύθυνοι για αυτήν είναι οι αλγόριθμοι συναίνεσης που αξιοποιούνται για την διατήρηση της συνέπειας των δεδομένων στο κατανεμημένο δίκτυο και θα μελετηθούν ενδελεχώς σε επόμενη ενότητα.

- **Μονιμότητα**: Το blockchain παρέχει την υποδομή μέσω της οποίας μπορεί να μετρηθεί η ακεραιότητα και επιτρέπει και στα δύο μέρη μιας συναλλαγής να αποδείξουν ότι τα δεδομένα τους είναι αυθεντικά και δεν έχουν τροποποιηθεί.

Εφόσον κάτι έχει προστεθεί σε κάποιο block και έχει γίνει μέρος του blockchain, θεωρείται εξαιρετικά δύσκολο να τροποποιηθεί καθώς αυτό θα απαιτούσε να αλλάξουν όλοι οι κατακερματισμοί (hash) των επόμενων block όπως και να επιβεβαιωθούν από άλλους χρήστες του δικτύου. Συνεπώς, οποιαδήποτε χειραγώγηση ή πλαστογράφηση δεδομένων ανιχνεύεται από το δίκτυο και για αυτό το λόγο θεωρείται σχεδόν αδύνατο να παραβιαστεί. Επομένως, το blockchain αποτελεί ένα αδιάβλητο κατανεμημένο λογιστικό κατάλογο.

- **Ανωνυμία**: Κάθε χρήστης μπορεί να αλληλοεπιδρά με το blockchain με μία δημιουργημένη διεύθυνση, η οποία δεν αποκαλύπτει την πραγματική του ταυτότητα. Παράλληλα, καθώς πρόκειται για ένα αποκεντρωμένο σύστημα, καμία κεντρική αρχή δεν παρακολουθεί ή καταγράφει τις ιδιωτικές πληροφορίες των χρηστών, επιτρέποντας έτσι στο blockchain να παρέχει ένα εγγυημένο επίπεδο ανωνυμίας μέσω του αξιόπιστου περιβάλλοντος του.

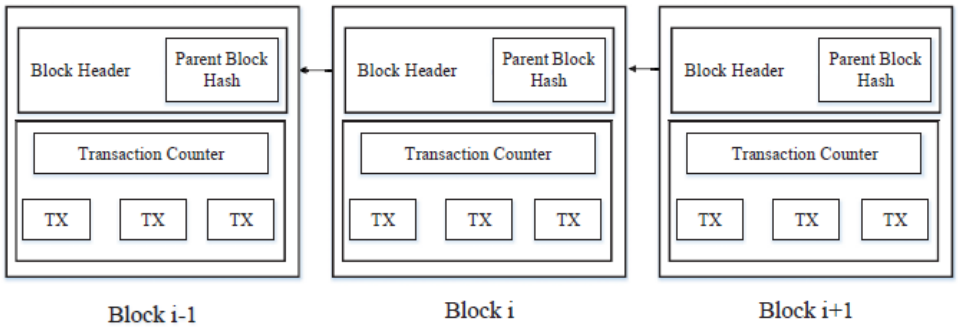
- **Ελεγξιμότητα**: Όλες οι συναλλαγές που συμβαίνουν σε ένα δίκτυο blockchain καταγράφονται από έναν ψηφιακό κατανεμημένο κατάλογο (ledger) και επικυρώνονται με μια ψηφιακή χρονοσφραγίδα. Έτσι καθίσταται δυνατό να ελεγχθούν και να εντοπιστούν προηγούμενες εγγραφές μέσω της προσπέλασης οποιουδήποτε κόμβου του δικτύου. Για παράδειγμα, το Bitcoin αποθηκεύει δεδομένα σχετικά με τα χρηματικά υπόλοιπα των χρηστών βάσει του μοντέλου των Μη Δαπανημένων Συναλλαγών (Unspent Transaction Output – UTXO). Κάθε συναλλαγή πρέπει να αναφέρεται σε κάποιες προηγούμενες μη δαπανημένες συναλλαγές. Μόλις η τρέχουσα συναλλαγή καταγραφεί στο blockchain, η κατάσταση των αναφερόμενων μη δαπανημένων συναλλαγών αλλάζει σε δαπανημένη και με αυτόν τον τρόπο μπορούν να ελεγχθούν και να παρακολουθηθούν εύκολα.

Επομένως, το blockchain αποτελεί μια καινοτόμο προσέγγιση η οποία μπορεί να μειώσει σημαντικά το κόστος και την αποδοτικότητα μέσω της εξάλειψης των μεσαζόντων και της επιτάχυνσης του διακανονισμού των συναλλαγών [4].

Επιπλέον, καθώς επιτρέπει την διεκπεραίωση συναλλαγών χωρίς κάποιον ενδιάμεσο φορέα, η τεχνολογία αυτή έχει πολυάριθμες εφαρμογές σε διάφορες χρηματοοικονομικές υπηρεσίες όπως οι ηλεκτρονικές πληρωμές και τα ψηφιακά περιουσιακά στοιχεία. Παράλληλα, μπορεί να αξιοποιηθεί και σε άλλους τομείς συμπεριλαμβανομένων των δημόσιων υπηρεσιών, του τομέα της ιατρικής, όπως και των υπηρεσιών ασφάλειας.

Αυτοί οι τομείς ευνοούνται από το blockchain καθώς είναι αμετάβλητο. Μια συναλλαγή δεν μπορεί να παραποιηθεί μόλις ενσωματωθεί στο blockchain. Συνεπώς οι επιχειρήσεις που απαιτούν υψηλή αξιοπιστία και ειλικρίνεια μπορούν να αξιοποιήσουν αυτή την τεχνολογία για να προσελκύσουν πελάτες. Παράλληλα, καθώς μιλάμε για ένα κατανεμημένο σύστημα, μπορούμε να αποφύγουμε προβλήματα που αφορούν κεντρικοποιημένα συστήματα [4] [8].

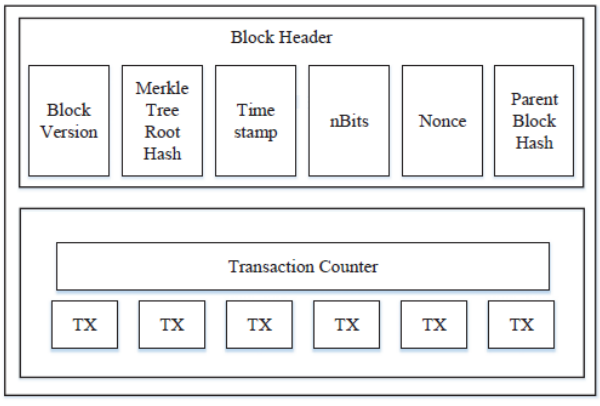
Στην εικόνα 1 παρουσιάζεται μια αποτύπωση ενός τυπικού blockchain.



Εικόνα 1: Παράδειγμα blockchain που αποτελείται από i + 1 blocks [8].

Μέσω της αξιοποίησης μιας συνάρτησης κατακερματισμού (hash function) το κρυπτογραφικό αποτύπωμα (hash) του προηγούμενου μπλοκ περιέχεται στην επικεφαλίδα του επόμενου μπλοκ, με κάθε ένα μπλοκ να έχει μόνο έναν γονέα ή parent block. Το πρώτο block ενός blockchain ονομάζεται γεννήτορας μπλοκ (genesis block) και δεν έχει κάποιον γονέα [8] [9]. Αξίζει να σημειωθεί πως στα Ethereum blockchains για την αντιμετώπιση προβλημάτων που σχετίζονται με την ασφάλεια του δικτύου αξιοποιείται το πρωτόκολλο GHOST ("Greedy Heaviest Observed Subtree"), όπου εντός του blockchain λαμβάνονται υπόψιν και οι κατακερματισμοί των "uncle" blocks τα οποία είναι τα παιδιά των προγόνων του μπλοκ που βρισκόμαστε [9]. Αναλυτικότερα, με σκοπό την διασφάλιση του υψηλού ρυθμού δημιουργίας block όπως και την διατήρηση της υψηλής ροής συναλλαγών, το Ethereum υιοθέτησε τον μηχανισμό των λεγόμενων uncle blocks. Τα uncle blocks αποτελούν στάσιμα (stale) blocks όπου αποθηκεύεται ένα μέρος της αμοιβής. Τα συνήθη blocks ενθαρρύνονται να αναφέρονται σε uncle blocks για να κερδίσουν περισσότερες αμοιβές [10]. Στο πλαίσιο αυτό, το GHOST υιοθετήθηκε ώστε να οριοθετεί τον αριθμό αυτών των στάσιμων block, οργανώνοντάς τα βάσει της σημαντικότητάς τους και απομακρύνοντας αυτά με την χαμηλότερη [11].

Εν συνεχεία θα αναλύσουμε την εσωτερική δομή ενός block:



Εικόνα 2: Δομή ενός block [8].

Το block αποτελείται από δύο βασικά μέρη, την επικεφαλίδα (header) και το σώμα (body). Ειδικότερα, η επικεφαλίδα περιέχει τα εξής:

1. **Block Version**: Υποδεικνύει ποιο σύνολο κανόνων επικύρωσης πρέπει να ακολουθηθεί.
2. **Merkle Tree Root Hash**: Η τιμή κατακερματισμού (hash) όλων των συναλλαγών στο block.
3. **Timestamp**: Η τρέχουσα ώρα σε μορφή δευτερολέπτων βάσει της παγκόσμιας ώρας από 1η Ιανουαρίου 1970.
4. **nBits**: Ο στόχος κατωφλιού ενός έγκυρου κατακερματισμού block.
5. **Nonce**: Ένα πεδίο 4 bytes, το οποίο συνήθως ξεκινάει από το 0 και αυξάνεται για κάθε υπολογισμό κατακερματισμού (hash).
6. **Parent Block Hash**: Μια τιμή κατακερματισμού 256 bits που δείχνει στο προηγούμενο block.

Αντίστοιχα, το σώμα του block αποτελείται από έναν μετρητή συναλλαγών (Transaction Counter) όπως και τις διάφορες συναλλαγές. Ο μέγιστος αριθμός συναλλαγών που μπορεί να περιέχει ένα block εξαρτάται τόσο από το μέγεθος του ίδιου του block όσο και από το μέγεθος της συναλλαγής. Τέλος, το blockchain χρησιμοποιεί έναν ασύμμετρο μηχανισμό κρυπτογράφησης για να επικυρώσει την αυθεντικότητα των συναλλαγών με την χρήση ψηφιακών υπογραφών που θα μελετηθούν σε επόμενες ενότητες [8].

2.1.2 Αρχιτεκτονική ενός δικτύου blockchain

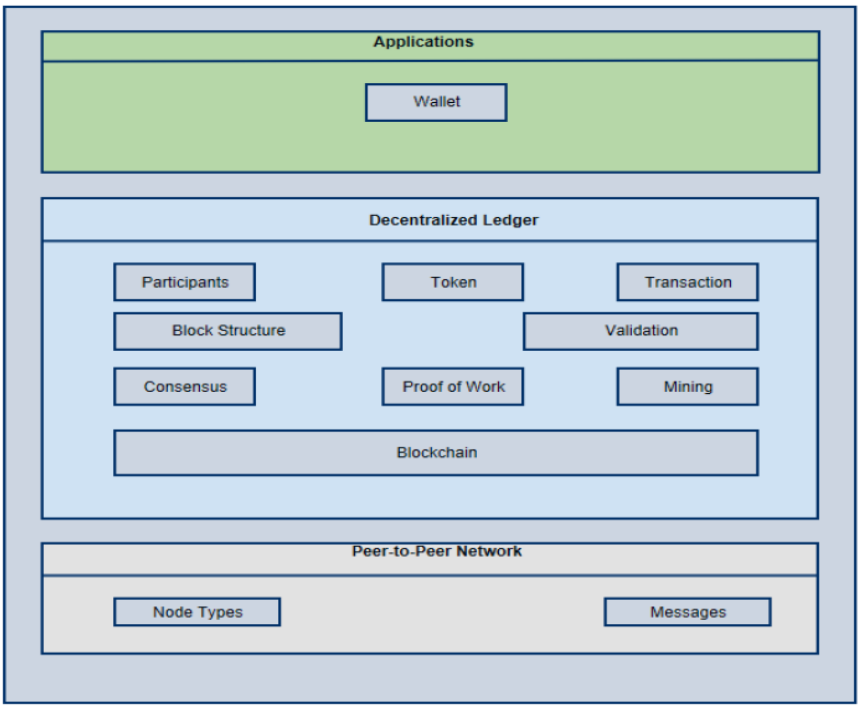
Όπως αναφέραμε και προηγουμένως, η τεχνολογία blockchain λειτουργεί με την λογική μιας αποκεντρωμένης βάσης δεδομένων η οποία υπάρχει σε πολλαπλούς υπολογιστές, με το κάθε υπολογιστή να διαθέτει ένα πανομοιότυπο αντίγραφο αυτής. Συνεπώς αποτελεί ένα δίκτυο ομότιμων κόμβων που λειτουργεί πάνω από το διαδίκτυο [12].

Αν και η 1η γενιά της τεχνολογίας blockchain (Blockchain 1.0) αξιοποιείται αμιγώς με τα κρυπτονομίσματα, η 2η γενιά (Blockchain 2.0) επέτρεψε την δημιουργία αποκεντρωμένων εφαρμογών αξιοποιώντας τεχνολογίες όπως το Ethereum, το οποίο θα μελετηθεί λεπτομερώς σε επόμενο κεφάλαιο. Στην παρούσα ενότητα θα μελετήσουμε διαφορετικές προσεγγίσεις της αρχιτεκτονικής ενός δικτύου blockchain καθώς αυτή εξελίσσεται μαζί με την τεχνολογία [13].

2.1.2.1 Αρχιτεκτονική 3-επιπέδων

Με βάση τον Simanta Shekhar Sarmah στο ερευνητικό άρθρο *«Understanding Blockchain Technology»* το 2018 [12] η αρχιτεκτονική του blockchain μπορεί να χωριστεί σε τρία επίπεδα τα οποία είναι:

1. **Επίπεδο Peer-to-Peer Δικτύου ή Επίπεδο Ομότιμων Κόμβων**
2. **Επίπεδο του αποκεντρωμένου λογιστικού καταλόγου (ledger)**
3. **Επίπεδο Εφαρμογών**



Εικόνα 3: Αρχιτεκτονική blockchain τριών επιπέδων [12].

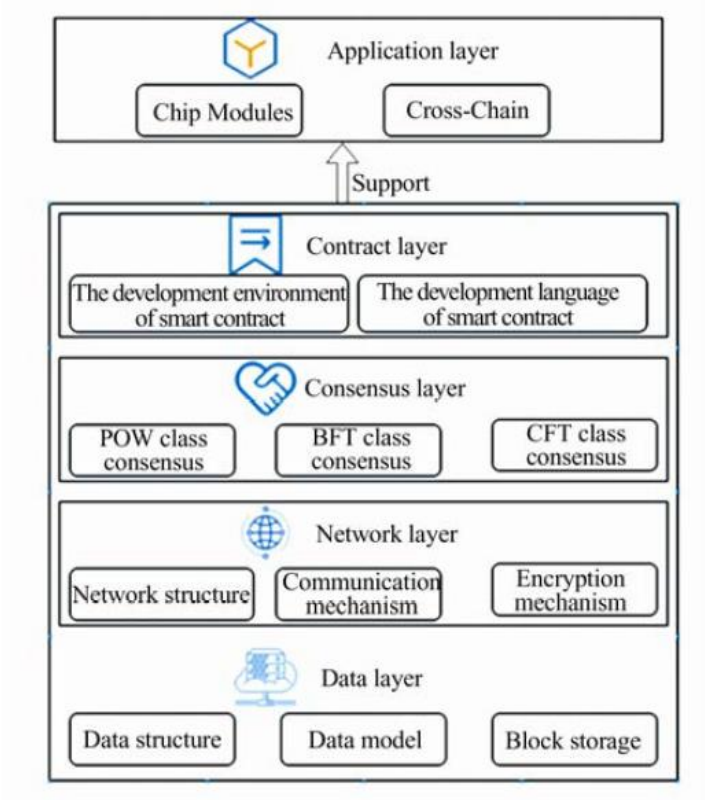
Το χαμηλότερο επίπεδο της αρχιτεκτονικής του blockchain θεωρείται πως είναι το **επίπεδο Ομότιμων Κόμβων (Peer-to-Peer Network).** Σε αυτό το επίπεδο οι διάφοροι τύποι κόμβων παίζουν διαφορετικούς ρόλους, ενώ παράλληλα ανταλλάσσονται μηνύματα με σκοπό την διασφάλιση του αποκεντρωμένου λογιστικού καταλόγου (Decentralized ledger). Πιο συγκεκριμένα, κάθε κόμβος αναλαμβάνει μια διαφορετική λειτουργία ανάλογα με τον ρόλο του εντός του δικτύου. Ένας κόμβος μπορεί να είναι μηχανισμός εξόρυξης (miner) όταν προτείνει και επιβεβαιώνει συναλλαγές συμβάλλοντας έτσι στην διατήρηση της συναίνεσης και της ασφάλειας του blockchain. Αντίστοιχα μπορεί να πραγματοποιεί απλές λειτουργίες, όπως την επιβεβαίωση κάποια πληρωμής αλλά και άλλες λειτουργίες ανάλογα με το είδος της τεχνολογίας blockchain.

Το **επίπεδο του αποκεντρωμένου λογιστικού καταλόγου** (**Decentralized Ledger**) είναι το ενδιάμεσο επίπεδο σε μια αρχιτεκτονική blockchain και αποτελείται από έναν αδιάβλητο και συνεπή παγκόσμιο λογιστικό κατάλογο (ledger). Στο επίπεδο αυτό, οι συναλλαγές μπορούν να ομαδοποιηθούν σε blocks τα οποία όπως είδαμε και προηγουμένως είναι κρυπτογραφικά συνδεδεμένα το ένα με το άλλο. Οι συναλλαγές μπορούν να οριστούν ως η ανταλλαγή token μεταξύ δύο συμμετεχόντων, με κάθε συναλλαγή να περνάει μια διαδικασία επικύρωσης πριν αυτή θεωρηθεί ως νόμιμη. Μια επόμενη βασική έννοια αυτού του επιπέδου είναι το mining ή αλλιώς εξόρυξη, που είναι η διαδικασία ομαδοποίησης συναλλαγών σε ένα block το οποίο προστίθεται στο τέλος του τρέχοντος blockchain. Το blockchain με την σειρά του χρησιμοποιεί έναν αλγόριθμο απόδειξης εργασίας (Proof of Work) για να αποφασίσει ποια αλυσίδα κατέβαλε τον περισσότερο κόπο για να δημιουργηθεί, ώστε να διασφαλιστεί η συναίνεση μεταξύ των κόμβων του δικτύου. Αναλυτικότερα, θα δούμε τους διαφορετικούς αλγόριθμους συναίνεσης σε επόμενη ενότητα.

Τέλος, το **επίπεδο εφαρμογής** εμπεριέχει το λογισμικό εφαρμογών του blockchain. Πρακτικά, παρέχει μια εύκολα κατανοητή διεπαφή όπου οι χρήστες μπορούν να παρακολουθούν και να ελέγχουν τις συναλλαγές τους, ενώ παράλληλα συμβάλλει στην διασφάλιση των κρυπτονομισμάτων. Παραδείγματος χάριν, το ψηφιακό πορτοφόλι του Bitcoin δημιουργεί και αποθηκεύει δημόσια και ιδιωτικά κλειδιά επιτρέποντας στους χρήστες να ελέγχουν τα αξόδευτα bitcoins που έχουν.

2.1.2.2 Αρχιτεκτονική 5-επιπέδων

Μια επόμενη προσέγγιση είναι αυτή των C. Wang, H. Jiang, J. Zeng, M. Yu, Q. Huang και Z. Zuo στο άρθρο τους με τίτλο *«A review of blockchain layered architecture and technology application research»* το 2021 [14]. Με την ένταξη του Ethereum και των smart contracts η αρχιτεκτονική ενός δικτύου blockchain άρχισε να εξελίσσεται. Η παρούσα έρευνα προτείνει ένα μοντέλο 5 επιπέδων όπως παρουσιάζεται στην εικόνα 4.



Εικόνα 4: Αρχιτεκτονική 5 επιπέδων ενός blockchain Network [14].

Συνεπώς, το δίκτυο blockchain χωρίζεται στα :

1. **Επίπεδο δεδομένων (Data layer)**
2. **Επίπεδο Δικτύου (Network Layer)**
3. **Επίπεδο Συναίνεσης (Consensus Layer)**
4. **Επίπεδο Συμβολαίων (Contract Layer)**
5. **Επίπεδο Εφαρμογής (Application Layer)**

Το **επίπεδο δεδομένων** αποτελεί το θεμέλιο του δικτύου blockchain και διαιρείται σε τρεις υποκατηγορίες, (α) την δομή δεδομένων, (β) το μοντέλο δεδομένων και (γ) την αποθήκευση των block. Κάθε μία από αυτές τις υποκατηγορίες παίζει έναν διαφορετικό ρόλο εντός αυτού του επιπέδου.

- **Η δομή δεδομένων** αποτελεί μια συλλογή από στοιχεία που έχουν ένα βαθμό λογικής διασύνδεσης μεταξύ τους.

- Το **μοντέλο δεδομένων** είναι η αφαιρετική διαδικασία, η οποία έχει σκοπό την μοντελοποίηση των δεδομένων ανάλογα με τις ανάγκες του κάθε δικτύου.

- Η **αποθήκευση block** δεν αναφέρεται στην αποθήκευση δεδομένων στο blockchain. Αφορά την δημιουργία μιας υπηρεσίας αποθήκευσης υψηλής διαθεσιμότητας και χαμηλού κόστους, μέσα από την διάθεση περισσότερων κόμβων που συνδέονται και αποθηκεύουν δεδομένα σε ατομικούς κόμβους. Ειδικότερα το Bitcoin και το Hyperledger Fabric αποθηκεύουν ένα μπλοκ ως αρχείο με το όνομα αυτού να αντικατοπτρίζει το ύψος του block.

Το **επίπεδο δικτύου** αφορά την δομή του δικτύου, τους μηχανισμούς επικοινωνίας και τις μεθόδους κρυπτογράφησης. Όπως έχουμε αναφέρει και προηγουμένως, η διαφορά μεταξύ ενός συμβατικού μοντέλου δικτύου και του blockchain, είναι η διαφορετική δομή που υιοθετείται από το blockchain, λόγω της αξιοποίησης ενός δικτύου ομότιμων κόμβων. Με αυτό τον τρόπο, αποφεύγεται η αδράνεια του κεντρικού κόμβου όπως και η κατάρρευση ολόκληρου του συστήματος λόγω επίθεσης. Ακόμα και να αποσύρουμε μια μερίδα κόμβων μπορεί ακόμα να εγγυηθεί η κανονική λειτουργία ολόκληρου του συστήματος.

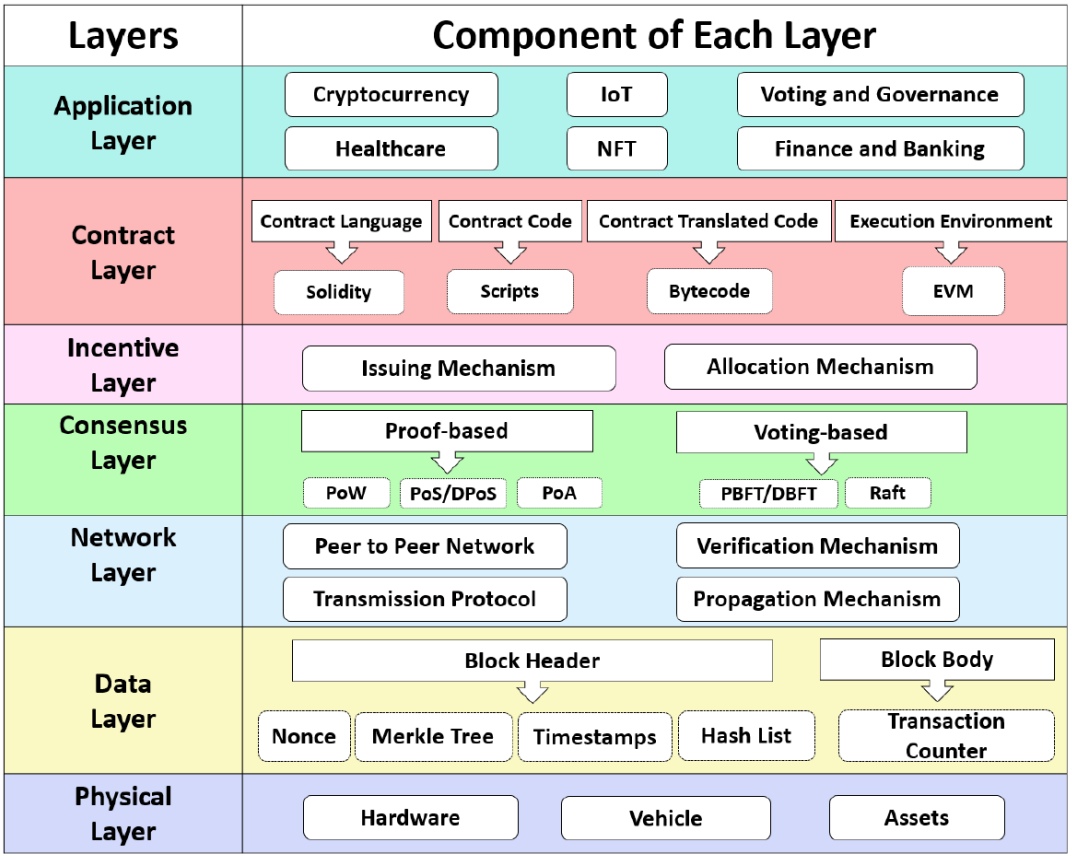
Το **επίπεδο συναίνεσης** εγγυάται την συνέπεια των πληροφοριών που καταγράφονται από κάθε κόμβο στο δίκτυο blockchain. Αποτελεί μία από τις κύριες τεχνολογίες του blockchain και για αυτό τον λόγο αξιοποιούνται αποδοτικοί αλγόριθμοι συναίνεσης ώστε να υποστηρίξουν την υψηλή παραλληλία της αρχιτεκτονικής του δικτύου. Αυτοί οι αλγόριθμοι συναίνεσης χωρίζονται σε δύο μεγάλες κατηγορίες, τους αλγόριθμους κινήτρου που ανήκουν κυρίως στην κλάση POX και τους μη βασιζόμενους σε κίνητρα αλγόριθμους που ανήκουν κυρίως στις κλάσεις CFT και BFT. Περισσότερα θα δούμε στην ενότητα 2.1.3 που αφορά αμιγώς τους αλγόριθμους συναίνεσης. Τέλος, αξίζει να σημειωθεί πως με την ανάπτυξη των εφαρμογών blockchain οι απαιτήσεις απόδοσης γίνονται όλο και υψηλότερες, γεγονός που απαιτεί την αναζήτηση βελτιστοποιήσεων στις διαφορετικές κατηγορίες αλγορίθμων.

Το **επίπεδο συμβολαίων** αποτελεί το θεμέλιο των προγραμματιζόμενων χαρακτηριστικών του blockchain και χωρίζεται σε δύο μεγάλες κατηγορίες. Η $1^{η}$ αφορά τις διαφορετικές γλώσσες που μπορούν να αξιοποιηθούν για την υλοποίηση των έξυπνων συμβολαίων ενώ η $2^{η}$ εστιάζει στο περιβάλλον ανάπτυξης των έξυπνων συμβολαίων. Η σωστή επιλογή αυτών των δύο κατηγοριών αποτελεί την ακρογωνιαία λίθο ανάπτυξης ενός σύγχρονου δικτύου blockchain και θα αναλυθεί περαιτέρω σε επόμενη ενότητα που αφορά τα έξυπνα συμβόλαια.

Τέλος το **επίπεδο εφαρμογής,** βάσει των C. Wang, H. Jiang, J. Zeng, M. Yu, Q. Huang και Z. Zuo, αποτελεί τον συνδετικό κρίκο μεταξύ του blockchain και της πραγματικότητας. Με την είσοδο των έξυπνων συμβολαίων στην τεχνολογία του blockchain, δόθηκε η δυνατότητα δημιουργίας πολυάριθμων εφαρμογών που ξεφεύγουν από τις κλασσικές υλοποιήσεις εφαρμογών χρηματοοικονομικού ενδιαφέροντος. Το παρόν άρθρο εστίασε στις ερευνητικές προσπάθειες που γίνονται σε δύο μεγάλες τεχνολογίες. Αυτές αφορούν την διασύνδεση αλυσίδων όπως και τον συνδυασμό μονάδων τσιπ (chip modules) με το blockchain, ώστε να επιτευχθεί η αυθεντικότητα των λαμβανόμενων δεδομένων.

2.1.2.3 Αρχιτεκτονική 7-επιπέδων

Μία τελευταία και νεότερη προσέγγιση είναι αυτή των Sepideh Mollajafari και Kamal Bechkoum στο άρθρο τους με τίτλο *«Blockchain Technology and Related Security Risks: Towards a Seven-Layer Perspective and Taxonomy»* το 2023 [15]. Με στόχο να μελετηθούν δυνατοί κίνδυνοι ασφάλειας όπως και αδυναμίες του συστήματος ενός blockchain δικτύου προτάθηκε ο διαχωρισμός της αρχιτεκτονικής σε 7 επίπεδα όπως φαίνονται στην παρακάτω εικόνα:



Εικόνα 5:Διαχωρισμός 7 επιπέδων [15].

Σε αντιδιαστολή με την προηγούμενη προσέγγιση βλέπουμε πως έχουν προστεθεί τα:

1. **Επίπεδο Κινήτρου (Incentive Layer)**

2. **Φυσικό Επίπεδο (Physical Layer)**

Για να διασφαλιστεί η αποκέντρωση και η ασφάλεια ενός συστήματος blockchain, απαιτείται ένας μεγάλος αριθμός ειλικρινών κόμβων οι οποίοι συμβάλλουν στην επαλήθευση και επικύρωση κάθε συναλλαγής (ποσοστό μεγαλύτερο από 50%). Οι μηχανισμοί κινήτρων είναι απαραίτητοι για την ενθάρρυνση των κόμβων να συμβάλλουν ενεργά στην ασφάλεια και συντήρηση του συστήματος. Το **επίπεδο κινήτρου** επομένως, παίζει έναν ζωτικό ρόλο στην διασφάλιση της ειλικρίνειας του δικτύου και αποτελεί αυτοτελές κομμάτι της αρχιτεκτονικής ενός δικτύου blockchain.

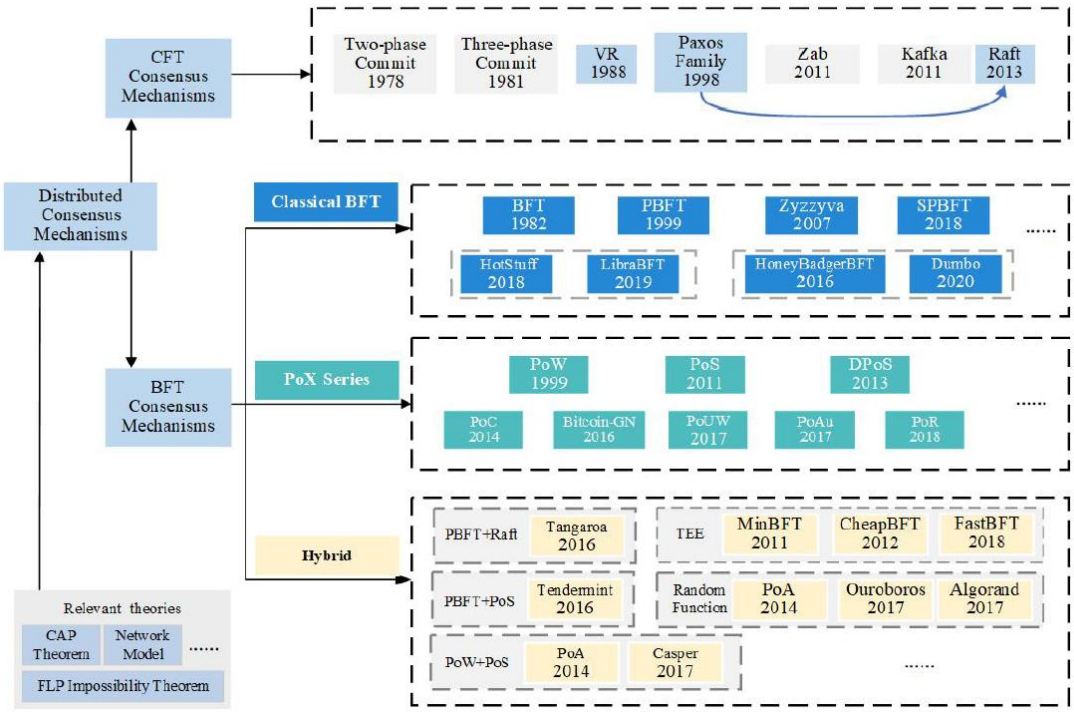
Το **φυσικό επίπεδο**, αποτελεί τον φυσικό δίαυλο που μεταφέρει τα δεδομένα (bits). Το κύριο χαρακτηριστικό αυτό του επιπέδου είναι οι συσκευές IoT (Internet of Things), οι οποίες συνδέονται στο διαδίκτυο και λειτουργούν ως κόμβοι στο blockchain.

Εν κατακλείδι, η εξέλιξη της αρχιτεκτονικής από την πρώτη προσέγγιση των τριών επιπέδων έως και την τελευταία προσέγγιση των επτά επιπέδων αντικατοπτρίζει την συνεχόμενη ανάπτυξη και βελτίωση της τεχνολογίας blockchain. Η διαφοροποίηση πλέον σε επτά επίπεδα, επιτρέπει τη λεπτομερή διαχείριση και ανάλυση πολλαπλών συνιστωσών και λειτουργιών του blockchain, προσφέροντας ένα ευρύτερο πλαίσιο για την αντιμετώπιση των προκλήσεων που σχετίζονται με την ασφάλεια, την αποκέντρωση και την κλιμάκωση ανοίγοντας έτσι νέους ορίζοντες για την τεχνολογία blockchain και τις πρακτικές εφαρμογές της στον πραγματικό κόσμο.

2.1.3 Αλγόριθμοι Συναίνεσης

Όπως έχουμε αναφέρει και προηγουμένως το blockchain αποτελεί ένα κατανεμημένο δίκτυο ομότιμων κόμβων (Peer to Peer Network). Οι μηχανισμοί συναίνεσης έχουν ενταχθεί στην τεχνολογία blockchain ως ένας ανθεκτικός σε σφάλματα μηχανισμός για την επαλήθευση συναλλαγών [16]. Κάθε συναλλαγή χρειάζεται επιβεβαίωση από πολλαπλούς χρήστες και οι αλγόριθμοι συναίνεσης χρησιμοποιούνται για να διατηρηθεί η συμφωνία μεταξύ των κόμβων στο δίκτυο. Οι πιο διαδεδομένοι αλγόριθμοι συναίνεσης είναι οι Proof of Work (PoW) και Proof of Stake (PoS). Παρ' όλα αυτά, υπάρχουν και άλλοι αλγόριθμοι συναίνεσης που αξιοποιούν εναλλακτικές υλοποιήσεις του PoW και PoS, υβριδικές υλοποιήσεις όπως και εντελώς καινούργιες στρατηγικές συναίνεσης [17].

Οι πιο σημαντικοί μηχανισμοί συναίνεσης παρουσιάζονται στην εικόνα 6.



Εικόνα 6: Οι πιο σημαντικοί αλγόριθμοι συναίνεσης του blockchain [18].

Η βασική κατηγοριοποίηση των διαφορετικών μηχανισμών συναίνεσης και τα χαρακτηριστικά τους παρουσιάζεται παρακάτω [18].

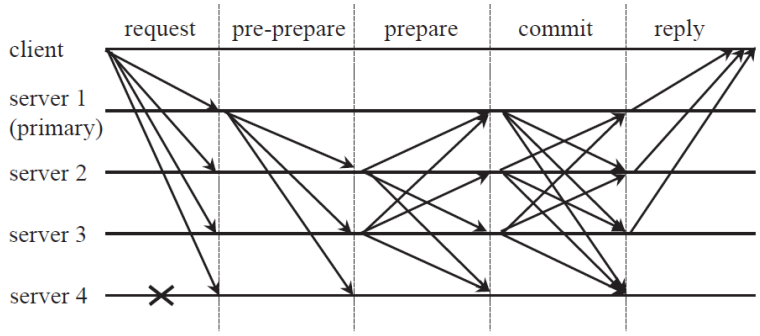
Τύπος Μηχανισμού Συναίνεσης	**Λόγος Κατηγοριοποίησης**	**Χαρακτηριστικά**
CFT Μηχανισμοί Συναίνεσης	Εφαρμόζονται μόνο σε μη - Βυζαντινά σενάρια.	Υψηλή απόδοση αλλά χαμηλή αντοχή σε σφάλματα.
Κλασσικοί BFT Μηχανισμοί Συναίνεσης	Κατάλληλοι για Βυζαντινά σενάρια και χρησιμοποιούνται κυρίως σε Permissioned blockchain.	Ντετερμινιστική συναίνεση.
PoX series Μηχανισμοί Συναίνεσης	Κατάλληλοι για Βυζαντινά σενάρια και χρησιμοποιούνται κυρίως σε Permissionless blockchains.	Πιθανοτική συναίνεση.
Υβριδικοί Μηχανισμοί Συναίνεσης	Συνδυασμός δύο ή περισσότερων μηχανισμών συναίνεσης.	Μπορούν να αξιοποιήσουν τα πλεονεκτήματα κάθε μηχανισμού μειώνοντας παράλληλα τις αδυναμίες τους, οδηγώντας έτσι σε πιο ανθεκτικά συστήματα.

Θα ξεκινήσουμε την ανάλυση μας από το προβλήματα που οδήγησαν στην αξιοποίηση αυτών των αλγορίθμων συναίνεσης. Το πρόβλημα του βυζαντινού στρατηγού (The Byzantine Generals Problem) αναλύθηκε για πρώτη φορά από τους L. Lamport, R. Shostak και M. Pease στο ερευνητικό τους άρθρο με τίτλο «*The Byzantine Generals Problem*» το 1982 [19] και έθετε προβληματισμούς αναφορικά με την αποτυχία επικοινωνίας. Πιο συγκεκριμένα, πως μπορεί κάθε κόμβος (‘στρατηγός’) σε ένα σύστημα να είναι σίγουρος ότι η πληροφορία που λαμβάνει είναι αληθής; Αυτό οδήγησε στην δημιουργία των Byzantine Fault Tolerance αλγόριθμων ή BFT. Οι αλγόριθμοι αυτοί στοχεύουν στην επίλυση του προβλήματος της επίτευξης συναίνεσης όταν οι κόμβοι μπορούν να παράγουν τυχαία δεδομένα [17]. Ο PBFT (Practical Byzantine Fault Tolerance Algorithm) είναι ένας κλασσικός αλγόριθμος αντιγραφής καταστάσεων BFT και ανήκει στην κατηγορία των αλγορίθμων βασιζόμενων σε ψηφοφορία (voting-based algorithms). Πρέπει να εγγυηθεί την ορθότητα των ακόλουθων δύο χαρακτηριστικών :

- **Ασφάλεια**: Όλα τα μη ελλατωματικά αντίγραφα εκτελούν τα αιτήματα με την ίδια σειρά.
- **Διατήρηση της ζωής (Liveness)**: Οι πελάτες τελικά λαμβάνουν τις απαντήσεις στα αιτήματα τους.

Για να εξασφαλίσει την διατήρηση της ζωής το PBFT χρησιμοποιεί την τεχνική της υπόθεσης ασθενούς συγχρονισμού, που εγγυάται ότι τα μηνύματα θα παραδοθούν μετά από ένα συγκεκριμένο χρονικό όριο. Όπως φαίνεται στην εικόνα 7, στις κανονικές περιπτώσεις ο πρωτεύων κόμβος ζητάει όλα τα αιτήματα από τους πελάτες και στην συνέχεια όλοι οι σωστοί κόμβοι φτάνουν σε μία συμφωνία μέσω της εκτέλεσης του πρωτοκόλλου τριών επιπέδων που περιλαμβάνει τα στάδια:

1. Προ-προετοιμασία
2. Προετοιμασία
3. Δέσμευση



Εικόνα 7:Κανονική Λειτουργία PBFT [20].

Ο πρωτεύων κόμβος μπορεί να καταρρεύσει ή να εκτελέσει διφορούμενη συμπεριφορά, δηλαδή, να στέλνει ασυνεπή μηνύματα σε διαφορετικά υποσύνολα διακομιστών, οδηγώντας στην αποτυχία της συναίνεσης στον τρέχοντα γύρο. Έτσι ενεργοποιείται η διαδικασία αλλαγής προοπτικής (view-change process) για να επιλεχθεί νέος πρωτεύων [20].

Εν συνεχεία, θα αναλύσουμε την κατηγορία των PoX αλγορίθμων συναίνεσης που αποτελούν και το θεμέλιο της τεχνολογίας blockchain όπως την ξέρουμε σήμερα.

Ο **Proof-of-Work (PoW)** μηχανισμός, προτάθηκε ως λύση για την επίτευξη της συναίνεσης εντός ενός permissionless blockchain από την πρώτη δημοσίευση αναφορικά με το blockchain το 2008 [5]. Ιστορικά, η έννοια του PoW ξεκίνησε από μια δημοσίευση του 1992 που αποσκοπούσε στην εύρεση μεθόδων αντιμετώπισης της ανεπιθύμητης αλληλογραφίας [21] [18]. Η βασική ιδέα ήταν να απαιτηθεί από τον αποστολέα του email να ολοκληρώσει κάποιες υπολογιστικές εργασίες πριν στείλει τα emails. Η κύρια εφαρμογή του blockchain όπως και η πιο τυπική εφαρμογή του PoW είναι το bitcoin. Για να καταγραφούν τα δεδομένα συναλλαγών blockchain και να επιτευχθεί η συναίνεση εντός ενός συγκεκριμένου χρονικού πλαισίου ο PoW προτείνει μια ιδέα: όλοι οι κόμβοι στο δίκτυο blockchain εκτελούν ανταγωνιστική λογιστική. Αυτό σημαίνει ότι αν οι κόμβοι θέλουν να παράγουν και να γράψουν ένα νέο μπλοκ στο blockchain, πρέπει να επιλύσουν έναν περίπλοκο γρίφο ο οποίος μπορεί εύκολα να επαληθευτεί αλλά δύσκολα να βρεθεί. Όποιος κόμβος του δικτύου λύσει πρώτος το πρόβλημα αποκτά τα δικαιώματα λογιστικής (accounting rights) [18]. Πρακτικά, ο σκοπός είναι η εύρεση του nonce, που αναφέρθηκε στην ενότητα 2.1.1. Λόγω όμως του τρόπου που λειτουργούν οι συναρτήσεις κατακερματισμού είναι αρκετά σύνθετο να βρεθεί καθώς αυτό επιτυγχάνεται μόνο μέσω των επαναλαμβανομένων δοκιμών σε πιθανές τιμές του [22] . Μετά από αυτό, ο κόμβος που αποκτά τα δικαιώματα λογιστικής μεταδίδει την λύση που βρήκε όπως και το αρχείο συναλλαγής σε άλλους κόμβους για επαλήθευση και ξεκινάει ο επόμενος γύρος εξόρυξης. Αν οι άλλοι συμμετέχοντες κόμβοι επαληθεύσουν την συναλλαγή του μπλοκ και η απάντηση στον γρίφο είναι σωστή τότε είναι και αξιόπιστη. Το νέο μπλοκ γράφεται στο blockchain του επαληθευτή και αυτός με την σειρά του εισέρχεται στον επόμενο γύρο ανταγωνιστικής εξόρυξης [18].

Ο **Proof-of-Stake (PoS)** μηχανισμός προτάθηκε με στόχο να λύσει το πρόβλημα της τεράστιας κατανάλωσης ισχύος που απαιτεί ο PoW. Στον PoS, η ηλικία του κρυπτονομίσματος είναι σημαντική και ορίζεται ως το ποσό του νομίσματος επί την περίοδο κατοχής (coin age). Για παράδειγμα, αν έχουμε 10 νομίσματα και τα κρατήσουμε για 10 ημέρες, αυτό σημαίνει ότι η ηλικία του νομίσματος μας είναι 100. Το πρώτο κρυπτονόμισμα που υιοθέτησε το PoS είναι το Peercoin, που προτάθηκε το 2012 από τους S. King και S. Nadal [23]. Εισήγαγαν ένα πεδίο χρονικής σφραγίδας σε κάθε συναλλαγή για να διευκολύνουν τον υπολογισμό της ηλικίας του κάθε νομίσματος. Αν τα νομίσματα αυτά ξοδευτούν, τότε η ηλικία του νομίσματος μηδενίζεται. Δηλαδή, η ηλικία του νομίσματος καθορίζει την δυσκολία της εξόρυξης [18]. Πιο συγκεκριμένα, κάθε κόμβος στο Peercoin λύνει ένα πρόβλημα PoW με την δυσκολία του η οποία μπορεί να μειωθεί καταναλώνοντας αυτό το coin age. Στις νεότερες υλοποιήσεις του PoS η εύρεση λύσεων σε προβλήματα έχει αφαιρεθεί πλήρως και οι αρχηγοί των μπλοκ δεν επιλέγονται βάσει της υπολογιστικής τους ισχύς. Αντ' αυτού, επιλέγονται βάσει του πονταρίσματος (stake) που κάνουν. Συνεπώς, μειώνεται αισθητά η κατανάλωση ενέργειας καθώς η διαδικασία επιλογής δεν βασίζεται στην υπολογιστή ισχύ. Παράλληλα, μόνο ένα μπλοκ δημιουργείται σε κάθε γύρο με αποτέλεσμα η δημιουργία νέων μπλοκ και η διαδικασία επιβεβαίωσης να είναι αρκετά ταχύτερες από αυτές του PoW, όπου κρατούνται σε σχετικά χαμηλά

επίπεδα ώστε να διατηρήσουν την ασφάλεια του δικτύου [22]. Τέλος, αξίζει να σημειωθεί πως η συγκεκριμένη προσέγγιση έχει κάποια μειονεκτήματα με τα κυριότερα να είναι:

1. Πολυπλοκότητα του αλγόριθμου που δυσκολεύει τις υλοποιήσεις.
2. Οι εξορύκτες (miners) προτιμούν να κρατήσουν τα νομίσματα τους αντί να τα πουλήσουν γεγονός που τους αφήνει ευάλωτους.
3. Καθώς η εξόρυξη είναι χαμηλού κόστους και εύκολη σε επιθέσεις, τίθενται προβλήματα ασφάλειας.

Για αυτούς τους λόγους προτείνονται βελτιωμένες μέθοδοι υλοποίησης, όπως αυτή στο πρωτόκολλο Ouroboros όπου προστέθηκε η έννοια των περιόδων. Παράλληλα, εμφανίζονται και βελτιωμένες προσεγγίσεις όπως η DPoS που αποτελεί μια εξελιγμένη έκδοση του κλασσικού PoS μηχανισμού [24].

2.1.4 Κρυπτογραφία και ασφάλεια δικτύου στην τεχνολογία Blockchain

Η κρυπτογραφία αποτελεί έναν από τους βασικούς πυλώνες ενός δικτύου blockchain και ορίζεται ως η διαδικασία διαφύλαξης των δεδομένων από εισβολείς μέσω της κωδικοποίησης τους. Συνεπώς μόνο συγκεκριμένοι χρήστες μπορούν να αποκτήσουν πρόσβαση και να κατανοήσουν την πληροφορία που έχει κρυπτογραφηθεί. Στην παρούσα ενότητα θα αναλύσουμε τις βασικές κρυπτογραφικές τεχνικές που αξιοποιούνται εντός ενός δικτύου blockchain.

Η κρυπτογραφία χωρίζεται σε δύο βασικές κατηγορίες:

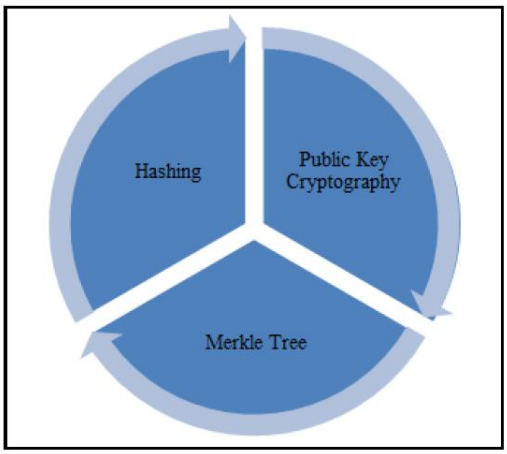
- **Συμμετρική Κρυπτογραφία**: Αποτελεί την πιο δημοφιλή εκδοχή και αξιοποιεί τους κλασσικούς αλγόριθμους κωδικοποίησης. Χρησιμοποιεί ένα μοναδικό ιδιωτικό κλειδί τόσο για την κωδικοποίηση όσο και για την αποκωδικοποίηση του μηνύματος και το οποίο πρέπει να έχει συμφωνηθεί και από τις δύο πλευρές.

- **Ασύμμετρη Κρυπτογραφία**: Δημιουργήθηκε για την επίλυση των προβλημάτων που έθετε η συμμετρική κρυπτογραφία και αποκαλείται και ως κρυπτογραφία δημόσιου κλειδιού (public key cryptography). Αξιοποιεί δύο κλειδιά, ένα ιδιωτικό και ένα δημόσιο με το δημόσιο κλειδί να χρησιμοποιείται για την κωδικοποίηση της πληροφορίας και το ιδιωτικό για την αποκωδικοποίηση αυτής.

Και οι δύο τεχνικές έχουν τα πλεονεκτήματα και τα μειονεκτήματα τους με την συμμετρική κρυπτογραφία να προσφέρει μεγαλύτερη ταχύτητα λόγω της αξιοποίησης ενός μοναδικού κλειδιού, ενώ η ασύμμετρη κρυπτογραφία προσφέρει πολύ μεγαλύτερη ασφάλεια. Νεότερες τεχνικές αξιοποιούν ένα συνδυασμό των δύο παραπάνω τεχνικών δημιουργώντας τους λεγόμενους ψηφιακούς φακέλους (digital folders) για την ασφαλή μετάδοση της πληροφορίας [25].

Τα δίκτυα blockchain αξιοποιούν τρεις βασικές κρυπτογραφικές τεχνικές:

1. **Κατακερματισμός (Hashing)**
2. **Κρυπτογραφία Δημόσιου Κλειδιού**
3. **Δέντρο Merkle**



Εικόνα 8: Η κρυπτογραφία εντός του blockchain [25].

Ο **κατακερματισμός (Hashing)** είναι μια διαδικασία που αντιστοιχίζει μια ακολουθία μηνυμάτων οποιουδήποτε μήκους σε μια συντομότερη με σταθερή τιμή μήκους. Όπως έχουμε δει και σε προηγούμενη ενότητα, αξιοποιείται για την διασφάλιση της ακεραιότητας των δεδομένων καθώς όταν τα δεδομένα αλλάζουν και η τιμή του κατακερματισμού τους αλλάζει αντίστοιχα. Συνεπώς, ακόμα και αν τα δεδομένα βρίσκονται σε ένα μη ασφαλές περιβάλλον η ακεραιότητα τους μπορεί να επιβεβαιωθεί βάσει της τιμής κατακερματισμού τους [26]. Ο αλγόριθμος SHA256 ανήκει στην τάξη των αλγορίθμων SHA-2 παράγοντας ένα μήνυμα 256 bit και αποτελείται από δύο στάδια:

1. **Προ-επεξεργασία μηνύματος (Message preprocessing),** όπου ανεξαρτήτως μήκους πληροφορίας εκτελείται μια διαδικασία προσθήκης δυαδικών bits με σκοπό την διάτμηση του μηνύματος σε πολλαπλά μπλοκ των 512 bit.

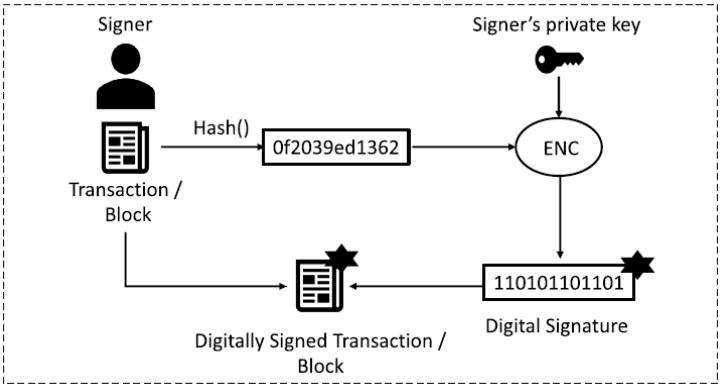
2. **Φάση κύριου βρόγχου (Main loop phase),** όπου κάθε μπλοκ μηνυμάτων επεξεργάζεται από μια συνάρτηση συμπίεσης. Σε αυτό το στάδιο η είσοδος της τρέχουσας συνάρτησης συμπίεσης είναι η έξοδος της προηγούμενης συνάρτησης συμπίεσης. Η έξοδος της τελευταίας συνάρτησης συμπίεσης είναι η τιμή κατακερματισμού του αρχικού μηνύματος.

Με αυτό τον τρόπο όπως είδαμε και στην ενότητα 2.1.1 δημιουργείται μια σειρά από μπλοκ που το καθένα περιέχει στην επικεφαλίδα του την τιμή του προηγούμενου μπλοκ. Στο σημείο αυτό, αξίζει να σημειωθεί και η έννοια του δείκτη κατακερματισμού (hash pointer). Ο δείκτης αυτός αποτελεί μια δομή δεδομένων που περιέχει επιπρόσθετα κάποιες πληροφορίες, όπως και κατακερματισμούς κωδικών που σχετίζονται με αυτές. Ένας τυπικός δείκτης αξιοποιείται για να ανακτήσει την πληροφορία ενώ ο δείκτης κατακερματισμού χρησιμοποιείται για να επαληθεύσει ότι αυτές οι πληροφορίες δεν έχουν παραποιηθεί [26].

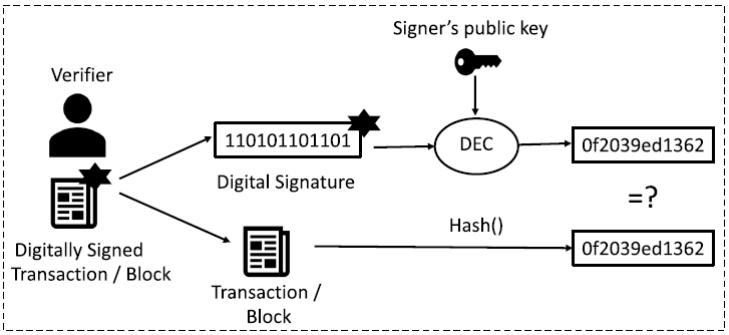
Η **κρυπτογραφία δημόσιου κλειδιού** αξιοποιείται με την μορφή των ψηφιακών υπογραφών (digital signatures) και αποσκοπεί στην δημιουργία σύντομων κωδικών οι οποίοι ορίζουν τις υπογραφές ψηφιακών μηνυμάτων. Αυτές οι υπογραφές δημιουργούνται μέσω της χρήσης ενός ιδιωτικού κλειδιού και μπορούν να επαληθευτούν με την χρήση του αντίστοιχου δημόσιου κλειδιού, συμβάλλοντας στην προστασία ενάντια στην παραποίηση και πλαστογράφηση των ψηφιακών μηνυμάτων. Αυτού του είδους το σχήμα χρησιμοποιείται εντός του blockchain δικτύου για την υπογραφή μιας συναλλαγής επιτυγχάνοντας έτσι:

1. Την επιβεβαίωση του προοριζόμενου αποστολέα.
2. Την ακεραιότητα της συναλλαγής.
3. Την μη-αποποίηση του αποστολέα.

Η κρυπτογραφία δημόσιου κλειδιού επομένως, αποτελεί την ακρογωνιαία λίθο ενός δικτύου blockchain κάνοντας το επαληθεύσιμο και με εφικτή συναίνεση. Στις εικόνες 9 και 10 μπορούμε να δούμε ένα τυπικό παράδειγμα αξιοποίησης μιας ψηφιακής υπογραφής εντός του blockchain [27].



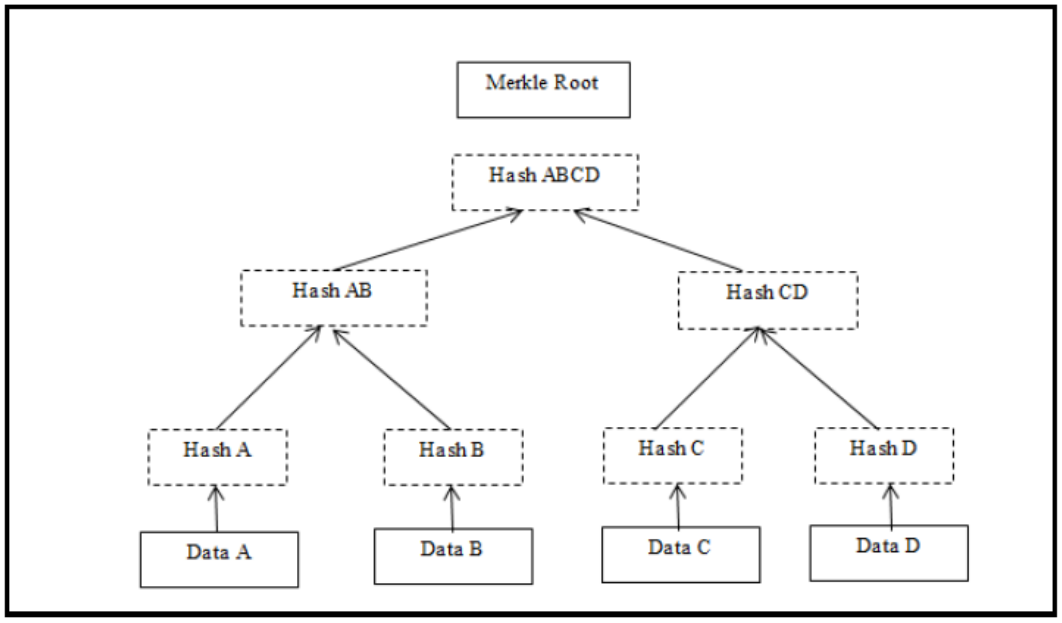
Εικόνα 9: Διαδικασία υπογραφής συναλλαγής στο blockchain [27].



Εικόνα 10: Διαδικασία επιβεβαίωσης ψηφιακά υπογεγραμμένης συναλλαγής στο block [27].

Στην πρώτη εικόνα βλέπουμε πως ένας χρήστης εντός του blockchain δημιουργεί μια ψηφιακά υπογεγραμμένη συναλλαγή ή block αξιοποιώντας το ιδιωτικό του κλειδί. Στην δεύτερη εικόνα βλέπουμε πως οι υπόλοιποι κόμβοι του δικτύου ελέγχουν αν η συναλλαγή (ή το μπλοκ) είναι έγκυρη αξιοποιώντας το δημόσιο κλειδί του χρήστη που υπέγραψε.

Τέλος, θα αναλύσουμε τα **δέντρα Merkle (Merkle Trees)** που αποτελούν μια μέθοδο κατακερματισμού που αφορά την διάτμηση ενός μεγάλου όγκου δεδομένων σε μικρά μπλοκ. Κάθε μπλοκ κατακερματίζεται και στην συνέχεια ενώνεται με άλλα μπλοκ και ξανά κατακερματίζεται ώστε να δημιουργηθεί ένας συνδυαστικός κατακερματισμός μεταξύ των δύο μπλοκ. Αυτή η διαδικασία συνεχίζεται μέχρις ότου καταλήξουμε σε έναν μοναδικό κατακερματισμό εντός της ομάδας των μπλοκ.



Εικόνα 11: Παράδειγμα Δέντρου Merkle [25].

Παράλληλα, τα δέντρα Merkle χρησιμοποιούν έναν μηχανισμό κατακερματισμού όπου διατηρείται μόνο η έξοδος του κατακερματισμού και όχι η ίδια η πληροφορία. Η ρίζα Merkle (Merkle Root) αποτελεί τον τελικό κόμβο του δέντρου και διατηρεί τον συνδυασμό όλων των κατακερματισμών [25].

2.1.5 Permissioned vs Permissionless Δίκτυα Blockchain

Σημαντική ενότητα της εισαγωγής στην τεχνολογία του blockchain αποτελεί ο σχολιασμός των διαφορετικών προσεγγίσεων που χαρακτηρίζουν ένα δίκτυο blockchain. Ανάλογα με τους συμμετέχοντες αυτού του δικτύου, μπορεί να διαχωριστεί σε:

1. **Δημόσιο ή permissionless δίκτυο**
2. **Ιδιωτικό ή permissioned δίκτυο**

Το **δημόσιο ή permissionless δίκτυο** επιτυγχάνει την συναίνεση μέσω ενός αποκεντρωμένου πρωτοκόλλου, το οποίο εφαρμόζεται σε ένα θεωρητικά απεριόριστο πλήθος από κόμβους ή συμμετέχοντες. Τα permissionless πρωτόκολλα δεν απαιτούν από τους κόμβους να αποκαλύπτουν τις ταυτότητες τους πέραν κάποιου ψευδώνυμου. Επιπλέον, οι συμμετέχοντες μπορούν να αποκτήσουν νέα αναγνωριστικά ή και να διαθέτουν πολλαπλά ψευδώνυμα την ίδια στιγμή [28]. Συνεπώς, τα δημόσια δίκτυα

blockchain δεν μπορούν να υποθέσουν κανένα επίπεδο εμπιστοσύνης στο περιβάλλον τους και πρακτικά είναι προσβάσιμα από οποιονδήποτε ενδιαφέρεται να συμμετάσχει, με τις κυριότερες εφαρμογές τους να αφορούν την εξόρυξη και ανταλλαγή κρυπτονομισμάτων [29]. Οι κλασσικοί αλγόριθμοι συναίνεσης που αναλύσαμε στην ενότητα 2.1.3 παίζουν σημαντικό ρόλο σε αυτά τα δίκτυα, με κυριότερο τον αλγόριθμο Proof-of-Work που όπως έχουμε ήδη αναφέρει επιτρέπει την ύπαρξη συναίνεσης παρά την ανωνυμία που διακατέχει το δίκτυο.

Εν αντιθέσει, τα **ιδιωτικά ή permissioned δίκτυα** δεν επιτρέπουν την ανοικτή συμμετοχή στις διαδικασίες υποβολής και επικύρωσης συναλλαγών. Απαιτούν οι κόμβοι επικύρωσης, οι οποίοι μπορούν να ενημερώσουν το blockchain, να εγκριθούν πριν αναλάβουν αυτόν τον ρόλο από έναν 'διαχειριστή'. Παράλληλα, κάποια ιδιωτικά δίκτυα θέτουν και απαιτήσεις έγκρισης σε κόμβους χρηστών με την διαδικασία έγκρισης να απαιτεί την πλήρη διαφάνεια σχετικά με την ταυτότητα του χρήστη. Συνεπώς οι μηχανισμοί συναίνεσης στα permissioned δίκτυα blockchain υποθέτουν κάποιο επίπεδο εμπιστοσύνης λόγω του υποκείμενου θεσμικού πλαισίου [28] . Τέτοια δίκτυα είναι χρήσιμα για υλοποιήσεις σε οργανισμούς και εταιρίες όπου μόνο επιλεγμένα μέλη μπορούν να συμμετάσχουν και έχουν πολυάριθμες εφαρμογές σε διαδικασίες ψηφοφορίας, διαχείριση της εφοδιαστικής αλυσίδας όπως και για την εύρεση και διαχείρισης ψηφιακών ταυτοτήτων και περιουσιακών στοιχείων.

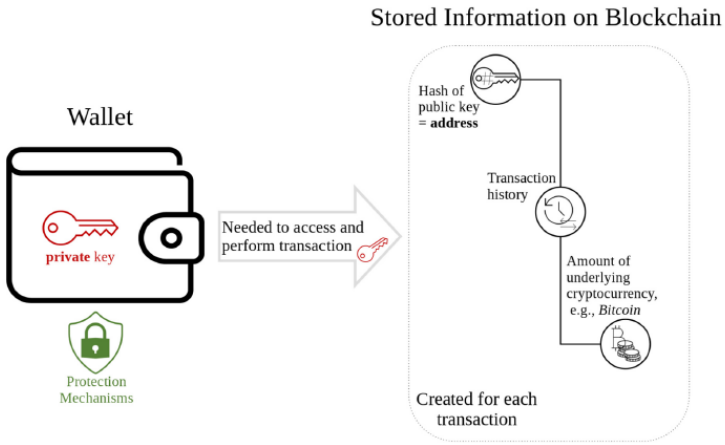
Στην συνέχεια, παρουσιάζονται τα κύρια χαρακτηριστικά και λειτουργίες των δύο κατηγοριών [29].

Δημόσια ή Permissionless Δίκτυα	*Ιδιωτικά ή Permissioned Δίκτυα*
Οποιοσδήποτε μπορεί να συμμετάσχει.	Οι συμμετέχοντες προ-επιλέγονται.
Απαιτούν την ύπαρξη κάποιου κρυπτονομίσματος.	Δεν απαιτείται η ύπαρξη κάποιας μορφής κρυπτονομίσματος.
Διαθέτουν υψηλό βαθμό αποκέντρωσης.	Έχει χαμηλό βαθμό αποκέντρωσης.
Έχουν χαμηλό ρυθμό διέλευσης.	Διαθέτει υψηλό ρυθμό διέλευσης.
Υψηλή κατανάλωση ενέργειας.	Χαμηλή κατανάλωση ενέργειας.

Τέλος, αξίζει να σημειωθεί πως υπάρχουν και τα **υβριδικά blockchains** που αποτελούν μια ενοποίηση των δύο ανωτέρω κατηγοριών και αξιοποιούνται εκεί που απαιτείται καλύτερος έλεγχος. Μια τέτοια προσέγγιση φαίνεται και στο ερευνητικό άρθρο των Yu Tang et al. με τίτλο «*Hedera: A Permissionless and Scalable Hybrid Blockchain Consensus Algorithm in Multiaccess Edge Computing for IoT*» το 2023 [30]. Πιο συγκεκριμένα, προτάθηκε ένας permissionless και επεκτάσιμος αλγόριθμος συναίνεσης που ονομάζεται 'Hedera' και ο οποίος συνδυάζει τον permissionless Proof-of-Capacity αλγόριθμο με τον permissioned ασύγχρονο Βυζαντινό αλγόριθμο.

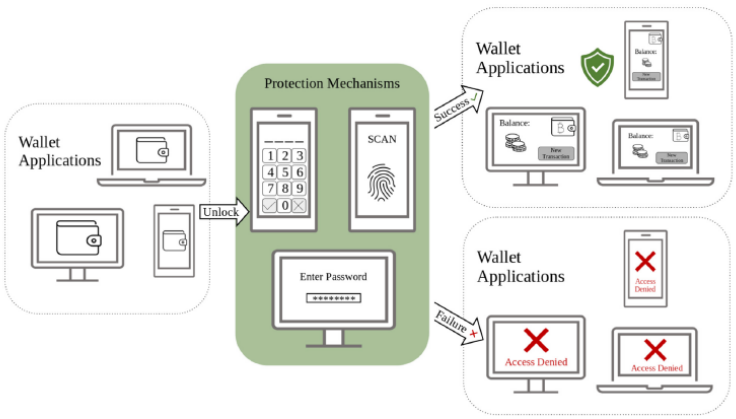
2.1.6 Cryptocurrency Wallets

Τα ψηφιακά πορτοφόλια κρυπτονομισμάτων αποτελούν αναπόσπαστο κομμάτι του blockchain. Κάθε χρήστης που επιθυμεί να αξιοποιήσει κάποια πλατφόρμα blockchain για οποιαδήποτε συναλλαγή πρέπει να χρησιμοποιήσει ένα τέτοιο crypto-wallet. Πρακτικά, αποτελούν την διεπαφή μεταξύ του χρήστη και του blockchain και είναι το μέρος όπου αποθηκεύεται και προστατεύεται το ιδιωτικό του κλειδί. Τα κρυπτονομίσματα όπως και άλλες πληροφορίες σχετικές με τον κατακερματισμό του δημόσιου κλειδιού του χρήστη, το οποίο συνήθως αντιστοιχεί με την διεύθυνση, το ιστορικό συναλλαγών και το πλήθος κρυπτονομισμάτων, αποθηκεύονται στο blockchain [31] [32].



Εικόνα 12: Ρόλος των wallets για μια συναλλαγή κρυπτονομισμάτων [32].

Για να μπορέσει ο χρήστης να έχει πρόσβαση στο ψηφιακό του πορτοφόλι απαιτείται κάποια μορφή πιστοποίησης. Αυτό επιτυγχάνεται μέσω της αξιοποίησης μηχανισμών κλειδώματος όπως η χρήση κάποιου PIN, κωδικού εισόδου ή βιομετρικών αυθεντικοποιήσεων. Με αυτό τον τρόπο, προστατεύεται το ιδιωτικό κλειδί όπως και άλλες ευαίσθητες πληροφορίες που βρίσκονται εντός του πορτοφολιού. Αναλυτικότερα, αξιοποιείται αυτός ο κωδικός ώστε οι αποθηκευμένες πληροφορίες να είναι κρυπτογραφημένες και να μην μπορούν να διαβαστούν από τρίτους και μόνο με την αξιοποίηση του να μπορεί να γίνει η αποκρυπτογράφηση του περιεχομένου [32].



Εικόνα 13: Μηχανισμοί προστασίας του Wallet [32].

Η κατηγοριοποίηση των ψηφιακών πορτοφολιών ξεκινάει από τον τρόπο λειτουργίας τους και πιο συγκεκριμένα κατά πόσο απαιτούν σύνδεση στο διαδίκτυο ή όχι. Έτσι έχουμε:

- **Hot Wallets**: Ψηφιακά πορτοφόλια που απαιτούν σύνδεση στο διαδίκτυο και συνήθως αξιοποιούνται για διαδικτυακές αγορές.
- **Cold Wallets**: Ψηφιακά πορτοφόλια που λειτουργούν offline. Πρακτικά, αυτού του είδους τα πορτοφόλια λειτουργούν σαν μια μορφή τράπεζας για αποθήκευση των ψηφιακών περιουσιακών στοιχείων.

Εν συνεχεία, αυτές οι δύο κύριες κατηγορίες υποδιαιρούνται σε επιμέρους ταξινομήσεις βάσει της λειτουργικότητας τους, με κάθε τύπο πορτοφολιού κρυπτονομισμάτων να διαθέτει μοναδικά επίπεδα ασφάλειας και ιδιωτικότητας. Αυτό γίνεται ώστε να διασφαλιστεί η προστασία του ιδιωτικού κλειδιού του χρήστη [33].

- **Desktop Wallets**: Αποτελεί λογισμικό το οποίο μπορεί να εγκατασταθεί σε έναν μοναδικό προσωπικό ή φορητό υπολογιστή. Είναι εύκολο στην χρήση του και προσφέρει ένα από τα μέγιστα επίπεδα ασφάλειας. Ωστόσο, αν ο υπολογιστής προσβληθεί από κάποιον ιό υπάρχει πιθανότητα απώλειας των κλειδιών και κατ' επέκταση των κεφαλαίων [31].
- **Online Wallets**: Αποτελεί λογισμικό το οποίο λειτουργεί στο cloud και θεωρητικά μπορεί να χρησιμοποιηθεί από οποιαδήποτε τοποθεσία επιλέγει ο χρήστης. Παρόλο που είναι βολικό στην πρόσβαση, τέτοια πορτοφόλια αποθηκεύουν τα ιδιωτικά και δημόσια κλειδιά online όπως και ελέγχονται από ένα τρίτο πρόσωπο. Συνεπώς είναι υποκείμενα σε κινδύνους διαδικτυακών επιθέσεων ή από κενά ασφαλείας αυτού του τρίτου προσώπου [31].
- **Mobile Wallets**: Αποτελεί λογισμικό το οποίο λειτουργεί ως εφαρμογή κινητού, κάνοντας το πιο αποδοτικό και εύκολο στην χρήση συγκριτικά με άλλου είδους πορτοφόλια. Μπορεί να προσπελαστεί από οπουδήποτε, αρκεί να υπάρχει σύνδεση στο διαδίκτυο. Ένα βασικό ζήτημα είναι τα θέματα ασφάλειας που προκύπτουν από αυτού του είδους τα πορτοφόλια. Πιο συγκεκριμένα, αν και τα νεότερα mobile wallets αξιοποιούν τους προηγμένους μηχανισμούς ασφαλείας των κινητών όπως το ARM TrustZone, συνεχίζουν να είναι ευάλωτα σε ιούς, malware και key loggers [33].
- **Multisignature Wallets**: Σε αυτού το είδους τα ψηφιακά πορτοφόλια απαιτούνται δύο ή τρία ιδιωτικά κλειδιά για να δοθεί η πρόσβαση σε χρηματικό ποσό μετά από κάποια συναλλαγή. Αυτή η μέθοδος είναι ωφέλιμη για επιχειρήσεις καθώς τους επιτρέπει να αναθέτουν ευθύνες σε πολλαπλούς υπαλλήλους, οι οποίοι πρέπει να παρέχουν το δικό τους ιδιωτικό κλειδί για να προσπελάσουν τα ψηφιακά περιουσιακά στοιχεία [33].

Μια τελευταία προσέγγιση είναι και αυτή των **Hardware Wallets** ή διαφορετικά ονομαζόμενα **Paper Wallets** όπου διαχωρίζονται από τις παραπάνω κατηγορίες που αποτελούν λογισμικό εφαρμογής. Σε αυτού του είδους τα πορτοφόλια, τα κλειδιά αποθηκεύονται σε φυσικά μέσα όπως κάποιο USB ή ακόμα και σε εκτυπωμένη έκδοση και αποτελούν έτσι μια από τις πιο ασφαλείς μεθόδους [31] [33].

2.2 Ethereum Based Blockchain Δίκτυα

Η τεχνολογία blockchain αποτελεί μια σχετικά νέα προσέγγιση στον τομέα των πληροφοριακών τεχνολογιών. Ως μια από τις πρώτες εφαρμογές, το Bitcoin αποτελεί βασικό πυλώνα τόσο της εξέλιξης της τεχνολογίας όσο και του ερευνητικού ενδιαφέροντος. Μαζί με το Ethereum, το οποίο έχει ως επίκεντρο τα έξυπνα συμβόλαια, αντιπροσωπεύουν τον πυρήνα της σύγχρονης ανάπτυξης του blockchain. Η παρούσα ενότητα παρέχει μια σύντομη παρουσίαση του δικτύου Ethereum, όπως και μια εποπτική ανάλυση των Ethereum Based Δικτύων και πιο συγκεκριμένα του πλαισίου αλυσίδας-κορμού Hyperledger Fabric που αποτελεί βασικό αντικείμενο της παρούσας διπλωματικής.

2.2.1 Εισαγωγή στο Ethereum

Το Ethereum είναι ένα δίκτυο από ομότιμους κόμβους (Peer-to-Peer) εικονικών μηχανών, το οποίο οποιοσδήποτε προγραμματιστής μπορεί να χρησιμοποιήσει ώστε να τρέξει κατανεμημένες εφαρμογές, αλλιώς αποκαλούμενες ως Dapps (Decentralized Apps). Αποτελεί μια εξελικτική προσέγγιση του blockchain, η οποία ονομάζεται blockchain 2.0, και επιτρέπει στους χρήστες την ανάπτυξη έξυπνων συμβολαίων. Αυτές οι κατανεμημένες εφαρμογές μπορεί να είναι οτιδήποτε, αλλά το δίκτυο είναι βελτιστοποιημένο για να εκτελεί κανόνες που ενεργοποιούνται αυτόματα όταν ικανοποιούνται κάποιες συγκεκριμένες συνθήκες, όπως ένα συμβόλαιο. Το Ethereum χρησιμοποιεί το δικό του αποκεντρωμένο δημόσιο blockchain για να αποθηκεύει, εκτελεί και να προστατεύει κρυπτογραφικά αυτά τα συμβόλαια. Κάθε υπολογιστής στο δίκτυο κατεβάζει μια εικονική μηχανή για να συγχρονιστεί με το blockchain και παραμένει διαθέσιμος για την εκτέλεση συμβολαίων. Σαν τεχνολογία υιοθετεί ένα μοντέλο λογαριασμών που προσομοιάζει αυτό ενός τραπεζικού συστήματος. Κάθε λογαριασμός έχει μια διεύθυνση των 20 byte και περιέχει τα εξής 4 πεδία:

1. Το nonce που είναι ένας μετρητής που χρησιμοποιείται για να διασφαλίζει ότι κάθε συναλλαγή μπορεί να επεξεργαστεί μόνο μια φορά.
2. Το τρέχον υπόλοιπο ether(ETH).
3. Τον κωδικό συμβολαίου του λογαριασμού αν υπάρχει.
4. Τον αποθηκευτικό χώρο του λογαριασμού ο οποίος είναι άδειος σαν προεπιλογή.

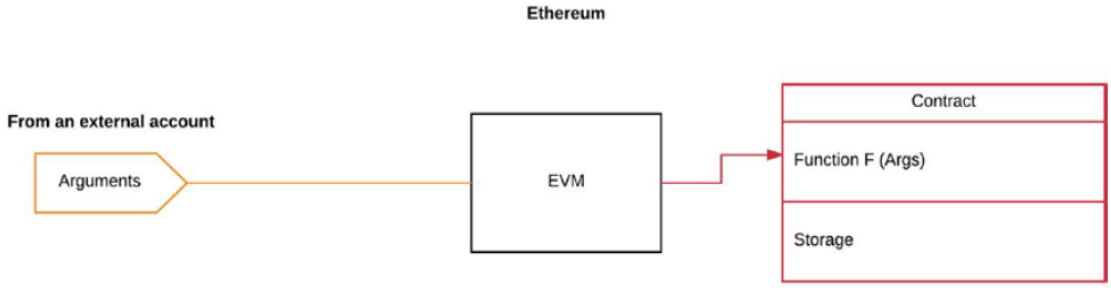
Αναλυτικότερα, το “Ether” αποτελεί το κύριο εσωτερικό κρυπτονόμισμα του Ethereum και χρησιμοποιείται για την πληρωμή των τελών συναλλαγής [34].

Παράλληλα, οι λογαριασμοί χωρίζονται σε δύο βασικές κατηγορίες.

1. **Εξωτερικά ιδιόκτητος λογαριασμός (EOA account)** ο οποίος ελέγχεται από ένα ζευγάρι δημόσιου/ιδιωτικού κλειδιού και χρησιμοποιείται για την διαχείριση των κρυπτονομισμάτων όπως και την αλληλεπίδραση με τα έξυπνα συμβόλαια μέσω συναλλαγών.

2. **Λογαριασμός συμβολαίου (Contract Account)**: Αποτελεί τον λογαριασμό ο οποίος ελέγχεται από τον κώδικα του έξυπνου συμβολαίου (smart contract). Αξιοποιεί κωδικούς χωρίς κλειδιά και χρησιμοποιείται κυρίως για την υλοποίηση διάφορων απαιτήσεων λειτουργίας όπως και την καταγραφή αλλαγών στην κατάσταση του συμβολαίου.

Σε αντίθεση με τα EOAs, οι λογαριασμοί συμβολαίων δεν μπορούν να πραγματοποιήσουν συναλλαγές αλλά μπορούν να στείλουν κλήσεις μηνυμάτων για να καλέσουν άλλα συμβόλαια. Παράλληλα, αν και δεν μπορούν να αλληλοεπιδράσουν προληπτικά με τα EOAs, διαθέτουν μηχανισμούς όπως η αυτοκαταστροφή κατά την οποία όλα τα κρατούμενα ethers επιστρέφονται στους δημιουργούς τους μετά από επιτυχή εκτέλεση του συμβολαίου [35]. Σημαντικό επίσης αποτελεί το γεγονός ότι ένας λογαριασμός συμβολαίου διατηρεί ένα ether balance πεδίο όπως και μια μοναδική διεύθυνση, η οποία καθορίζεται από την διεύθυνση του δημιουργού αυτού του συμβολαίου, σε συνδυασμό με την συναλλαγή η οποία εκτελέστηκε για την δημιουργία του [36].



Εικόνα 14: Συνοπτική παρουσίαση του υπολογιστικού πλαισίου σε ένα Ethereum Δίκτυο [37].

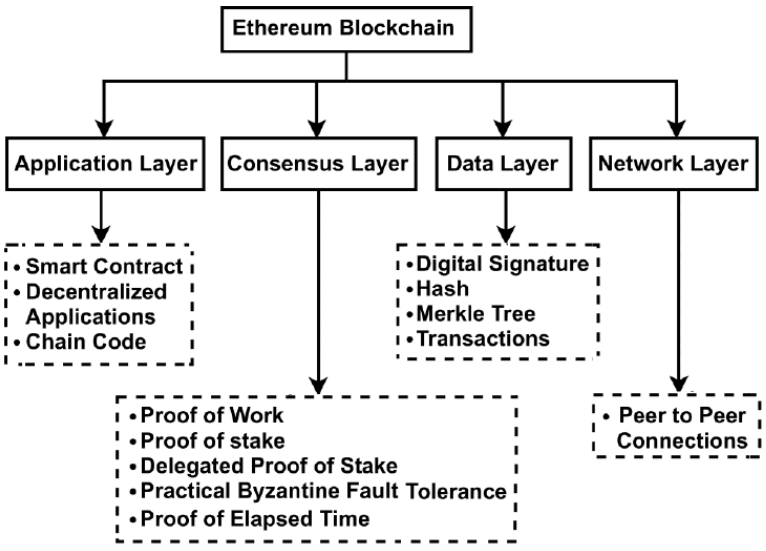
Το Ethereum τοποθετεί ένα EVM (Ethereum Virtual Machine) σε κάθε κόμβο, ώστε να μπορεί να εκτελεστεί επαληθεύσιμος κώδικας στο blockchain. Στην Εικόνα 14 βλέπουμε έναν εξωτερικό λογαριασμό να περνάει κάποια ορίσματα σε μια συνάρτηση. Το EVM θα κατευθύνει αυτή την κλήση στο κατάλληλο συμβόλαιο όπου και θα την εκτελέσει, εφόσον παρέχεται η κατάλληλη ποσότητα ether. Επομένως βλέπουμε πως κάθε συναλλαγή στο Ethereum μπορεί να θεωρηθεί και ως μια κλήση συνάρτησης. Αναλυτικότερα, τα συμβόλαια γράφονται σε μια υψηλότερου επιπέδου γλώσσα όπως το Solidity και στην συνέχεια μεταγλωττίζονται σε bytecode που αξιοποιείται στο EVM. Αυτό το bytecode στην συνέχεια μεταφορτώνεται στο blockchain μέσω της διεπαφής του Ethereum. Το EVM είναι σχεδιασμένο να είναι εντελώς απομονωμένο από το περιβάλλον και το υπόλοιπο δίκτυο. Ο κώδικας που τρέχει εντός του EVM δεν έχει καμία πρόσβαση στο δίκτυο και τις διεργασίες του. Τα συμβόλαια έχουν πρόσβαση στον εξωτερικό κόσμο όπως και στα υπόλοιπα συμβόλαια μόνο όταν μεταγλωττιστούν σε bytecode [37].

Τέλος, αξίζει να σημειωθεί και η έννοια του state. Το Ethereum υιοθετεί μια στρατηγική σχεδίασης που χρησιμοποιεί ένα αντικείμενο κατάστασης (state object).

Η κατάσταση αποθηκεύει μια λίστα λογαριασμών, όπου κάθε λογαριασμός έχει ένα υπόλοιπο, καθώς και συγκεκριμένες πληροφορίες για το blockchain που αφορούν τον κώδικα και την αποθήκευση δεδομένων. Μια συναλλαγή θεωρείται έγκυρη αν ο λογαριασμός του αποστολέα έχει αρκετό υπόλοιπο για να την πληρώσει. Εν συνεχεία, εκτελείται ο κώδικας του παραλήπτη της πληρωμής, αν αυτός υπάρχει. Η εκτέλεση ενός συμβολαίου ή κώδικα που συσχετίζεται με κάποιον λογαριασμό μπορεί να έχει διαφορετικές επιδράσεις στην κατάσταση [37].

2.2.2 Ανάλυση Δομικών Στοιχείων του Network – Smart Contracts

Όπως έχει ήδη αναφερθεί το Ethereum αποτελεί μια Turing-complete πλατφόρμα blockchain όπου οι χρήστες μπορούν να γράψουν δικά τους έξυπνα συμβόλαια με διαφορετικούς κανόνες. Στο κεφάλαιο 2.1.3 αναλύσαμε την εξέλιξη των αρχιτεκτονικών ενός blockchain δικτύου. Για ένα τυπικό Ethereum blockchain αυτή είναι ως εξής:

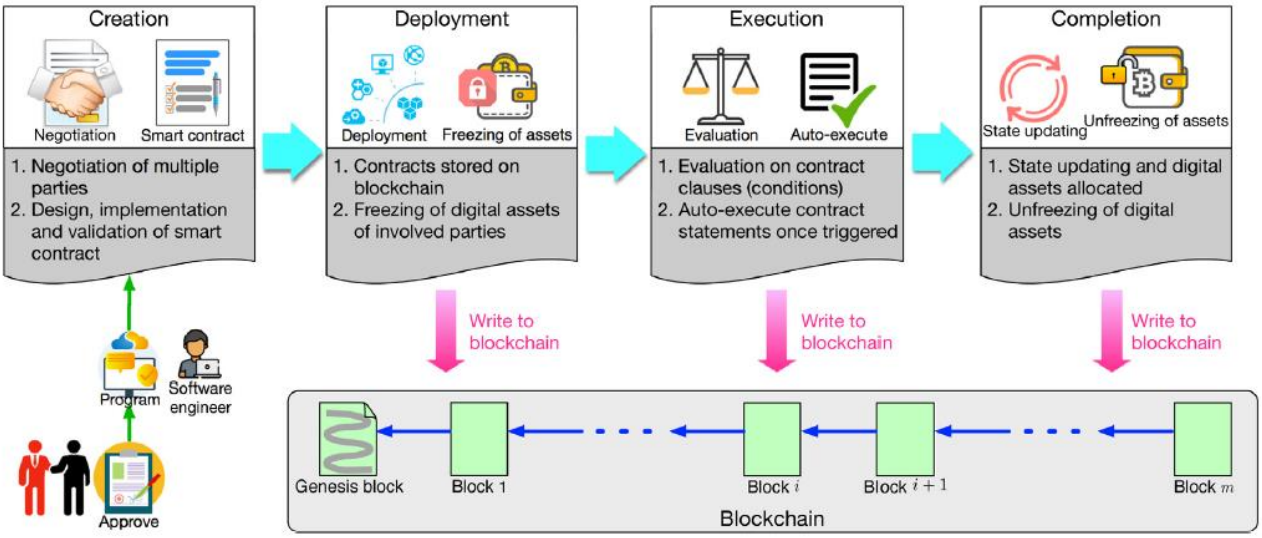


Εικόνα 15: Αρχιτεκτονική Δικτύου Ethereum [38].

Η παρούσα ενότητα θα εστιάσει στα έξυπνα συμβόλαια (Smart Contracts) που ανήκουν στο επίπεδο εφαρμογής και αποτελούν το θεμέλιο τόσο του Ethereum όσο και των Ethereum based δικτύων. Ένα έξυπνο συμβόλαιο πρακτικά είναι μια κοινή συμφωνία μεταξύ δύο ή περισσότερων οντοτήτων με την μορφή εκτελέσιμου κώδικα ο οποίος τρέχει εντός του blockchain. Αποσκοπεί στην διευκόλυνση, την εκτέλεση και την επίβλεψη μιας συμφωνίας μεταξύ μη αξιόπιστων συμμετεχόντων χωρίς την ανάγκη εμπλοκής ενός τρίτου αξιόπιστου μέρους. Τα έξυπνα συμβόλαια προσέφεραν αυτοματοποίηση ενός δικτύου όπως και την δυνατότητα μετατροπής των γραπτών συμβολαίων σε ψηφιακά. Σε σύγκριση με τα παραδοσιακά συμβόλαια, επιτρέπουν στους χρήστες να κωδικοποιήσουν τις συμφωνίες τους παρέχοντας αυτοματοποιημένες συναλλαγές χωρίς την επίβλεψη κάποια κεντρικής αρχής.

Προκειμένου να μην υπάρξει νοθεία αυτών των συμβολαίων, αυτά αντιγράφονται σε κάθε κόμβο του blockchain δικτύου. Έτσι, με την δυνατότητα εκτέλεσης των λειτουργιών από υπολογιστές και υπηρεσίες που παρέχονται από πλατφόρμες blockchain, εξασφαλίζεται η αμεταβλητότητα όπως και η σωστή εκτέλεση του συμβολαίου, εφόσον μειώνεται ο παράγοντας του ανθρώπινου λάθους. Πρακτικά, ένα έξυπνο συμβόλαιο μπορεί να θεωρηθεί ως μια κλάση που περιλαμβάνει μεταβλητές κατάστασης, τροποποιητές συναρτήσεων, γεγονότα και δομές. Σκοπός όλων αυτών είναι η διεκπεραίωση και ο έλεγχος όλων των σχετικών γεγονότων σύμφωνα με τους όρους του συμβολαίου [39].

Αναλυτικότερα, ο κύκλος ζωής ενός smart contract βάσει των Zibin Zheng, Shaoan Xie, Hong-Ning Dai et al στο ερευνητικό τους άρθρο με τίτλο «*An overview on smart contracts: Challenges, advances and platforms*» το 2020 [40] είναι ως εξής:



Εικόνα 16: Κύκλος Ζωής ενός Έξυπνου Συμβολαίου [40].

- **Στάδιο Δημιουργίας**: Σε αυτό το σημείο όλοι οι εμπλεκόμενοι φορείς διαπραγματεύονται τις υποχρεώσεις, τα δικαιώματα και τις απαγορεύσεις στα συμβόλαια. Νομικοί σύμβουλοι μπορεί να βοηθήσουν ώστε να συνταχθεί η αρχική συμφωνία και στην συνέχεια αυτή βγαίνει προς διαβούλευση με σκοπό την επίτευξη μιας συμφωνίας μεταξύ των μερών. Η συμφωνία αυτή, μεταφράζεται μέσω των κατάλληλων ειδικών σε ένα έξυπνο συμβόλαιο γραμμένο σε κάποια γλώσσα προγραμματισμού. Η διαδικασία μετατροπής μιας αρχικής συμφωνίας σε έξυπνο συμβόλαιο αποτελεί μια επαναλαμβανόμενη διεργασία που εμπλέκει πολλούς γύρους διαπραγματεύσεων και επαναλήψεων όπως και νομικές οντότητες και προγραμματιστές. Αποτελείται από τρία στάδια και πιο συγκεκριμένα την σχεδίαση, την υλοποίηση και την επικύρωση του συμβολαίου μέσω δοκιμών.

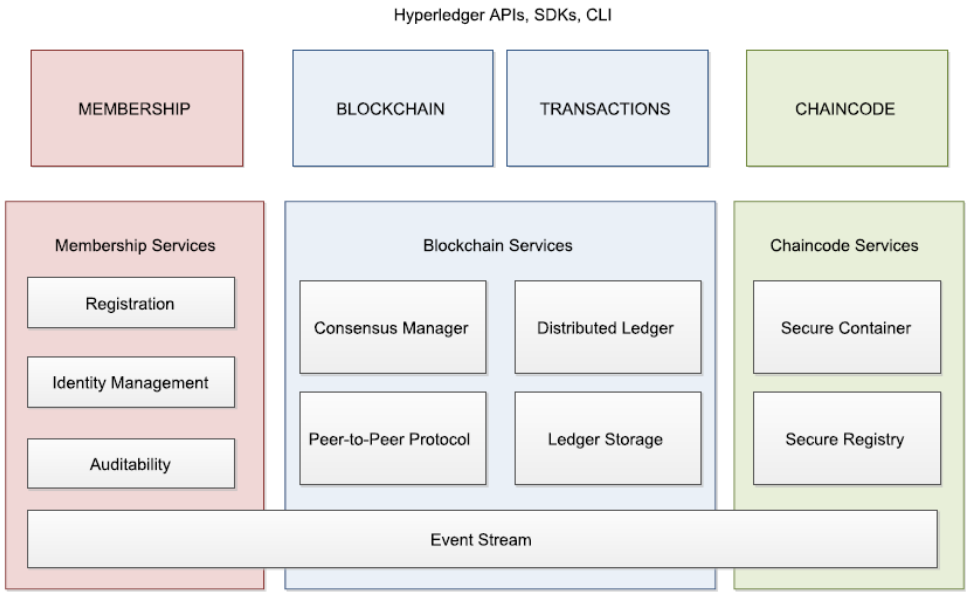
- **Στάδιο Μεταφόρτωσης**: Τα επικυρωμένα πλέον συμβόλαια μεταφορτώνονται στο δίκτυο blockchain. Στο στάδιο αυτό όλα τα μέρη μπορούν να έχουν πρόσβαση στα συμβόλαια τα οποία πλέον δεν μπορούν να τροποποιηθούν καθώς έχουν αποθηκευτεί εντός του blockchain. Κάθε τροποποίηση απαιτεί την δημιουργία ενός νέου συμβολαίου. Παράλληλα, τα ψηφιακά assets και των δύο πλευρών δεσμεύονται μέσω του κλειδώματος των αντίστοιχων ψηφιακών πορτοφολιών τους. Για παράδειγμα, μεταφορές κρυπτονομισμάτων εντός και εκτός πορτοφολιών σχετικών με το συμβόλαιο μπλοκάρονται.

- **Στάδιο Εκτέλεσης**: Στο στάδιο αυτό γίνεται ο έλεγχος και η αξιολόγηση των όρων της σύμβασης που έχει μεταφορτωθεί στο blockchain. Μόλις αυτές οι διαδικασίες ολοκληρωθούν και όλες οι συμβατικές ρήτρες έχουν επιτευχθεί (π.χ η παραλαβή του προϊόντος), εκτελούνται αυτόματα οι διαδικασίες που ορίζονται εντός του συμβολαίου. Τα έξυπνα συμβόλαια αποτελούνται από μια σειρά δηλώσεων που έχουν λογική διασύνδεση μεταξύ τους. Συνεπώς, όταν μια συνθήκη πυροδοτηθεί, η αντίστοιχη δήλωση θα εκτελεστεί αυτόματα με αποτέλεσμα η συναλλαγή να εκτελείται και να επικυρώνεται αυτόματα από τους εξορύκτες του δικτύου blockchain. Οι πλέον δεσμευμένες συναλλαγές και οι ενημερώσεις αυτών αποθηκεύονται εντός του blockchain από εκείνο το σημείο και έπειτα.

- **Στάδιο Ολοκλήρωσης**: Αφού ένα έξυπνο συμβόλαιο έχει εκτελεστεί, οι νέες καταστάσεις των εμπλεκόμενων μερών ενημερώνονται. Όπως αναφέρθηκε και προηγουμένως όλες οι εξελίξεις αλλά και οι συναλλαγές κατά την διάρκεια της εκτέλεσης των συμβολαίων εγγράφονται στο blockchain. Παράλληλα, τα ψηφιακά assets μεταφέρονται από το ένα μέρος στο άλλο (π.χ. η μεταφορά χρημάτων από τον αγοραστή στον προμηθευτή) και εν συνεχεία τα ψηφιακά πορτοφόλια των συμμετεχόντων ξεκλειδώνονται. Έτσι το έξυπνο συμβόλαιο έχει ολοκληρώσει τον κύκλο της ζωής του.

2.2.3 Τα projects του Hyperledger

Το Hyperledger αποτελεί μια από τις πιο γρήγορα αναπτυσσόμενες προσπάθειες ανοιχτού κώδικα που δημιουργήθηκε για να προωθήσει τις τεχνολογίες blockchain μεταξύ των κλάδων του εργασιακού χώρου. Δημιουργήθηκε από το Linux Foundation και στοχεύει στην δημιουργία ενός περιβάλλοντος όπου κοινότητες ανάπτυξης λογισμικού και εταιρίες θα συνεργάζονται με σκοπό την δημιουργία πλαισίων blockchain. Σημαντική διαφοροποίηση μεταξύ των κλασσικών προσεγγίσεων του blockchain και του Hyperledger είναι ότι το τελευταίο δεν υποστηρίζει κάποια μορφή κρυπτονομίσματος. Αντ' αυτού, στοχεύει στην δημιουργία μιας νέας γενιάς εφαρμογών που έχουν ως θεμέλιο τις βασικές αξίες του blockchain, διευκολύνοντας την παραγωγική διαδικασία και την αυτοματοποίηση της [37] [41].



Εικόνα 17: Αρχιτεκτονική του Hyperledger [41].

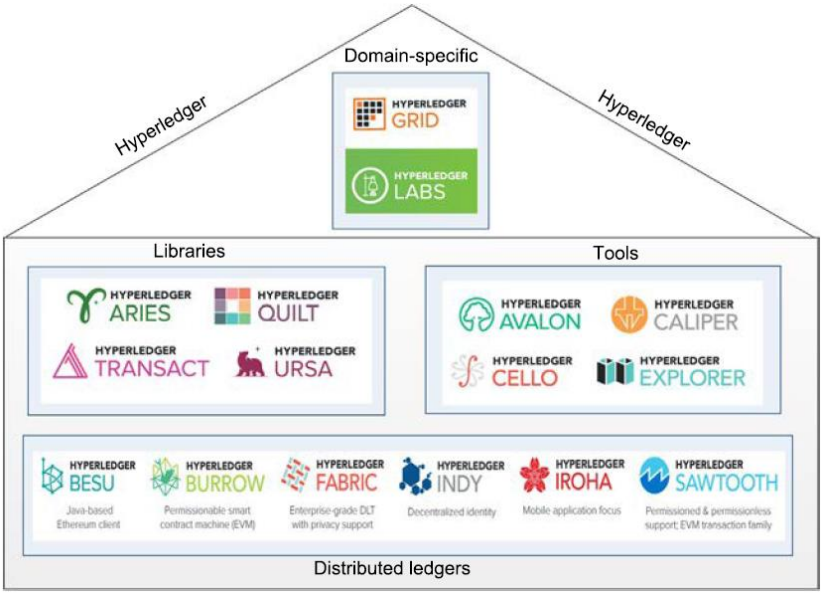
Κάτω από την ομπρέλα του Hyperledger υπάρχουν πολυάριθμα projects τα οποία διαχωρίζονται σε πλαίσια (frameworks) και εργαλεία/βιβλιοθήκες. Τρία από τα πιο βασικά πλαίσια τα οποία αξίζει να αναφέρουμε είναι τα:

- **Sawtooth**: Αποτελεί μια προσπάθεια της Intel για την δημιουργία ενός blockchain ανοιχτού κώδικα. Είναι μια modular πλατφόρμα για την δημιουργία, ανάπτυξη και εκτέλεση κατανεμημένων καταλόγων (ledgers). Περιλαμβάνει μια νέα προσέγγιση συγκριτικά με τους κλασσικούς αλγόριθμους συναίνεσης που ονομάζεται Proof of Elapsed Time (PoET) και στοχεύει σε μεγάλους πληθυσμούς κατανεμημένων επικυρωτών με ελάχιστη κατανάλωση πόρων. Τέλος, επιτρέπει την δημιουργία τόσο permissioned όσο και permissionless δικτύων σε μεγάλη κλίμακα [37].

- **Iroha**: Αποτελεί ένα συνεκτικό σύνολο από βιβλιοθήκες και εργαλεία που επιτρέπουν την υιοθέτηση τεχνολογιών blockchain σε ήδη υπάρχουσες υποδομές. Εστιάζει στην δημιουργία κινητών εφαρμογών με την αποθήκευση των δεδομένων και τον συγχρονισμό τους να πραγματοποιούνται εκτός συσκευής αξιοποιώντας ένα προεπιλεγμένο reputation σύστημα σε όλο το δίκτυο ώστε να διασφαλιστεί η επικύρωση των κόμβων [37].

- **Fabric**: Αποτελεί ένα από τα πιο σημαντικά και δημοφιλή projects του Hyperledger όπως και το επίκεντρο της παρούσας διπλωματικής. Αξιοποιείται για την αποδοτική και γρήγορη διαχείριση συναλλαγών ενώ διαφέρει από τα υπόλοιπα έργα του Hyperledger καθώς η λογική του είναι η δημιουργία ιδιωτικών και permissioned δικτύων μεταξύ οργανισμών. Είναι απαραίτητο για όλα τα μέλη του δικτύου να συνδεθούν μετά από την ταυτοποίηση τους μέσω κάποιου έγκυρου παρόχου υπηρεσιών μέλους (Membership service provider-MSP) [41].

Στην εικόνα 18 παρουσιάζονται τα projects του Hyperledger όπως και ο διαχωρισμός τους βάσει της λειτουργίας τους.



Εικόνα 18: Hyperledger Projects [41].

Συνεπώς, βλέπουμε πως το Hyperledger αποτελεί μια σύγχρονη και καινοτόμα προσέγγιση της τεχνολογίας του blockchain με πολυάριθμες εφαρμογές σε κάθε έκφανση του σύγχρονου γίγνεσθαι. Το επόμενο κεφάλαιο θα εστιάσει στην πλατφόρμα του Hyperledger Fabric που αποτελεί το βασικό αντικείμενο της παρούσας διπλωματικής.

Κεφάλαιο 3

Hyperledger Fabric

Στο συγκεκριμένο κεφάλαιο θα αναλυθεί διεξοδικά η πλατφόρμα του Hyperledger Fabric, μία από τις πλέον διαδεδομένες και ευέλικτες πλατφόρμες για την ανάπτυξη εφαρμογών που βασίζονται στην τεχνολογία του blockchain. Η παρουσίαση θα επικεντρωθεί στις διαφοροποιήσεις και τα πλεονεκτήματα που προσφέρει έναντι άλλων προσεγγίσεων, δίνοντας έμφαση στα χαρακτηριστικά και στις προκλήσεις που προκύπτουν από την χρήση της. Μέσα σε αυτό το πλαίσιο, θα εξεταστεί το θεωρητικό υπόβαθρο που υποστηρίζει την υλοποίηση του δικτύου της παρούσας διπλωματικής.

3.1 Εισαγωγή στο Hyperledger Fabric

Το Hyperledger Fabric αποτελεί ένα ιδιωτικό και permissioned blockchain το οποίο παρουσιάζει μια νέα αρχιτεκτονική η οποία στοχεύει στην δημιουργία ενός ανθεκτικού, εμπιστεύσιμου, ευέλικτου και επεκτάσιμου δικτύου. Έχει δημιουργηθεί ώστε να υποστηρίζει πολλαπλές λειτουργίες αξιοποιώντας τον modular τρόπο σχεδίασής του. Αποτελεί το πρώτο σύστημα blockchain το οποίο υποστηρίζει την εκτέλεση κατανεμημένων εφαρμογών χρησιμοποιώντας προγραμματιστικές γλώσσες γενικού σκοπού, με τέτοιον τρόπο ώστε να υπάρχει συνεπής εκτέλεσή τους σε πολλαπλούς κόμβους [42] [43]. Βάσει των Xiaojie Zhao, Shangping Wang, Yaling Zhang και Yu Wang στο ερευνητικό τους άρθρο με τίτλο «*Attribute-based access control scheme for data sharing on Hyperledger Fabric*» το 2022 [44] το Hyperledger Fabric διακρίνεται από τα εξής πλεονεκτήματα έναντι των κλασσικών προσεγγίσεων:

- **Εξάλειψη των εξορυκτών (miners) και των κινήτρων**: Το Fabric υιοθετεί μια νέα μέθοδο εκτέλεσης συναλλαγών ακολουθώντας το μοντέλο execute-order-validate. Αρχικά ο κόμβος που είναι υπεύθυνος για την έγκριση εκτελεί την συναλλαγή ώστε να επαληθεύσει την ορθότητα της και στην συνέχεια αυτή ταξινομείται στο συνδεόμενο πρωτόκολλο συναίνεσης. Στην συνέχεια η συναλλαγή επιβεβαιώνεται βάσει της προσυμφωνημένης στρατηγικής έγκρισης. Σημαντικό είναι το γεγονός πως οι συναλλαγές εκτελούνται παράλληλα και χρειάζονται μόνο ένα μέρος των κόμβων, όπως ορίζει η πολιτική της έγκρισης του συστήματος, γεγονός που επιτρέπει την υψηλή απόδοση και την επεκτασιμότητα του δικτύου.

- **Αυστηροί κανόνες πρόσβασης**: Μέσω της αξιοποίησης παρόχου υπηρεσιών μέλους (Membership service provider-MSP) επιτρέπεται ο έλεγχος και η οριοθέτηση των δικαιωμάτων πρόσβασης ενός χρήστη, ο οποίος μπορεί να συνδεθεί στο δίκτυο μόνο έπειτα από την ταυτοποίηση του.

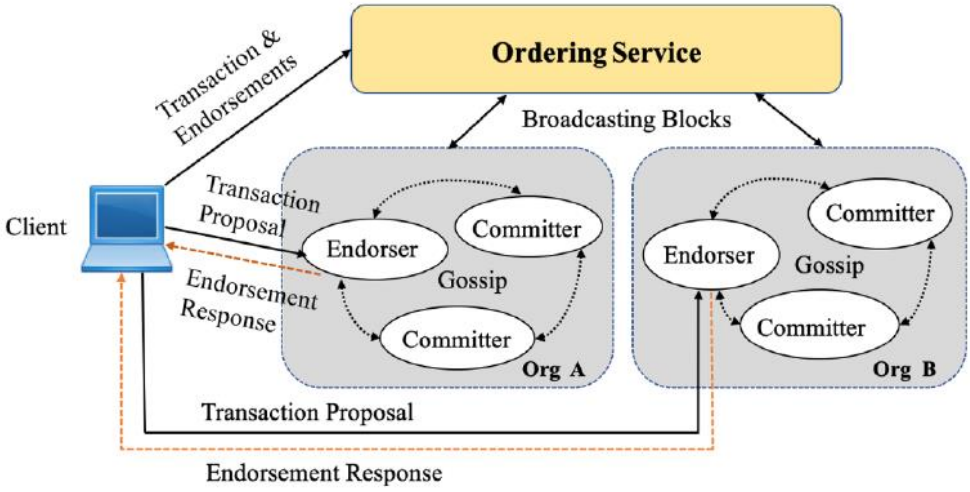
- **Υποστήριξη προγραμματιστικών γλωσσών γενικού σκοπού**: Σε αντίθεση με τους προκάτοχούς του, το Hyperledger Fabric υποστηρίζει γλώσσες γενικού σκοπού όπως η Java, Go, NodeJS κλπ. για την δημιουργία των έξυπνων συμβολαίων διευκολύνοντας έτσι την ανάπτυξη εφαρμογών.

- **Συστήματα αποθήκευσης υψηλής απόδοσης**: Το world state αξιοποιείται για να περιγράψει την κατάσταση λογαριασμού (account state) σε μια χρονική περίοδο και τα world states διαφορετικών chaincodes δεν έχουν πρόσβαση το ένα στο άλλο. Παράλληλα, υποστηρίζει πολυάριθμα συστήματα διαχείρισης βάσεων δεδομένων όπως το GoLevelDB και το CouchDB.

- **Ενσωματωμένη υπηρεσία ταξινόμησης**: Για τον καθορισμό της παγκόσμιας σειράς των συναλλαγών αξιοποιείται ένα χωριστό πρωτόκολλο συναίνεσης (όπως το Raft, Kafka) το οποίο είναι ανεξάρτητο από την εκτέλεση και την επιβεβαίωση των συναλλαγών.

3.2 Βασικά δομικά στοιχεία ενός Hyperledger Fabric δικτύου

Μια τυπική δομή ενός δικτύου Hyperledger Fabric παρουσιάζεται στην εικόνα 19.



Εικόνα 19 Παράδειγμα ενός Fabric blockchain με δύο οργανισμούς(Org A, Org B) [45].

Στο παράδειγμα φαίνεται ένα δίκτυο που αποτελείται από δύο οργανισμούς (Org A, Org B) καθένας από τους οποίους αποτελείται από τρεις peers: ο ένας είναι υπεύθυνος για το endorsement και ονομάζεται επικυρωτής (endorser) ενώ οι υπόλοιποι απλώς πραγματοποιούν το commit της συναλλαγής. Ένας πελάτης (client) στέλνει μια πρόταση συναλλαγής σε peers οι οποίοι έχουν καθοριστεί από την πολιτική έγκρισης. Εν συνεχεία, οι peers εκτελούν αυτή την συναλλαγή καλώντας ένα προκαθορισμένο κώδικα αλυσίδας ή chaincode. Μετά την εκτέλεση, ο πελάτης συγκεντρώνει αρκετά αποτελέσματα έγκρισης και υποβάλλει την συναλλαγή στην υπηρεσία ταξινόμησης (ordering service). Σε αυτό το σημείο, οι orderers χρησιμοποιούν έναν συνδεόμενο μηχανισμό συναίνεσης για να δημιουργήσουν μία

καθολική σειρά για όλες τις συναλλαγές ώστε να δημιουργηθούν τα blocks. Τα blocks αυτά μεταδίδονται σε όλους τους peers χρησιμοποιώντας το πρωτόκολλο Gossip και κάθε peer επικυρώνει τις συναλλαγές εντός των μπλοκ και ενημερώνει την κατάσταση του ledger [45].

Σημαντική επομένως είναι η κατανόηση των βασικών εννοιών ενός Hyperledger δικτύου οι οποίες είναι οι εξής:

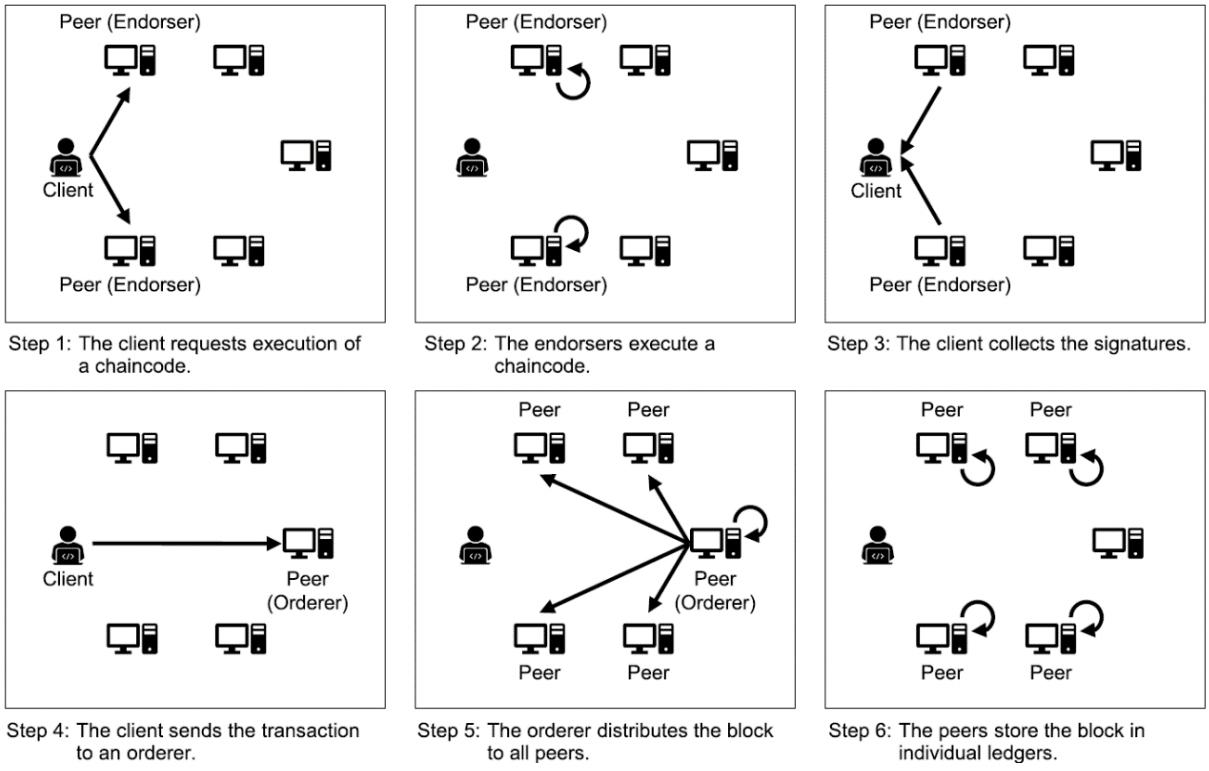
- **Κόμβοι (Nodes)**
- **Membership Service Provider (MSP)**
- **Κανάλι (Channel)**
- **Πολιτικές έγκρισης (Endorsement policies)**
- **Ordering Service**
- **Κώδικας αλυσίδας (Chaincode)**

3.2.1 **Η σημασιολογία των κόμβων σε ένα Fabric δίκτυο**

Ένα Hyperledger Fabric δίκτυο αποτελείται από ένα σύνολο κόμβων οι οποίοι αναλαμβάνουν έναν από τρείς ρόλους:

1. **Client**
2. **Peer**
3. **Orderer**

Αυτοί οι ρόλοι έχουν διαφορετικές λειτουργίες κατά τη διάρκεια της εκτέλεσης μιας συναλλαγής.



Εικόνα 20: Διαδικασία Συναλλαγής εντός του Hyperledger Fabric [46].

Στην εικόνα 20, παρουσιάζεται η διαδικασία που ακολουθείται κατά την διάρκεια εκτέλεσης μιας συναλλαγής εντός του Hyperledger Fabric. Αναλυτικότερα:

- Ο **client** είναι υπεύθυνος για την δημιουργία και την υποβολή των συναλλαγών. Πιο συγκεκριμένα, κατά την διάρκεια εκτέλεσης μιας συναλλαγής ο client υπογράφει και στέλνει μια πρόταση συναλλαγής σε έναν ή περισσότερους επικυρωτές (endorsers) ώστε να εκτελεστεί. Εν συνεχεία, λαμβάνει όλες τις επικυρώσεις και εξετάζει κατά πόσο έχουν εκπληρωθεί όλες οι απαιτήσεις σύμφωνα με την πολιτική έγκρισης και του κώδικα αλυσίδας [43] .

- Οι **peers** είναι υπεύθυνοι για την εκτέλεση και την επικύρωση των συναλλαγών. Διατηρούν το ledger του blockchain που όπως έχει αναφερθεί αποτελεί μια δομή δεδομένων στην οποία καταγράφονται όλες οι συναλλαγές υπό την μορφή μιας αλυσίδας μπλοκ. Παράλληλα, διατηρούν έναν δείκτη ενημέρωσης που ονομάζεται state και αποτελεί μια συνοπτική αναπαράσταση της τωρινής κατάστασης του ledger [42]. Χωρίζονται σε δύο βασικές κατηγορίες:
    1. **Endorsing Peers**: Είναι υπεύθυνοι για την εκτέλεση των συναλλαγών που προέρχονται από τους clients. Είναι η μόνη κατηγορία peer στην οποία πρέπει να έχει εγκατασταθεί ο κώδικας αλυσίδας (chaincode) και οι μόνοι που μπορούν να τον εκτελέσουν. Μετά την εκτέλεση του αιτήματος επιστρέφουν στον client ένα επικυρωμένο αποτέλεσμα [47].
    2. **Committing Peers**: Είναι υπεύθυνοι για τον έλεγχο της εγκυρότητας όλων των συναλλαγών που περιέχονται στο ληφθέν μπλοκ το οποίο στην συνέχεια εντάσσουν στο ledger. Όλοι οι peers αναλαμβάνουν αυτόν τον ρόλο γεγονός που δεν ισχύει και για τους endorsing peers [47].

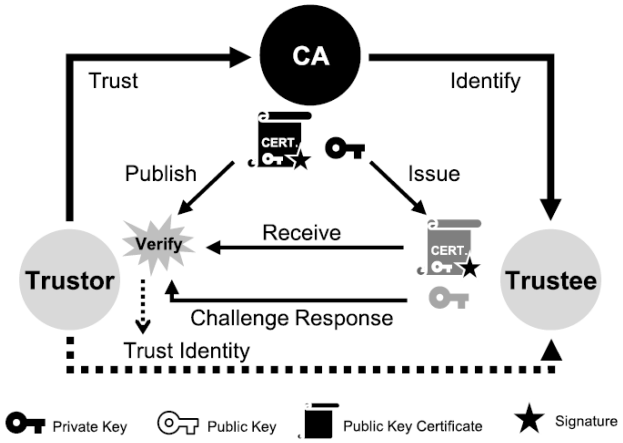
- Οι **orderers** αποτελούν ένα σύμπλεγμα κόμβων και μια ειδική κατηγορία peer, οι οποίοι είναι υπεύθυνοι για την καθιέρωση μια καθολικής σειράς μεταξύ όλων των συναλλαγών στο Hyperledger Fabric. Οι κόμβοι αυτοί είναι εντελώς ανενημέρωτοι για την κατάσταση της εφαρμογής και δεν συμμετέχουν στην επικύρωση ή εκτέλεση των συναλλαγών [48]. Πιο συγκεκριμένα, **οι orderers** υλοποιούν την υπηρεσία ταξινόμησης (ordering service) βάσει ενός συνδεόμενου μηχανισμού συναίνεσης (consensus mechanism) ώστε να παραχθεί μια συνολική ακολουθία όλων των συναλλαγών και να τις ομαδοποιήσουν σε blocks [45].

3.2.2 Η έννοια της ταυτοποίησης σε ένα fabric δίκτυο – MSPs

Μια επόμενη πολύ σημαντική ενότητα είναι αυτή της ταυτοποίησης ενός χρήστη εντός ενός δικτύου Hyperledger Fabric.

Ο **πάροχος υπηρεσιών μέλους**, γνωστός και ως **Membership Service Provider (MSP)**, διαχειρίζεται και διατηρεί τις ταυτότητες όλων των κόμβων στο δίκτυο, περιλαμβάνοντας clients, peers και OSNs. Είναι υπεύθυνος για την έκδοση των πιστοποιητικών κόμβων, τα οποία χρησιμοποιούνται για την εξουσιοδότηση και την πιστοποίηση στο δίκτυο. Καθώς το Fabric αποτελεί ένα permissioned δίκτυο, όλες οι

αλληλεπιδράσεις μεταξύ των κόμβων πραγματοποιούνται μέσω μηνυμάτων που είναι πιστοποιημένα, συνήθως με ψηφιακές υπογραφές. Συνεπώς, η υπηρεσία μέλους αποτελεί ένα αναπόσπαστο μέρος κάθε κόμβου ο οποίος με την σειρά του μπορεί να αυθεντικοποιήσει και να επαληθεύσει την εγκυρότητα συναλλαγών, να υπογράψει και να επικυρώσει εγκρίσεις όπως και να αυθεντικοποιήσει άλλες λειτουργίες του blockchain. Εργαλεία τόσο για την διαχείριση των κλειδιών όσο και για την εγγραφή των κόμβων είναι μέρος του MSP. Πρακτικά, το MSP αποτελεί μια αφαιρετική έννοια για την οποία είναι δυνατές διαφορετικές υλοποιήσεις. Η προεπιλεγμένη υλοποίηση MSP στο Fabric χειρίζεται τις τυπικές μεθόδους PKI για αυθεντικοποίηση με βάση ψηφιακές υπογραφές και μπορεί να προσαρμοστεί σε εμπορικές αρχές πιστοποίησης (CAs) [42]. Αναλυτικότερα, η Δημόσια Υποδομή Κλειδιών (Public Key Infrastructure, PKI) χρησιμοποιείται κυρίως για την εγγύηση των ταυτοτήτων. Πρόκειται για μια υποδομή στην οποία μια αρχή πιστοποίησης (Certificate Authority, CA), η οποία αποτελεί έναν αξιόπιστο τρίτο φορέα, εγγυάται την σχέση μεταξύ μιας οντότητας και ενός δημόσιου κλειδιού. Παράλληλα, περιλαμβάνει και μηχανισμούς όπως για την επαλήθευση των επικοινωνούντων συνεργατών μέσω ψηφιακών υπογραφών. Κάθε CA πρέπει να είναι προετοιμασμένη για απειλές από εσωτερικούς και εξωτερικούς παράγοντες, αναπτύσσοντας τα κατάλληλα μέτρα ασφάλειας [46].



Εικόνα 21: Αρχή Πιστοποίησης CA [46].

Η εικόνα 21 παρουσιάζει τον τρόπο λειτουργίας μιας Αρχής Πιστοποίησης (CA). Όταν αυτή λαμβάνει ένα αίτημα υπογραφής πιστοποιητικού, επιβεβαιώνει την ταυτότητα του αιτούντος και δημιουργεί ένα αντίστοιχο πιστοποιητικό. Η CA εγγυάται την ταυτότητα υπογράφοντας το πιστοποιητικό με ένα ιδιωτικό κλειδί της. Έτσι, ο παραλήπτης (trustor) μπορεί να επιβεβαιώσει την ταυτότητα του αποστολέα (trustee), λαμβάνοντας το πιστοποιητικό του και ελέγχοντας τον εκδότη του. Εάν ο εκδότης επιβεβαιωθεί ως μια αξιόπιστη CA από τον παραλήπτη και ο αποστολέας διαθέτει ένα ιδιωτικό κλειδί για αυτό το πιστοποιητικό, τότε υπάρχει επαλήθευση της ταυτότητας. Σε ένα μεγάλης έκτασης σύστημα PKI το οποίο διαθέτει πολλαπλές CAs, μια CA καθιερώνει σχέσεις εμπιστοσύνης με μια άλλη CA μέσω του cross-certification ή διασταυρούμενης πιστοποίησης [46].

Συμπερασματικά, η Αρχή Πιστοποίησης (Certificate Authority, CA) είναι ένα από τα πιο σημαντικά στοιχεία σε ένα σύστημα Δημόσιας Υποδομής Κλειδιών (PKI) και έχει

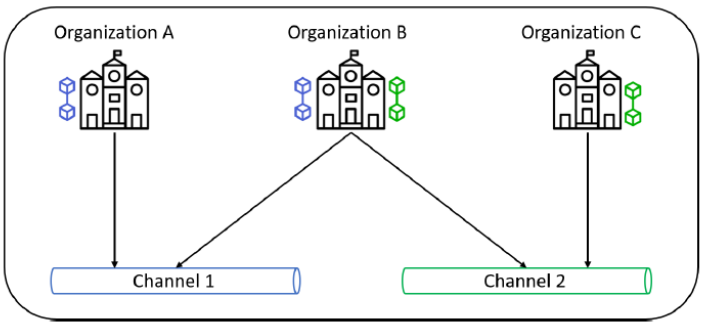
τη δυνατότητα να εκδίδει και να επαληθεύει πιστοποιητικά για όλους τους συμμετέχοντες, που περιλαμβάνουν users ή clients αποκεντρωμένων εφαρμογών (DApp), κόμβους συναλλαγών (peers) και κόμβους ταξινόμησης (ordering nodes). Η Fabric CA δημιουργεί ένα ζεύγος δημόσιου και ιδιωτικού κλειδιού για κάθε συμμετέχοντα. Το ιδιωτικό κλειδί διατηρείται από τον συμμετέχοντα, ενώ το δημόσιο κλειδί είναι προσβάσιμο σε όλους. Όταν ένας πελάτης DApp επιθυμεί να πραγματοποιήσει μια συναλλαγή, πρέπει να την υπογράψει με το ιδιωτικό του κλειδί προκειμένου αυτή μαζί με την υπογραφή του να υποβληθεί στο δίκτυο του Fabric. Στη συνέχεια, η Fabric CA επαληθεύει την ταυτότητα αυτού που υποβάλλει την συναλλαγή, ελέγχοντας την υπογραφή. Η συναλλαγή θα απορριφθεί εάν ο υποβάλλων δεν ανήκει στην κοινοπραξία (consortium) του blockchain του Fabric [49].

3.2.3 Η έννοια του καναλιού (channel)

Το Hyperledger Fabric εντάσσει την έννοια του καναλιού (channel) εντός της λογικής του. Πρακτικά, αποτελεί ένα υπό-δίκτυο μεταξύ μιας ομάδας οργανισμών με σκοπό την ιδιωτική επικοινωνία τους εντός του blockchain δικτύου. Κάθε κανάλι αποτελείται από τα εξής:

- Τους συμμετέχοντες οργανισμούς (organizations).
- Τους peers που ανήκουν στον κάθε οργανισμό.
- Το κοινό ledger των συμμετεχόντων.
- Τους κόμβους που συντελούν την υπηρεσία ταξινόμησης (ordering service).

Κάθε συναλλαγή εντός του δικτύου πραγματοποιείται σε ένα κανάλι, όπου οι συμμετέχοντες πρέπει να είναι αυθεντικοποιημένοι και εξουσιοδοτημένοι για να προβούν σε αυτή [50]. Κάτι τέτοιο καθίσταται εφικτό καθώς κάθε κανάλι διατηρεί το δικό του ledger και μόνο τα μέλη του καναλιού έχουν αντίγραφο του. Τα μέλη ορίζονται στην πολιτική του καναλιού κατά την δημιουργία του μπλοκ διαμόρφωσης (configuration block) το οποίο ορίζει και τον τύπο υπηρεσίας ταξινόμησης. Οποτεδήποτε τροποποιείται αυτό το μπλοκ διαμόρφωσης όπως με προσθήκη νέων μελών τότε δημιουργείται ένα νέο μπλοκ και προστίθεται στην αλυσίδα [51].



Εικόνα 22: Παράδειγμα καναλιών εντός του Hyperledger Fabric [51].

Στην εικόνα 22 παρουσιάζεται ένα δίκτυο Hyperledger Fabric όπου έχουμε τρεις οργανισμούς και δύο διαθέσιμα κανάλια.

Οι οργανισμοί Α και Β έχουν πρόσβαση στο κανάλι 1 και οι οργανισμοί Β και Γ στο κανάλι 2. Οι peers τόσο του οργανισμού Α όσο και του οργανισμού Γ έχουν ένα αντίγραφο του ledger για το κανάλι που ανήκουν και πιο συγκεκριμένα, το κανάλι 1 για τους peers του οργανισμού Α και το κανάλι 2 για αυτούς του οργανισμού Γ. Εν αντιθέσει, οι peers του οργανισμού Β έχουν ένα αντίγραφο και από τα δύο ledgers των καναλιών 1 και 2 καθώς ο οργανισμός τους συμμετέχει και στα δύο κανάλια [51].

Συνεπώς, τα κανάλια μπορούν να χρησιμοποιηθούν για την διαίρεση της κατάστασης του δικτύου blockchain, χωρίς όμως να υπάρχει συντονισμός συναίνεσης μεταξύ των καναλιών, ενώ η συνολική σειρά των συναλλαγών για κάθε κανάλι είναι διαφορετική από τα υπόλοιπα [42].

3.2.4 Οι πολιτικές έγκρισης εντός του Hyperledger Fabric

Οι πολιτικές έγκρισης (endorsement policies) αποτελούν το βασικό εργαλείο του Hyperledger Fabric ώστε να μπορέσει να υιοθετήσει σωστά το μοντέλο ασφάλειας οποιασδήποτε επιχειρηματικής λογικής που πραγματοποιείται μέσω των έξυπνων συμβολαίων. Πιο συγκεκριμένα, επιτρέπει την επιλογή ενός συνόλου κόμβων οι οποίοι απαιτείται να συμφωνήσουν στο αποτέλεσμα μιας κλήσης ενός έξυπνου συμβολαίου ώστε αυτή να θεωρείται σωστή από το δίκτυο. Monotone formula

Πρακτικά, μια πολιτική έγκρισης αποτελεί μια απλή φόρμουλα μοντελοποιημένη ως δέντρο. Κάθε φύλλο του δέντρου αποτελεί μια βασική αρχή της πολιτικής (policy principal). Οι αρχές αυτές παρέχουν τον τρόπο με το οποίο ο ελεγκτής της πολιτικής θα αναγνωρίσει κατά πόσο το κρυπτογραφημένο αντικείμενο ικανοποιεί την πολιτική αυτή. Παραδείγματος χάριν, μια βασική αρχή μπορεί να είναι η απαίτηση κάποιος να είναι μέλος ενός οργανισμού ή να είναι μια οντότητα με έναν συγκεκριμένο ρόλο ή ένα συγκεκριμένο είδος κόμβου εντός του Fabric δικτύου. Όλοι οι υπόλοιποι κόμβοι του δέντρου αποτελούν τα λεγόμενα threshold nodes που προσομοιώνουν AND και OR πύλες. Πιο συγκεκριμένα, κάθε τέτοιος κόμβος συσχετίζεται με μία παράμετρο t η οποία είναι ένας θετικός ακέραιος μικρότερος ή ίσος με τον υπερβάθμιο κόμβο (outdegree node). Μόλις τουλάχιστον t από τους διαδοχικούς κόμβους ικανοποιηθούν τότε ικανοποιείται και ο συγκεκριμένος threshold κόμβος. Η πολιτική έγκρισης ικανοποιείται όταν η ρίζα του δέντρου έχει ικανοποιηθεί [52].

Οι πολιτικές έγκρισης μπορούν να γραφούν χρησιμοποιώντας μια γλώσσα ειδικού σκοπού που αποτελείται από (α) AND, OR και OutOf εκφράσεις, (β) έναν ελάχιστο αριθμό απαιτούμενων υποστηρικτών (endorsers) που χρειάζεται για την ικανοποίηση της πολιτικής και (γ) έναν κατάλογο που περιέχει τους ρόλους αυτών των υποστηρικτών σχετικά με την πολιτική. Με αυτόν τον τρόπο μπορεί να δημιουργηθούν οι κατάλληλες πολιτικές οι οποίες εκφράζουν καταστάσεις της πραγματικής ζωής. Για παράδειγμα, όταν ένα άτομο υποβάλλει αίτηση για ένα νέο διαβατήριο, ένας κυβερνητικός οργανισμός θα πρέπει να επικυρώσει την ταυτότητά του πριν από την έγκριση της συναλλαγής. Για το σκοπό αυτό, η πολιτική θα μπορούσε να χρησιμοποιήσει μια έκφραση OR μεταξύ διαφορετικών εξουσιοδοτημένων οργανισμών που θα μπορούσαν να εγκρίνουν την ταυτότητα και το αίτημα του ατόμου. Ως άλλο παράδειγμα, κατά την πώληση ενός αυτοκινήτου

τόσο ο αγοραστής όσο και ο πωλητής πρέπει να συμφωνήσουν για την τιμή που αναφέρεται στη συναλλαγή, ενώ και οι δύο πρέπει να έχουν ενεργό ρόλο στην πολιτική έγκρισης. Συνεπώς η συναλλαγή αυτή δεν πρέπει να προστεθεί στο ledger χωρίς τη συγκατάθεση και των δύο (έκφραση AND) [53].

```
// endorsement required by all these peers
AND('Org1.peer', 'Org2.peer', 'Org3.peer')

// endorsement by any of these couples of peers
OR(
  AND('Org1.peer', 'Org2.peer'),
  AND('Org2.peer', 'Org3.peer'),
  AND('Org3.peer', 'Org4.peer')
)

// endorsement by three out of four referenced peers
OutOf(3, 'Org1.peer', 'Org2.peer', 'Org3.peer',
        'Org4.peer')
```

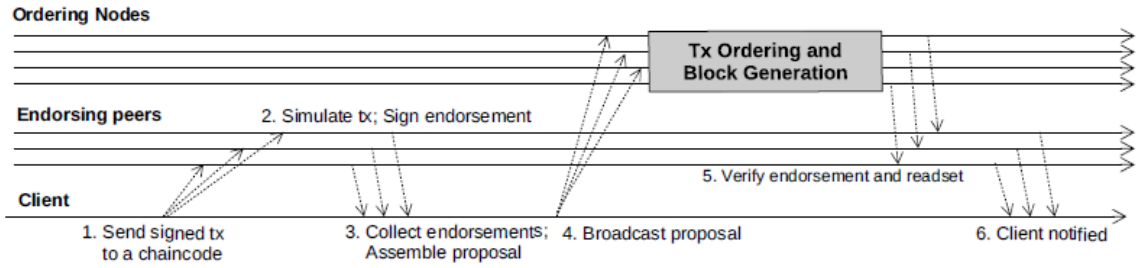
Εικόνα 23: Παράδειγμα Πολιτικής Έγκρισης στο Hyperledger Fabric [53].

Στην εικόνα 23 παρουσιάζεται ένα παράδειγμα τριών πολιτικών που θα μπορούσαν να εφαρμοστούν σε ένα δίκτυο Hyperledger Fabric.

- Στην 1η περίπτωση βλέπουμε μια πολιτική που απαιτεί την έγκριση από τρεις peers διαφορετικών οργανισμών.
- Στην 2η περίπτωση βλέπουμε μια πολιτική που απαιτεί την έγκριση από δύο peers οι οποίοι μπορούν να έχουν διαφορετικούς συνδυασμούς. Δηλαδή, μπορεί να είναι ένας peer του οργανισμού 1 και του οργανισμού 2 ή ένας peer του οργανισμού 2 και του οργανισμού 3 κ.ο.κ.
- Στην 3η και τελευταία περίπτωση βλέπουμε πως χρειαζόμαστε την συμφωνία τριών από τους αναγραφόμενους peers των οργανισμών σε οποιονδήποτε συνδυασμό.

3.2.5 Το Ordering Service στο Hyperledger Fabric

Για την επίτευξη της συναίνεσης (consensus) το Hyperledger Fabric εισάγει την έννοια των κόμβων ταξινόμησης (ordering nodes) οι οποίοι στο σύνολό τους ορίζουν την υπηρεσία ταξινόμησης (ordering service). Ο σκοπός και η κύρια ευθύνη αυτής της υπηρεσίας είναι η οργάνωση των συναλλαγών σε μπλοκ και η μετάδοση τους στους peers για επικύρωση [51].



Εικόνα 24: Το Ordering Service στο Hyperledger Fabric [54].

Το Hyperledger Fabric προσφέρει τρεις διαφορετικές εφαρμογές της υπηρεσίας ταξινόμησης [50] [51]:

- **Solo**: Σε αυτή την εκδοχή, υπάρχει μόνο ένας κόμβος ταξινόμησης γεγονός που το καθιστά μη ρεαλιστικό σε πραγματικά σενάρια και αξιοποιείται μόνο για πειραματικούς σκοπούς.

- **Kafka**: Αυτή η εκδοχή αποτελεί μια Crash Fault Tolerant ή CFT προσέγγιση που βασίζεται στην λογική του ηγέτη (leader) και των ακολούθων (followers) όπου ο leader orderer στέλνει τις συναλλαγές του στους follower orderers. Η επιλογή του ηγέτη γίνεται με δυναμικό τρόπο και για την διαχείριση των clusters αξιοποιείται η υπηρεσία συντονισμού Zookeeper. Η υπηρεσία αυτή αποσκοπεί στην διατήρηση των πληροφοριών διαμόρφωσης, την βοήθεια στο συντονισμό των εργασιών όπως και την παροχή κατανεμημένου συγχρονισμού και την ένταξη σε ομάδες.

- **Raft**: Αποτελεί την νεότερη εκδοχή που αξιοποιείται στο Hyperledger Fabric και βασίζεται στο πρωτόκολλο raft. Όπως και στην εκδοχή του Kafka το Raft έχει το πλεονέκτημα ότι είναι μια CFT προσέγγιση η οποία ακολουθεί την λογική του ηγέτη και των ακολούθων. Αυτό που την ξεχωρίζει από τον προκάτοχό της είναι ο απλούστερος τρόπος εγκατάστασης καθώς ο Kafka προσθέτει διοικητικό βάρος στην διαχείριση, γεγονός που τον καθιστά δύσκολο στην υλοποίηση απαιτώντας κάποιον ειδικό για την εγκατάσταση του.

3.2.6 Η έννοια των έξυπνων συμβολαίων (Smart Contracts)

Στο Hyperledger Fabric τα έξυπνα συμβόλαια, γνωστά ως κώδικας αλυσίδας (chaincode), είναι ένα κομμάτι κώδικα το οποίο υλοποιεί την επιχειρηματική λογική για την δημιουργία και επεξεργασία λογικών αντικειμένων στο ledger. Σε αντίθεση με άλλες προσεγγίσεις το Hyperledger Fabric παρέχει ένα σύστημα έξυπνων συμβολαίων στο οποίο αξιοποιούνται γλώσσες γενικού σκοπού όπως η Java, Go και NodeJS και όχι κάποια ad hoc γλώσσα όπως στο Ethereum [47].

Το chaincode εγκαθίσταται και αρχικοποιείται στους peers κάποιου καναλιού και για την εκτέλεση του απαιτείται:

- Να πραγματοποιηθεί από κάποιο αυθεντικοποιημένο μέλος που του επιτρέπεται η εκτέλεση του κώδικα.
- Να πραγματοποιηθεί μέσω του Fabric chaincode life cycle.

Αποτελείται από δύο βασικές συναρτήσεις: Init και Invoke. Η πρώτη συνάρτηση αξιοποιείται για την αρχικοποίηση κάποιας δομής δεδομένων (data structure) που μπορεί να χρειάζεται το chaincode και καλείται μόνο μία φορά, κατά την αρχικοποίηση. Η δεύτερη συνάρτηση αποτελεί το αρχικό σημείο της επιχειρηματικής λογικής του κώδικα και αναλόγως με τα δεδομένα που έχει λάβει, το chaincode αποφασίζει ποια διεργασία θα ακολουθήσει. Κάθε chaincode πρέπει να ορίζει μια

πολιτική έγκρισης η οποία ελέγχεται από τους commiters κατά το στάδιο επιβεβαίωσης. Μια τυπική πολιτική έγκρισης, όπως αναφέρθηκε και στην ενότητα 3.2.4, επιτρέπει στον κώδικα αλυσίδας να προσδιορίζει τους endorsers που χρειάζεται παρέχοντας έτσι μια σωστή εκτέλεση βάσει της επιχειρηματικής λογικής που είναι μέρος του κώδικα [55].

Στο σημείο αυτό, είναι σημαντικό να αναλύσουμε τον κύκλο ζωής μιας συναλλαγής (transaction life cycle) και τα στάδια που χρειάζονται ώστε να επιτύχουμε την εγγραφή μιας συναλλαγής, καθώς αποτελεί τον βασικό πυλώνα των έξυπνων συμβολαίων [43] [55].

1. **Έναρξη συναλλαγής**: Ο πελάτης δημιουργεί ένα αίτημα συναλλαγής για να καλέσει (invoke) μια συνάρτηση ενός chaincode. Στο αίτημα αυτό προσδιορίζονται τα δεδομένα που χρειάζονται για την εκτέλεση του κώδικα. Στην συνέχεια, το αίτημα υπογράφεται από τον πελάτη και υποβάλλεται στο κανάλι στο οποίο έχει εγκατασταθεί το chaincode. Σύμφωνα με την πολιτική έγκρισης για το συγκεκριμένο chaincode, ο πελάτης περιμένει ένα σύνολο από εγκρίσεις.

2. **Εκτέλεση συναλλαγής**: Ο πελάτης υποβάλλει την υπογεγραμμένη πλέον συναλλαγή στους endorsing peers. Κάθε endorsing peer επαληθεύει εάν ο πελάτης είναι εξουσιοδοτημένος να εκτελέσει την συναλλαγή και εκτελεί ή προσομοιάζει το αποτέλεσμα του chaincode με τα δεδομένα που έχουν δοθεί, δηλαδή καλείται μια Invoke συνάρτηση. Αξίζει να σημειωθεί, πως αυτή η διαδικασία γίνεται παράλληλα και χωρίς τον συντονισμό των endorsing peers. Η έξοδος αυτής της εκτέλεσης αποτελείται από ένα σύνολο read/write και αξιοποιείται για να δημιουργηθεί το μήνυμα επικύρωσης. Ο peer υπογράφει την επικύρωση και την στέλνει πίσω στον πελάτη.

3. **Ταξινόμηση συναλλαγής**: Αρχικά ο πελάτης δέχεται όλες τις επικυρώσεις και εξετάζει, συγκρίνει και επαληθεύει εάν έχει εκπληρώσει όλες τις απαιτήσεις σύμφωνα με την πολιτική έγκρισης του chaincode. Για μια λειτουργία ανάγνωσης (read operation) ο πελάτης δεν χρειάζεται να στείλει κάποιο αίτημα ταξινόμησης. Αν το αίτημα αφορά κάποια λειτουργία εγγραφής μέσω της κλήσης μιας συνάρτησης του chaincode, τότε όλες οι επικυρώσεις ενοποιούνται σε μια συναλλαγή και αυτή υποβάλλεται στον orderer ο οποίος με την σειρά του συλλέγει, ταξινομεί και οργανώνει όλες τις συναλλαγές σε μπλοκ βάσει του αλγόριθμου συναίνεσης.

4. **Επικύρωση συναλλαγής**: Το τελευταίο στάδιο εκτέλεσης μιας συναλλαγής. Ο orderer στέλνει την ταξινομημένη πλέον σε μπλοκ συναλλαγή σε όλους τους peers στο κανάλι. Σύμφωνα με την πολιτική έγκρισης η συναλλαγή επαληθεύεται από τους peers και στην συνέχεια προστίθεται στο ledger. Αυτή η διαδικασία απαιτεί δύο προϋποθέσεις για να ικανοποιηθεί. Η πρώτη αφορά τον έλεγχο αν οι επικυρώσεις της συναλλαγής ικανοποιούν την πολιτική έγκρισης του chaincode, ενώ η δεύτερη αφορά τον έλεγχο αν το σύνολο read/write δεν αντιφάσκει με συντρέχουσες ενημερώσεις που έχουν ήδη γίνει commited. Είναι υποχρεωτικό όλοι οι peers να κάνουν commit την συναλλαγή.

Τέλος, αξίζει να σημειωθεί πως το chaincode μπορεί να κληθεί για την ανεύρεση πληροφοριών κατάστασης (state information) όπως το αποτέλεσμα κάποιου υπολογισμού ή την ενημέρωση αναφορικά με το world state. Σε αυτή την περίπτωση η κλήση του chaincode ονομάζεται query και δεν χρειάζεται να πραγματοποιηθούν τα βήματα 3 και 4 καθώς ο πελάτης απλά ζητάει την ανεύρεση πληροφοριών και όχι την ενημέρωση του δικτύου με νέες πληροφορίες [55].

3.3 Private Data Collection στο Hyperledger Fabric

3.3.1 Peer-to-Peer Gossip Πρωτόκολλο

Για την κατανόηση του τρόπου με τον οποίο λειτουργούν οι ιδιωτικές συλλογές δεδομένων είναι σημαντικό να αναλύσουμε αρχικά το πρωτόκολλο Gossip.

Αρκετά σύγχρονα συστήματα blockchain με σκοπό να διατηρήσουν την ακεραιότητα και την συνέπεια των δεδομένων τους όπως και να διαχειριστούν σωστά δίκτυα μεγάλης έκτασης χρησιμοποιούν ένα ανθεκτικό πρωτόκολλο κατανεμημένης επικοινωνίας που ονομάζεται Gossip. Το πρωτόκολλο αυτό, απαιτεί από κάθε κόμβο του δικτύου να στέλνει τα δεδομένα του επαναληπτικά σε όλους τους γειτονικούς κόμβους, μέχρι τα δεδομένα αυτά να διαδοθούν σωστά σε όλους τους κόμβους του δικτύου [56].

Ειδικότερα, το Hyperledger Fabric αξιοποιεί το **Gossip πρωτόκολλο** για πολλούς σκοπούς [57]:

- Οι peers το χρησιμοποιούν για να δημιουργήσουν και να διατηρήσουν μια τοπική οπτική (local view) των άλλων peers στο δίκτυο. Παράλληλα, το χρησιμοποιούν την πρώτη φορά που εντάσσονται στο δίκτυο ή μετά από κάποιο σφάλμα στην λειτουργία τους.

- Αξιοποιείται για την διάδοση μεταδεδομένων (metadata) όπως το ύψος του ledger και τα ενεργά chaincodes.

- Τελευταία και κύρια λειτουργία του είναι η διάδοση νέων μπλοκ σε όλους τους peers εντός ενός οργανισμού, οι οποίοι στην συνέχεια επικυρώνουν ανεξάρτητα το περιεχόμενο τους και εφαρμόζουν τα αποτελέσματα των έγκυρων συναλλαγών στο τοπικό τους αντίγραφο του ledger.

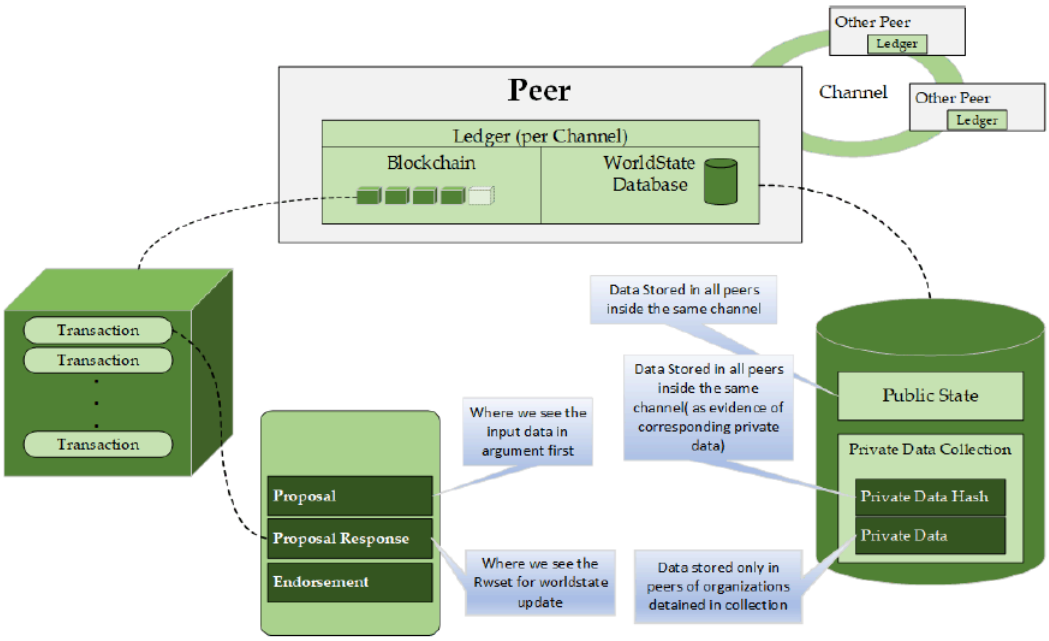
Το gossip πρωτόκολλο επομένως διαχειρίζεται την ανακάλυψη νέων peers στο κανάλι μέσω της συνεχούς αναγνώρισης των διαθέσιμων εγγεγραμμένων μελών και προοδευτικά τον εντοπισμό αυτών που έχουν διακόψει την λειτουργία τους.

Παράλληλα, συμβάλλει στον συγχρονισμό των δεδομένων στο δίκτυο καθώς κάθε peer με δεδομένα ασύγχρονα με το υπόλοιπο κανάλι, εντοπίζει τα μπλοκ που του λείπουν και συγχρονίζεται αντιγράφοντας τα σωστά δεδομένα. Τέλος επιτρέπει στους νέους peers που συνδέονται στο δίκτυο να ενημερώνονται μέσω της peer-to-peer μεταφοράς των δεδομένων του ledger [50].

3.3.2 Τι είναι το Private Data Collection

Το Hyperledger Fabric, για να βελτιώσει την διαχείριση εμπιστευτικών δεδομένων και να προσφέρει ένα επιπλέον επίπεδο ιδιωτικότητας πέραν από αυτό που παρέχουν τα κανάλια, εισήγαγε την έννοια της Ιδιωτικής Συλλογής Δεδομένων (Private Data Collection ή PDC).

Το PDC πρακτικά αποτελεί μια μέθοδο με την οποία μια υπό-ομάδα οργανισμών εντός κάποιου καναλιού μπορεί να διατηρήσει και να ανταλλάξει ευαίσθητα δεδομένα χωρίς αυτά να κοινοποιούνται σε όλους τους συμμετέχοντες του καναλιού.



Εικόνα 25: Private Data Collection στο Hyperledger Fabric [58].

Ένα **Private Data Collection** αποτελείται από δύο μέρη [50]:

1. **Τα ιδιωτικά δεδομένα**: Αποθηκεύονται σε μια ιδιωτική βάση δεδομένων στους peers εξουσιοδοτημένων οργανισμών, οι οποίοι μπορούν να τα προσπελάσουν μέσω του κατάλληλου chaincode. Η υπηρεσία ταξινόμησης δεν συμμετέχει σε αυτές τις συλλογές και δεν μπορεί να δει τα ιδιωτικά δεδομένα. Ο τρόπος με τον οποίο κοινοποιούνται είναι μέσω του gossip πρωτόκολλου που αναλύθηκε στην προηγούμενη ενότητα.

2. **Ο κατακερματισμός (hash) των δεδομένων:** Ο κατακερματισμός είναι αυτός που επικυρώνεται, ταξινομείται και γράφεται στα ledgers όλων των peers στο κανάλι. Πρακτικά, αξιοποιείται σαν απόδειξη της συναλλαγής και χρησιμοποιείται για την επικύρωση του state και μπορεί να χρησιμοποιηθεί για σκοπούς ελέγχου (audit purposes).

Στην εικόνα 25 παρουσιάζεται μια οπτικοποίηση της ιδιωτικής συλλογής δεδομένων. Ο peer ο οποίος συμμετέχει σε ένα PDC διατηρεί πέραν του Public State και ένα Private state στο οποίο διατηρούνται οι πληροφορίες της συλλογής. Επιπλέον, ο

κατακερματισμός του PDC διατηρείται σε όλους τους peers ανεξαρτήτως από το αν συμμετέχουν ή όχι στην ιδιωτική συλλογή [58].

Τέλος, αξίζει να αναφερθεί και τρόπος διαχείρισης των ιδιωτικών δεδομένων σε σύγκριση με τα δημόσια δεδομένα τόσο στον τρόπο αποθήκευσης τους όσο και κατά την διαδικασία κάποιας συναλλαγής [59].

- **Σύγκριση τρόπου αποθήκευσης Δεδομένων:** Τα ιδιωτικά δεδομένα αποθηκεύονται με διαφορετικό τρόπο απ' ότι τα δημόσια δεδομένα στο world state (παγκόσμια κατάσταση). Τα δημόσια δεδομένα αποθηκεύονται με την μορφή <κλειδί, τιμή, έκδοση> σε όλους τους peers του καναλιού.

  Εν αντιθέσει, τα ιδιωτικά δεδομένα όπως αναφέρθηκε και προηγουμένως έχουν δύο μορφές αποθήκευσης:
  1. <κλειδί, τιμή, έκδοση> για το υποσύνολο των peers που συμμετέχουν στην ιδιωτική συλλογή.
  2. <hash(κλειδί), hash(τιμή), έκδοση> σε όλους τους υπόλοιπους peers του καναλιού.

- **Σύγκριση ροής κατά τη διαδικασία συναλλαγών**: Η ροή εργασιών για συναλλαγές ιδιωτικών δεδομένων διαφέρει από αυτή των δημόσιων στο στάδιο εκτέλεσης. Το read/write σύνολο που επιστρέφεται σαν απάντηση του αιτήματος συναλλαγής είναι κατακερματισμένο (hashed) για αποφυγή αποκάλυψης των ιδιωτικών δεδομένων. Οι επικυρωτές (endorsers) διατηρούν και στέλνουν το αρχικό read/write σύνολο μέσω του peer-to-peer gossip πρωτοκόλλου στους συμμετέχοντες peers της ιδιωτικής συλλογής που χρειάζονται τα δεδομένα για την φάση επικύρωσης της συναλλαγής (validation phase). Η συναλλαγή με το hash του read/write συνόλου οργανώνεται σε μπλοκ και διανέμεται από τους orderers σε όλους τους peers του καναλιού οι οποίοι με την σειρά τους θα την επικυρώσουν. Τέλος, πριν ενημερωθεί το ledger οι peers που ανήκουν στην ιδιωτική συλλογή επιβεβαιώνουν πως το αρχικό read/write σύνολο ταιριάζει με το hash της συναλλαγής.

Μέρος ΙΙ

Πρακτικό Μέρος

Κεφάλαιο 4

Εισαγωγή Πρακτικού Μέρους

4.1 Γενικά

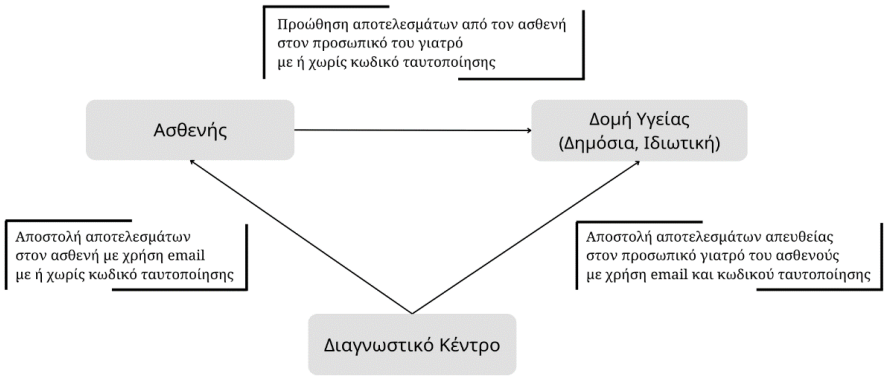
Στη σημερινή εποχή της ραγδαίας τεχνολογικής εξέλιξης, η καθημερινότητά μας μετασχηματίζεται συνεχώς και οι νέες τεχνολογίες ενσωματώνονται σε όλο και περισσότερες πτυχές της ζωής μας. Από την επικοινωνία και την εργασία έως τις αγορές και την ψυχαγωγία, οι ψηφιακές λύσεις παίζουν πλέον καθοριστικό ρόλο. Μέσα σε αυτό το πλαίσιο, η ασφάλεια των προσωπικών δεδομένων αναδεικνύεται ως μια από τις πλέον σημαντικές προκλήσεις της σύγχρονης εποχής.

Η τεχνολογία blockchain προσφέρει μια καινοτόμο προσέγγιση για την αντιμετώπιση αυτού του ζητήματος, παρέχοντας αξιόπιστους μηχανισμούς που μπορούν να διασφαλίσουν την προστασία των δεδομένων. Με τη χρήση αποκεντρωμένων και κρυπτογραφημένων αρχείων, το blockchain προσφέρει ένα νέο επίπεδο ασφάλειας και διαφάνειας, ενισχύοντας την εμπιστοσύνη των χρηστών στις ψηφιακές συναλλαγές και εφαρμογές.

4.2 Διατύπωση Προβλήματος – Problem Formulation

Ένας βασικός τομέας όπου η ασφάλεια των δεδομένων παίζει κομβικό ρόλο είναι αυτός της ιατρικής, ειδικά όσον αφορά τα ευαίσθητα προσωπικά δεδομένα του ασθενούς, όπως οι ιατρικές εξετάσεις και οι συνταγογραφήσεις. Μέχρι σήμερα, ο ασθενής καλείται τις περισσότερες φορές να αναλαμβάνει την ευθύνη για τυχόν υποκλοπές των δεδομένων του, καθώς συχνά οι εξετάσεις αποστέλλονται μέσω email, με ή χωρίς κωδικό ταυτοποίησης. Αυτή η πρακτική ενέχει κινδύνους για την ιδιωτικότητα και την ασφάλεια των ευαίσθητων πληροφοριών.

Αναλυτικότερα, η σημερινή προσέγγιση για την αποστολή αποτελεσμάτων ιατρικών εξετάσεων του ασθενούς από κάποιο διαγνωστικό κέντρο παρουσιάζεται στην εικόνα 26.



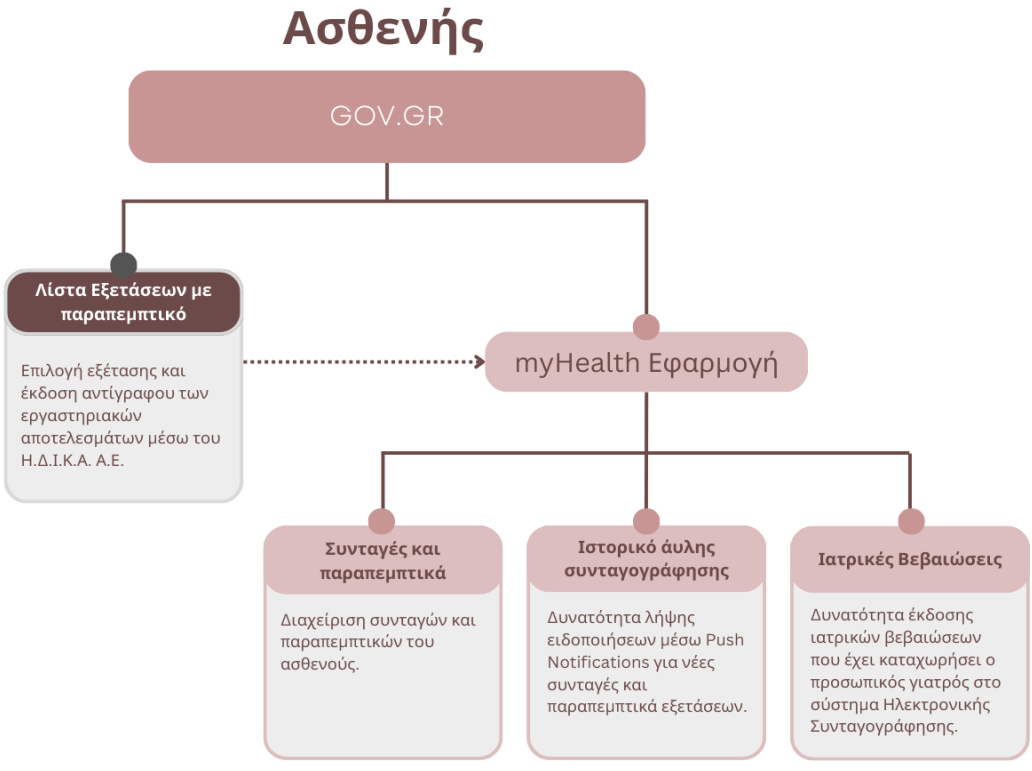
Εικόνα 26: Σημερινή προσέγγιση αποστολής ιατρικών εξετάσεων στην Ελλάδα.

Ο ασθενής, όταν χρειάζεται να πραγματοποιήσει κάποια ιατρική εξέταση, έχει δύο επιλογές: είτε να παραλάβει τα αποτελέσματα αυτοπροσώπως είτε να του αποσταλούν μέσω email. Παράλληλα, σε αρκετές περιπτώσεις η ιατρική εξέταση δεν πραγματοποιείται στην Δομή Υγείας που τον παρακολουθεί αλλά σε κάποιο Διαγνωστικό Κέντρο ή κάποια άλλη Δομή. Ο σκοπός είναι να λάβει τα αποτελέσματα ο προσωπικός γιατρός του ασθενούς. Αυτό γίνεται είτε:

- Με αποστολή των αποτελεσμάτων στο προσωπικό email του ασθενούς, ο οποίος στην συνέχεια τα παραδίδει στον προσωπικό του γιατρό.
- Με απευθείας αποστολή στο προσωπικό email του γιατρού.

Για μεγαλύτερη ασφάλεια, παρέχεται η δυνατότητα προστασίας του αρχείου με έναν κωδικό ταυτοποίησης που ορίζει ο ασθενής και ο οποίος σχεδόν σε όλες τις περιπτώσεις αξιοποιείται.

Βάσει των ΦΕΚ 5940/Β/21-11-2022 [60] και 6973 Β/11-12-23 [61] με σκοπό την διασφάλιση και προστασία των προσωπικών δεδομένων του ασθενούς και των κανονισμών 2016/679 του ευρωπαϊκού κοινοβουλίου για την προστασία των φυσικών προσώπων έναντι της επεξεργασίας των δεδομένων προσωπικού χαρακτήρα, η ηλεκτρονική χορήγηση αποτελεσμάτων διαγνωστικών εργαστηριακών εξετάσεων ασθενούς θα πραγματοποιείται μέσω του Gov.gr.



Εικόνα 27: Πύλη εισόδου Gov.gr για ταυτοποίηση ασθενούς και πρόσβαση σε υπηρεσίες Υγείας.

Αναλυτικότερα βάσει της ΚΥΑ 5940 για την πρόσβαση στην ηλεκτρονική εφαρμογή ο ενδιαφερόμενος ασθενής θα αυθεντικοποιείται μέσω της Ενιαίας Ψηφιακής Πύλης της Δημόσιας Διοίκησης (gov.gr- ΕΨΠ) με τη χρήση κωδικών - διαπιστευτηρίων του

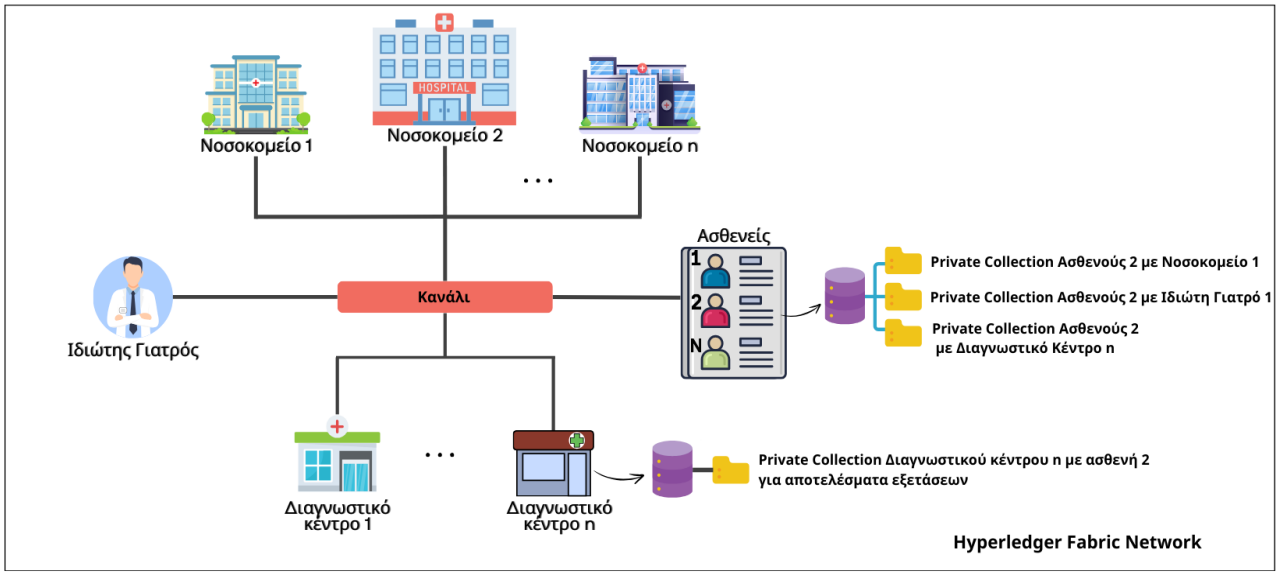
άρθρου 24 του ν. 4727/2020 (Α' 184) και για την ολοκλήρωση της αυθεντικοποίησης θα απαιτείται η καταχώρηση κωδικού μιας χρήσης (One Time Password - OTP), ο οποίος αποστέλλεται στον αριθμό κινητού τηλεφώνου του ασθενούς.

Μέχρι σήμερα ο ασθενής με τον ίδιο μηχανισμό ταυτοποίησης μέσω της ψηφιακής πύλης του gov.gr μπορεί να έχει πρόσβαση σε πολυάριθμες υπηρεσίες υγείας όπως αυτές που περιλαμβάνονται στην myHealth εφαρμογή. Σε αυτές τις υπηρεσίες θα προστεθεί και η λίστα εξετάσεων του ασθενούς όπου θα μπορεί να βλέπει αναλυτικά τα αποτελέσματα των εξετάσεων του. Αυτή η τεχνολογική εξέλιξη θα προσφέρει ένα υψηλότερο επίπεδο προστασίας και αξιοπιστίας στην διαχείριση των ιατρικών πληροφοριών.

Παρ' όλα αυτά μια τέτοια κεντροποιημένη προσέγγιση συστήματος μπορεί να θέσει ακόμα περισσότερα ζητήματα ασφάλειας.

4.3 Προτεινόμενη Λύση

Η προσέγγιση που προτείνεται στην παρούσα διπλωματική είναι η αξιοποίηση της τεχνολογίας blockchain και ειδικότερα του Hyperledger Fabric με σκοπό την δημιουργία ενός αποκεντρωμένου δικτύου όπου ο ασθενής θα έχει τον πρώτο λόγο στα προσωπικά του δεδομένα.



Εικόνα 28: Οπτικοποίηση Δικτύου Hyperledger Fabric.

Συμμετέχοντες αυτού του δικτύου θα είναι κάθε είδους Δομή Υγείας που μπορεί να αφορά σε Δημόσια ή Ιδιωτικά Νοσοκομεία, Κέντρα Υγείας, Ιδιώτες Γιατρούς κ.ο.κ. Μέσω της αξιοποίησης ιδιωτικών συλλογών δεδομένων (Private Data Collections) ο κάθε ασθενής θα έχει την δυνατότητα να γνωρίζει ακριβώς ποιοι έχουν πρόσβαση στα προσωπικά του δεδομένα όπως και να έχει έλεγχο αυτών, χωρίς να κοινοποιούνται σε μη εξουσιοδοτημένες οντότητες. Παράλληλα, μέσω του ασφαλούς διαύλου που προσφέρει η τεχνολογία του blockchain, θα επιτρέπεται η ασφαλής λήψη και μεταφορά των εξετάσεων του ασθενούς, είτε αυτές πραγματοποιούνται εξωτερικά σε

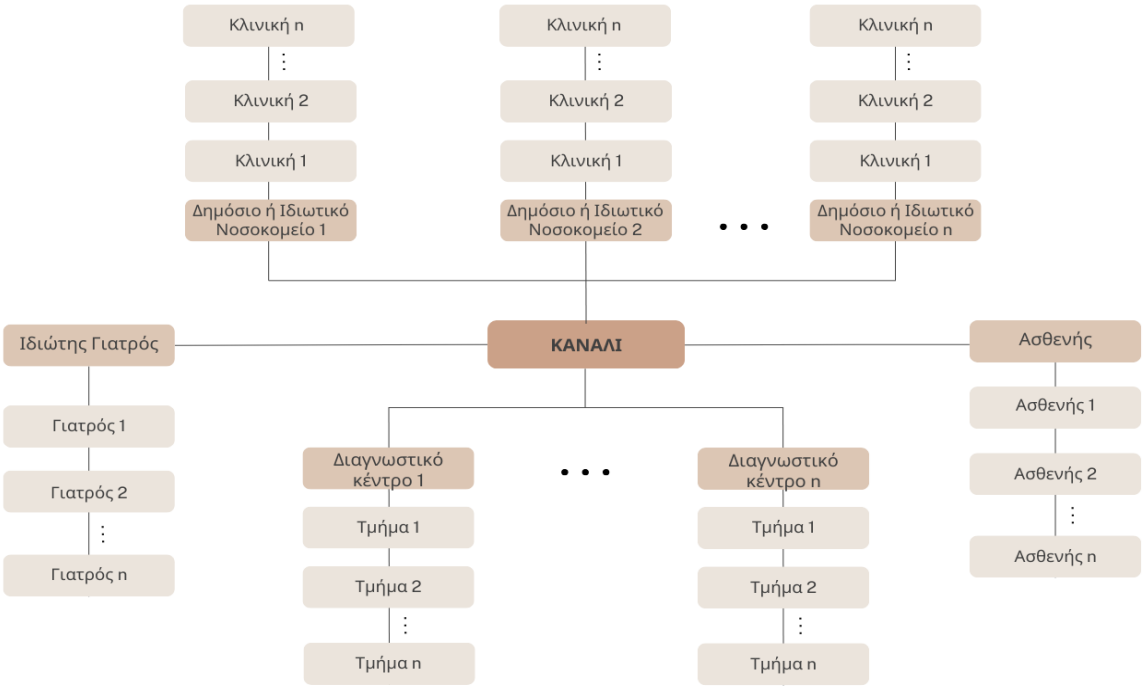
κάποιο Διαγνωστικό Κέντρο είτε από την ίδια τη Δομή Υγείας στην οποία ανήκει ο προσωπικός του γιατρός. Το πλεονέκτημα της δημιουργίας ενός τέτοιου ενοποιημένου δικτύου είναι ότι ο ασθενής θα μπορεί να επικοινωνεί άμεσα με τις Δομές Υγείας που συνεργάζεται, χωρίς μεσάζοντες αξιοποιώντας το ασφαλές περιβάλλον που προσφέρει η τεχνολογία του blockchain. Επιπλέον, το προτεινόμενο δίκτυο χαρακτηρίζεται από επεκτασιμότητα, χωρίς περιορισμούς τόσο σε ότι αφορά στους συμμετέχοντες οργανισμούς όσο και στην δημιουργία νέων ιδιωτικών συλλογών του ασθενούς με διαφορετικές Δομές Υγείας που μπορεί να επιλέξει στο μέλλον. Στην εικόνα 28 δίνεται μια οπτικοποίηση του Δικτύου στο Hyperledger Fabric όπως και μια συνοπτική παρουσίαση των ιδιωτικών συλλογών δεδομένων.

Για παράδειγμα, ο ασθενής 2 διατηρεί τρεις ιδιωτικές συλλογές:

- Η $1^{η}$ είναι μεταξύ αυτού και του Νοσοκομείου 1.
- Η $2^{η}$ είναι μεταξύ αυτού και ενός Ιδιώτη Γιατρού.
- Η $3^{η}$ είναι μεταξύ αυτού και του Διαγνωστικού Κέντρου n.

Παρατηρούμε ότι και το Διαγνωστικό Κέντρο n διατηρεί την αντίστοιχη ιδιωτική συλλογή, η οποία αφορά τα αποτελέσματα εξετάσεων που έχει πραγματοποιήσει εκεί μέσω κάποιου παραπεμπτικού. Ο ασθενής έχει την δυνατότητα να επιλέξει την κοινοποίηση των αποτελεσμάτων που βρίσκονται σε αυτή την ιδιωτική συλλογή είτε σε αυτή που μοιράζεται με το Νοσοκομείο 1 ή ακόμα και σε αυτή που μοιράζεται με τον Ιδιώτη Γιατρό. Σε κάθε περίπτωση όμως, τα δεδομένα αυτά δεν κοινοποιούνται παρά μόνο με την εξουσιοδότηση του ίδιου και κανένας μη εγκεκριμένος χρήστης αυτών των συλλογών δεν μπορεί να τα δει.

Αναλυτικότερα στην εικόνα 29 παρουσιάζεται η εσωτερική δομή των οντοτήτων που συμμετέχουν στο δίκτυο.



Εικόνα 29: Εσωτερική Οργάνωση Οντοτήτων του Δικτύου.

Με σκοπό την διασφάλιση και προστασία των προσωπικών δεδομένων του ασθενούς οι ιδιωτικές συλλογές δεδομένων που δημιουργούνται αφορούν συγκεκριμένες οντότητες εντός του ίδιου οργανισμού. Παραδείγματος χάριν, ο Ασθενής 2 ο οποίος διατηρεί μια ιδιωτική συλλογή με το Νοσοκομείο 1 δεν διατηρεί με ολόκληρο το Νοσοκομείο 1 αλλά με κάποια ή περισσότερες από τις κλινικές του. Συνεπώς αν η ιδιωτική συλλογή αφορά τον ίδιο και την Κλινική 1 του Νοσοκομείου 1, χρήστες της κλινικής 2 δεν μπορούν να διαβάσουν τα δεδομένα εντός της συλλογής ακόμα και αν ανήκουν στο Νοσοκομείο 1.

Επομένως, μέσω της αξιοποίησης του Hyperledger Fabric δημιουργείται ένα ενοποιημένο δίκτυο που παρέχει ασφάλεια, αποκέντρωση και επεκτασιμότητα.

4.4 Εργαλεία και τεχνολογίες για την υλοποίηση της προτεινόμενης λύσης

Η εργασία πραγματοποιήθηκε σε λειτουργικό σύστημα Mac OS Version 12.7.5 και βασίστηκε στην νεότερη έκδοση του Hyperledger Fabric [50].

Για την λειτουργία του δικτύου και την συγγραφή του κώδικα χρειάστηκαν τα εξής :

1. Docker
2. Απαραίτητα εργαλεία κώδικα Git, cURL, Go language
3. Hyperledger Fabric Repositories και Test Network
4. Visual Studio Code

4.4.1 Docker

Το Docker αποτελεί ένα εργαλείο που επιτρέπει την ανάπτυξη και εκτέλεση εφαρμογών μέσα σε απομονωμένα περιβάλλοντα που ονομάζονται containers. Για μεγαλύτερη διευκόλυνση επιλέχθηκε το Docker Desktop το οποίο εμπεριέχει το docker engine και docker compose που απαιτούνται για την δημιουργία του δικτύου. Παράλληλα το Docker Desktop προσφέρει ένα περιβάλλον ελέγχου των containers που έχουμε δημιουργήσει [62].

Πρακτικά, ένα container έχει αντίστοιχη λογική με αυτή των εικονικών μηχανών (VMs). Αξιοποιώντας τις δυνατότητες απομόνωσης και εικονικοποίησης (virtualization) που προσφέρει το kernel των Linux συστημάτων το Docker επιτρέπει την δημιουργία ανεξάρτητων εφαρμογών εντός του ίδιου kernel του host. Σε αντίθεση όμως με τις εικονικές μηχανές τα containers απαιτούν λιγότερους υπολογιστικούς πόρους καθώς χρειάζονται μόνο τις διεργασίες και τις εξαρτήσεις του λειτουργικού συστήματος που είναι απαραίτητες για την εκτέλεση του κώδικα όπως και επιτρέπουν την ευκολότερη και ταχύτερη ανάπτυξη. Οι εφαρμογές εντός των containers αρκεί να δημιουργηθούν μια φορά και μπορούν να τρέξουν οπουδήποτε [63].

Η εγκατάσταση του Docker Desktop έγινε από την επίσημη ιστοσελίδα του Docker [64] βάσει των απαιτήσεων του documentation του Hyperledger Fabric [50].

Κάθε node του δικτύου μας τρέχει μέσα σε ένα Docker container επιτρέποντας έτσι την ανεξάρτητη εκτέλεσή του, ενώ παράλληλα προσφέρει τις κατάλληλες μεθόδους

για την επικοινωνία του με τα υπόλοιπα nodes, η οποία είναι κομβική για την σωστή λειτουργία του δικτύου.

4.4.2 Απαραίτητα εργαλεία κώδικα Git, cURL, Go language

Το επόμενο βήμα είναι η εγκατάσταση των απαραίτητων εργαλείων που είναι προ-απαιτούμενα για την εκτέλεση Docker-based Fabric δικτύων. Καθώς χρησιμοποιήθηκε λειτουργικό MacOS για μεγαλύτερη διευκόλυνση η εγκατάσταση των Git, cURL και Go έγινε μέσω Homebrew και του command prompt. Τα βήματα που ακολουθήθηκαν όπως και οι εντολές του homebrew παρουσιάζονται στο documentation του Hyperledger Fabric [65].

Ειδικότερα:

- Το Git αποτελεί ένα δωρεάν και ανοιχτού κώδικα κατανεμημένο σύστημα ελέγχου εκδόσεων (version control system) που χρησιμοποιείται κυρίως για την παρακολούθηση των αλλαγών στον πηγαίο κώδικα κατά τη διάρκεια της ανάπτυξης λογισμικού. Είναι ένα βασικό εργαλείο για την ανάπτυξη λογισμικού, επιτρέποντας συνεργασία, παρακολούθηση αλλαγών και διαχείριση της ανάπτυξης κώδικα με τρόπο αποδοτικό και ασφαλή.

- Το cURL είναι ένα εργαλείο γραμμής εντολών που χρησιμοποιείται για τη μεταφορά δεδομένων από ή προς έναν διακομιστή, χρησιμοποιώντας διάφορα πρωτόκολλα. Είναι ένα ευέλικτο και ισχυρό εργαλείο που επιτρέπει στους χρήστες να εκτελούν αιτήματα HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP, LDAPS, FILE και άλλες μορφές URL.

- Η γλώσσα Go αποτελεί μια από τις βασικές γλώσσες που αξιοποιούνται στο Hyperledger Fabric για την ανάπτυξη έξυπνων συμβολαίων. Επιλέχθηκε καθώς διαθέτει πολυάριθμα εργαλεία και βιβλιοθήκες, επιτρέποντας έτσι τον εύκολο και αποδοτικό τρόπο ανάπτυξης έξυπνων συμβολαίων.

4.4.3 Hyperledger Fabric Repositories και Test Network

Στο σημείο αυτό εγκαθιστούμε όλες τις απαραίτητες βιβλιοθήκες και εργαλεία που χρειάζεται το Hyperledger Fabric [66]. Σημαντική είναι η αναφορά των CLI (client) εργαλείων που αξιοποιούμε καθώς παίζουν κομβικό ρόλο σε αρκετά σημεία του κώδικα τόσο κατά την διαδικασία εγκατάστασης του δικτύου μας όσο και για την επικοινωνία μας με αυτό. Τα βασικότερα είναι:

1. **Configtxgen**: Είναι ένα εργαλείο που χρησιμοποιείται για την δημιουργία των αρχικών διαμορφώσεων των καναλιών και των πολιτικών τους. Δημιουργεί τα βασικά block (genesis block) που απαιτούνται για την αρχική διαμόρφωση των καναλιών καθώς και τα αρχεία που περιγράφουν αυτή τη διαμόρφωση (channel configuration).

2. **Configtxlator**: Χρησιμοποιείται για την μετάφραση και τροποποίηση αρχείων διαμόρφωσης του Fabric. Επιτρέπει την μετατροπή αυτών των αρχείων σε διάφορες μορφές (όπως JSON) διευκολύνοντας τις αλλαγές και τις ενημερώσεις στις διαμορφώσεις του καναλιού του δικτύου.

3. **Cryptogen**: Χρησιμοποιείται για την δημιουργία των κρυπτογραφικών υλικών που απαιτούνται για το δίκτυο Fabric, δηλαδή τα απαραίτητα κλειδιά όπως και πιστοποιητικά των οντοτήτων (peers, users, orderers, Orgs).

4. **Orderer**: Αποτελεί το εκτελέσιμο πρόγραμμα των orderer κόμβων. Αξιοποιείται για την εκκίνηση και εκτέλεση του ordering service που είναι κρίσιμο για την σωστή λειτουργία του δικτύου.

5. **Osnadmin**: Χρησιμοποιείται για την διαχείριση των orderer κόμβων. Επιτρέπει την εκτέλεση διοικητικών λειτουργιών όπως η προσθήκη και η αφαίρεση orderer κόμβων.

6. **Peer**: Αποτελεί το κύριο εργαλείο CLI για την αλληλεπίδραση με τους peer κόμβους. Μέσω των εντολών που διαθέτει, μας επιτρέπει την εκτέλεση πολυάριθμων λειτουργιών στο δίκτυο όπως την προσθήκη peers σε κάποιο κανάλι και την εκτέλεση κάποιου chaincode.

Παράλληλα, εγκαθιστούμε και τα fabric samples [67] που παρέχονται από το Hyperledger Fabric και τα οποία αξιοποιήθηκαν για την δημιουργία του δικτύου του Κεφαλαίου 5.

4.4.4 Visual Studio Code

Τελευταίο εργαλείο που εγκαταστάθηκε είναι μια εφαρμογή συγγραφής κώδικα και ειδικότερα το Visual Studio Code για την δημιουργία των κατάλληλων scripts όπως και του chaincode που εγκαταστάθηκε στο δίκτυο Fabric. Η εκτέλεση τους πραγματοποιήθηκε μέσω του command prompt.

4.5 Παράδειγμα χρήσης της προτεινόμενης προσέγγισης

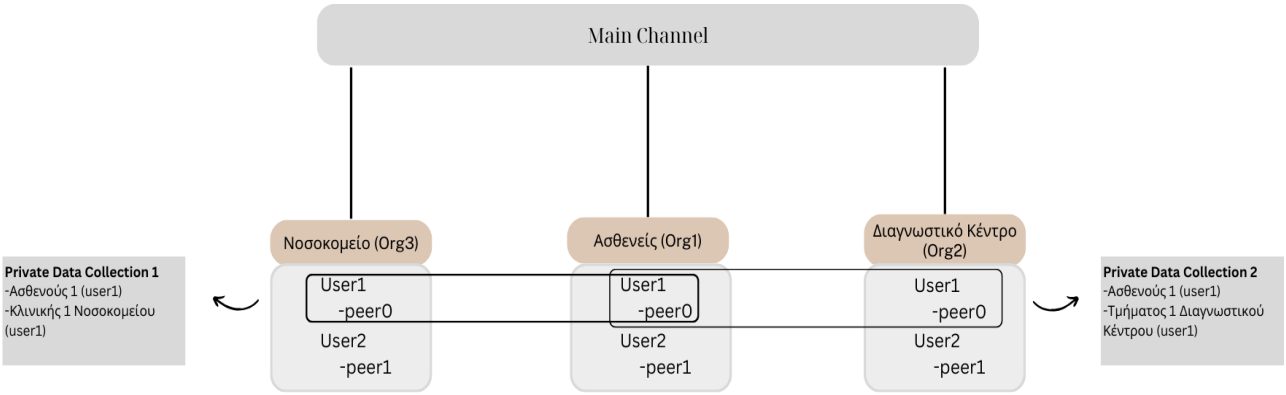
Για τους σκοπούς της παρούσας διπλωματικής το δίκτυο που έχουμε δημιουργήσει αποτελείται από τρεις οργανισμού Org1, Org2 και Org3 με κάθε οργανισμό να διαθέτει:

- 2 users (user1, user2)
- 2 peers (peer0, peer1)

Ο Org1 αντιπροσωπεύει την οντότητα του Ασθενή, ο Org2 του Διαγνωστικού Κέντρου και ο Org3 του Νοσοκομείου. Διαθέτουμε 2 χρήστες και 2 peers ώστε να οπτικοποιήσουμε και την εσωτερική δομή των παραπάνω οντοτήτων. Όπως

αναφέρθηκε και στην ενότητα 4.3 η οντότητα του Ασθενή θα περιέχει όλους τους ασθενείς που θα συμμετέχουν στο Δίκτυο. Στο σενάριο εδώ, έχουμε δημιουργήσει 2 ασθενείς τον Ασθενή 1 (user1) και τον Ασθενή 2 (user2). Αντίστοιχα το Διαγνωστικό Κέντρο αποτελείται από τμήματα και το Νοσοκομείο από κλινικές. Σε ένα πραγματικό σενάριο, ένα διαφορετικό τμήμα ή μια διαφορετική κλινική ενός νοσοκομείου δεν πρέπει να έχει πρόσβαση στα προσωπικά δεδομένα του ασθενούς. Παραδείγματος χάριν, εάν ο ασθενής πραγματοποιήσει εξετάσεις στην ρευματολογική κλινική του νοσοκομείου, η χειρουργική κλινική δεν πρέπει να έχει πρόσβαση σε αυτά τα στοιχεία παρά μόνο με την εξουσιοδότηση του ασθενούς. Συνεπώς η αξιοποίηση δύο διαφορετικών users παρέχει την κατάλληλη βάση ώστε να λειτουργήσει σωστά το δίκτυο μας. Σκοπός είναι οι διαφορετικοί χρήστες να μην μπορούν να δουν τα δεδομένα ιδιωτικών συλλογών εφόσον δεν ανήκουν στην ιδιωτική συλλογή ακόμα και αν συμμετέχουν στον ίδιο οργανισμό.

Σημαντικό είναι να διαθέτουμε εκτός από δύο users και τουλάχιστον δύο peers ώστε στην περίπτωση που έχουμε ένα παράλληλο αίτημα και από τους δύο users του ίδιου οργανισμού να υπάρχει διαθέσιμος peer που να μπορεί να εκτελέσει το αίτημα αυτό.



Εικόνα 30: Εσωτερική Δομή Δικτύου που υλοποιήθηκε στο Hyperledger Fabric.

Στην εικόνα 30 παρουσιάζεται το δίκτυο το οποίο έχει υλοποιηθεί. Η κλινική 1 του Νοσοκομείου μοιράζεται μια ιδιωτική συλλογή με τον Ασθενή 1. Παράλληλα ο Ασθενής 1 διατηρεί και μια ιδιωτική συλλογή με το Τμήμα 1 του Διαγνωστικού Κέντρου.

Κεφάλαιο 5

Σχεδίαση και Υλοποίηση Σεναρίων

Στο κεφάλαιο αυτό θα αναλυθεί ο κώδικας για την δημιουργία του δικτύου όπως και τα αποτελέσματα την προσέγγισης που παρουσιάστηκε στην ενότητα 4.5. Το project βασίστηκε στο test-network [67] [68] που παραχωρεί το Hyperledger Fabric για πειραματικούς σκοπούς, κάνοντας τις απαραίτητες τροποποιήσεις και προσθήκες για τις ανάγκες υλοποίησης της προτεινόμενης λύσης της παρούσας διπλωματικής.

5.1 Δημιουργία και Εγκατάσταση Δικτύου

Η παρούσα ενότητα αφορά την δημιουργία όπως και την εγκατάσταση των επιμέρους μερών του δικτύου με ανάλυση των μεθόδων που ακολουθήθηκαν. Αποτελείται από δύο μέρη, αρχικά παρουσιάζονται οι συναρτήσεις και οι εντολές οι οποίες αξιοποιήθηκαν για την υλοποίηση και στην συνέχεια παρουσιάζεται η επιτυχής εκτέλεση τους μέσω του command prompt το οποίο προσομοιώνει το client που αξιοποιεί ο εκάστοτε χρήστης (admin δικτύου, Ασθενής, Νοσοκομείο κ.ο.κ).

5.1.1 Δημιουργία Οργανισμών και Peers

Το δίκτυο μας αποτελείται από τρεις οργανισμούς: Org1, Org2 και Org3 καθένας από τους οποίους διαθέτει δύο peers και δύο users. Παράλληλα, διαθέτουμε και έναν orderer. Για την δημιουργία των κρυπτογραφικών υλικών χρησιμοποιούμε το cryptogen ως εξής:

```
cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output="organizations"
```

Εικόνα 31: Δημιουργία κρυπτογραφικών υλικών για τον Org1.

Αυτή η εντολή δημιουργεί τα απαραίτητα κρυπτογραφικά υλικά για όλες τις οντότητες εντός του Org1 βάσει του yaml αρχείου crypto-config.yaml το οποίο έχουμε επεξεργαστεί κατάλληλα ως εξής:

```
PeerOrgs:
  # ---------------------------------------------------------------------------
  # Org1
  # ---------------------------------------------------------------------------
  - Name: Org1
    Domain: org1.example.com
    EnableNodeOUs: true
    Specs:
      - Hostname: peer0
        SANS:
         - localhost
      - Hostname: peer1
        SANS:
        - localhost
    Users:
      Count: 2
```

Εικόνα 32: Config.yaml αρχείο του Org1.

Ορίζουμε την ύπαρξη δύο peers (peer0 και peer1) όπως και δύο users οι οποίοι θα έχουν τα δικά τους certificates που θα αξιοποιήσουμε στη συνέχεια.

Σημαντικό επίσης για τον συντονισμό των Private Collections είναι η σωστή επιλογή ports στις οποίες θα ακούν οι peers τόσο για το chaincode όσο και για το gossip πρωτόκολλο. Αυτό ορίζεται από δύο σημεία στον κώδικα. Το πρώτο αφορά τον ορισμό των κατάλληλων ports στο container του docker που θα δημιουργηθεί όπως φαίνεται παρακάτω για τον peer1 του Org3:

```
peer1.org3.example.com:
  container_name: peer1.org3.example.com
  image: hyperledger/fabric-peer:latest
  labels:
    service: hyperledger-fabric
  environment:
    - FABRIC_CFG_PATH=/etc/hyperledger/peercfg
    - FABRIC_LOGGING_SPEC=INFO
    #- FABRIC_LOGGING_SPEC=DEBUG
    - CORE_PEER_TLS_ENABLED=true
    - CORE_PEER_PROFILE_ENABLED=false
    - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
    - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
    - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
    # Peer specific variables
    - CORE_PEER_ID=peer1.org3.example.com
    - CORE_PEER_ADDRESS=peer1.org3.example.com:12051
    - CORE_PEER_LISTENADDRESS=0.0.0.0:12051
    - CORE_PEER_CHAINCODEADDRESS=peer1.org3.example.com:12052
    - CORE_PEER_CHAINCODELISTENADDRESS=0.0.0.0:12052
    - CORE_PEER_GOSSIP_BOOTSTRAP=peer1.org3.example.com:12051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=peer1.org3.example.com:12051
    - CORE_PEER_LOCALMSPID=Org3MSP
    - CORE_PEER_MSPCONFIGPATH=/etc/hyperledger/fabric/msp
    - CORE_OPERATIONS_LISTENADDRESS=peer1.org3.example.com:9449
    - CORE_METRICS_PROVIDER=prometheus
    - CHAINCODE_AS_A_SERVICE_BUILDER_CONFIG={"peername":"peer1org3"}
    - CORE_CHAINCODE_EXECUTETIMEOUT=300s
  volumes:
    - ../organizations/peerOrganizations/org3.example.com/peers/peer1.org3.example.com:/etc/hyperledger/fabric
    - peer1.org3.example.com:/var/hyperledger/production
  working_dir: /root
  command: peer node start
  ports:
    - 12051:12051
    - 9449:9449
  networks:
    - test
```

Εικόνα 33: Docker container κανόνες για τον peer1 του Org3.

Για το container που ονομάζουμε peer1.org3.example.com έχουμε ορίσει ότι θα επικοινωνεί μέσω του port 12051 για γενικούς σκοπούς, ενώ για αναφορές σε κάποιο chaincode στο port 12052. Τέλος για ειδικούς σκοπούς αξιοποιείται το port 9449. Αντίστοιχα ορίσαμε τα ports και για τα υπόλοιπα containers ως εξής:

- Για τον **Org1**:
    1. **Peer0**: port 7051 και 7052
    2. **Peer1**: port 8051 και 8052
- Για τον **Org2**:
    1. **Peer0**: port 9051 και 9052
    2. **Peer1**: port 10051 και 10052
- Για τον **Org3**:
    1. **Peer0**: port 11051 και 11052

2. **Peer1**: port 12051 και 12052

- Για τον **Orderer**:
  1. port 7050, 7053 και 9443

Αυτά τα ports όπως και το σύνολο των κανόνων εκάστοτε οργανισμού όπως η ονομασία του MSP, το certificate του κάθε peer κ.ο.κ, πρέπει να ορίζονται σε ένα Connection Profile (CCP) το οποίο το δημιουργούμε με δύο μορφές (α) json και (β) yaml. Αυτό το επιτυγχάνουμε ως εξής:

```
ORG=1
P0PORT=7051
P1PORT=8051
CAPORT=7054
PEERPEM=organizations/peerOrganizations/org1.example.com/tlsca/tlsca.org1.example.com-cert.pem
CAPEM=organizations/peerOrganizations/org1.example.com/ca/ca.org1.example.com-cert.pem

echo "$(json_ccp $ORG $P0PORT $CAPORT $PEERPEM $CAPEM $P1PORT)" > organizations/peerOrganizations/org1.example.com/connection-org1.json
echo "$(yaml_ccp $ORG $P0PORT $CAPORT $PEERPEM $CAPEM $P1PORT)" > organizations/peerOrganizations/org1.example.com/connection-org1.yaml

ORG=2
P0PORT=9051
P1PORT=10051
CAPORT=9054
PEERPEM=organizations/peerOrganizations/org2.example.com/tlsca/tlsca.org2.example.com-cert.pem
CAPEM=organizations/peerOrganizations/org2.example.com/ca/ca.org2.example.com-cert.pem

echo "$(json_ccp $ORG $P0PORT $CAPORT $PEERPEM $CAPEM $P1PORT)" > organizations/peerOrganizations/org2.example.com/connection-org2.json
echo "$(yaml_ccp $ORG $P0PORT $CAPORT $PEERPEM $CAPEM $P1PORT)" > organizations/peerOrganizations/org2.example.com/connection-org2.yaml

ORG=3
P0PORT=11051
P1PORT=12051
CAPORT=11054
PEERPEM=organizations/peerOrganizations/org3.example.com/tlsca/tlsca.org3.example.com-cert.pem
CAPEM=organizations/peerOrganizations/org3.example.com/ca/ca.org3.example.com-cert.pem

echo "$(json_ccp $ORG $P0PORT $CAPORT $PEERPEM $CAPEM $P1PORT)" > organizations/peerOrganizations/org3.example.com/connection-org3.json
echo "$(yaml_ccp $ORG $P0PORT $CAPORT $PEERPEM $CAPEM $P1PORT)" > organizations/peerOrganizations/org3.example.com/connection-org3.yaml
echo "Generating peer information..."
```

Εικόνα 34: Δημιουργία connection αρχείων.

Φτιάχνουμε το struct που θα έχουν τα json και yaml αρχεία και μεταφέρουμε τα απαραίτητα στοιχεία. Στην συνέχεια το αποθηκεύουμε στο κατάλληλο directory.

Μέσω του command prompt καλούμε την συνάρτηση up η οποία θα αξιοποιήσει τους παραπάνω κανόνες για να δημιουργήσει τα μέλη του δικτύου με τα κρυπτογραφικά τους υλικά όπως και τα απαραίτητα Connection profiles.

```
nyra@gsf test-network % ./network.sh up
Using docker and docker-compose
Starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb' with crypto from 'cryptogen'
LOCAL_VERSION=v2.5.4
DOCKER_IMAGE_VERSION=v2.5.4
/Users/nyra/MyNetwork/test-network/../bin/cryptogen
Generating certificates using cryptogen tool
Creating Org1 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org1.yaml --output=organizations
org1.example.com
+ res=0
Creating Org2 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org2.yaml --output=organizations
org2.example.com
+ res=0
Creating Org3 Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-org3.yaml --output=organizations
org3.example.com
+ res=0
Creating Orderer Org Identities
+ cryptogen generate --config=./organizations/cryptogen/crypto-config-orderer.yaml --output=organizations
+ res=0
Generating CCP files for Org1 and Org2 and Org3
Generating peer information...
```

Εικόνα 35: Δημιουργία πιστοποιητικών για τα μέλη του δικτύου μας.

Στην συνέχεια η συνάρτηση δημιουργεί και ξεκινάει τα containers με τους κανόνες που έχουμε ορίσει:

```
[+] Running 16/16
 ✔ Network fabric_test                          Created                                                      0.1s
 ✔ Volume "compose_peer0.org3.example.com"      Created                                                      0.0s
 ✔ Volume "compose_peer1.org3.example.com"      Created                                                      0.0s
 ✔ Volume "compose_orderer.example.com"         Created                                                      0.0s
 ✔ Volume "compose_peer0.org1.example.com"      Created                                                      0.0s
 ✔ Volume "compose_peer1.org1.example.com"      Created                                                      0.0s
 ✔ Volume "compose_peer0.org2.example.com"      Created                                                      0.0s
 ✔ Volume "compose_peer1.org2.example.com"      Created                                                      0.0s
 ✔ Container orderer.example.com                Started                                                      3.4s
 ✔ Container peer1.org2.example.com             Started                                                      4.0s
 ✔ Container peer0.org3.example.com             Started                                                      4.2s
 ✔ Container peer1.org1.example.com             Started                                                      3.7s
 ✔ Container peer0.org2.example.com             Started                                                      4.4s
 ✔ Container peer1.org3.example.com             Started                                                      4.1s
 ✔ Container peer0.org1.example.com             Started                                                      3.8s
 ✔ Container cli                                Started                                                      5.4s
CONTAINER ID   IMAGE                               COMMAND             CREATED         STATUS                  PORTS
   NAMES
050270d5e7f8   hyperledger/fabric-tools:latest     "/bin/bash"         6 seconds ago   Up Less than a second
   cli
9d0cb3b56770   hyperledger/fabric-peer:latest      "peer node start"   6 seconds ago   Up 1 second             0.0.0.0:9447->9447/tcp, 7051/tcp, 0.0.0.0:10051->10051/tcp
   peer1.org2.example.com
920010c8291d   hyperledger/fabric-orderer:latest   "orderer"           6 seconds ago   Up 2 seconds            0.0.0.0:7050->7050/tcp, 0.0.0.0:7053->7053/tcp, 0.0.0.0:9443->9443/tcp
   orderer.example.com
9777d3ce9c45   hyperledger/fabric-peer:latest      "peer node start"   6 seconds ago   Up 1 second             0.0.0.0:9051->9051/tcp, 7051/tcp, 0.0.0.0:9446->9446/tcp
   peer0.org2.example.com
4b6ec86b30ba   hyperledger/fabric-peer:latest      "peer node start"   6 seconds ago   Up 1 second             0.0.0.0:9448->9448/tcp, 7051/tcp, 0.0.0.0:11051->11051/tcp
   peer0.org3.example.com
cc341c036644   hyperledger/fabric-peer:latest      "peer node start"   6 seconds ago   Up 2 seconds            0.0.0.0:8051->8051/tcp, 7051/tcp, 0.0.0.0:9445->9445/tcp
   peer1.org1.example.com
51765bf6be66   hyperledger/fabric-peer:latest      "peer node start"   6 seconds ago   Up 2 seconds            0.0.0.0:7051->7051/tcp, 0.0.0.0:9444->9444/tcp
   peer0.org1.example.com
c00870e57430   hyperledger/fabric-peer:latest      "peer node start"   6 seconds ago   Up 1 second             0.0.0.0:9449->9449/tcp, 7051/tcp, 0.0.0.0:12051->12051/tcp
   peer1.org3.example.com
```

Εικόνα 36: Δημιουργία και εκκίνηση των συμμετεχόντων του δικτύου.

Βλέπουμε πως τα containers με τα ports που έχουμε ορίσει είναι ενεργά και τρέχουν.

5.1.2 Δημιουργία καναλιού και προσθήκη συμμετεχόντων

Το επόμενο βήμα είναι η δημιουργία του καναλιού και η προσθήκη των τριών οργανισμών με τον orderer. Για τον σκοπό αυτό δημιουργήσαμε το script createMainChannel.sh το οποίο καλείται μέσω της συνάρτησης createMainChannel που έχει προστεθεί στο network.sh script.

```
function createMainChannel() {
  # now run the script that creates a channel. This script uses configtxgen once
  # to create the channel creation transaction and the anchor peer updates.
  if ! $CONTAINER_CLI info > /dev/null 2>&1 ; then
    fatalln "$CONTAINER_CLI network is required to be running to create a channel"
  fi

  # check if all containers are present
  CONTAINERS=($($CONTAINER_CLI ps | grep hyperledger/ | awk '{print $2}'))
  len=$(echo ${#CONTAINERS[@]})
  scripts/createMainChannel.sh
}
```

Εικόνα 37: CreateMainChannel Function στο κεντρικό μας script.

Η createMainChannel αφού πρώτα ελέγξει αν υπάρχουν τα κατάλληλα containers καλεί το αντίστοιχο script για την δημιουργία και προσθήκη των συμμετεχόντων. Για την δημιουργία του καναλιού αξιοποιούμε το εργαλείο Configtxgen [69] ώστε να δημιουργηθεί το Genesis Block, βάσει των πολιτικών καναλιού που έχουμε ορίσει

στο configtx.yaml αρχείο μας. Παράλληλα, αξιοποιούμε μια κατάλληλα τροποποιημένη συνάρτηση που ονομάζεται SetGlobals ώστε να εξάγουμε τα απαραίτητα στοιχεία του κάθε peer για να μπορέσει ο ίδιος να κάνει join στο κανάλι μετά την δημιουργία του. Η συνάρτηση που αναλαμβάνει να τρέξει όλα αυτά τα στάδια με τις κατάλληλες κλήσεις άλλων συναρτήσεων παρουσιάζεται στην εικόνα 38.

```
createAndJoinChannel() {
  CHANNEL_NAME=$1

  if [ "$CHANNEL_NAME" == "common" ]; then
    BLOCKFILE="./channel-artifacts/${CHANNEL_NAME}.block"
    infoln "Generating channel genesis block '${CHANNEL_NAME}.block'"
    FABRIC_CFG_PATH=${PWD}/configtx/
    createChannelGenesisBlock

    # Create channel
    infoln "Creating channel ${CHANNEL_NAME}"
    createChannel
    successln "Channel '$CHANNEL_NAME' created"

    # Join peers to the channel
    infoln "Joining org1 peer0 to the channel..."
    joinChannel 1 "peer0"
    infoln "Joining org1 peer1 to the channel..."
    joinChannel 1 "peer1"
    infoln "Joining org2 peer0 to the channel..."
    joinChannel 2 "peer0"
    infoln "Joining org2 peer1 to the channel..."
    joinChannel 2 "peer1"
    infoln "Joining org3 peer0 to the channel..."
    joinChannel 3 "peer0"
    infoln "Joining org3 peer1 to the channel..."
    joinChannel 3 "peer1"


    successln "Channel '$CHANNEL_NAME' joined"
  fi
```

Εικόνα 38: Συνάρτηση αναφοράς που δημιουργεί το κανάλι και προσθέτει τους συμμετέχοντες.

Έχοντας θέσει το path του configtx.yaml αρχείου μας, καλούμε την συνάρτηση createChannelGenesisBlock:

```
createChannelGenesisBlock() {

  which configtxgen
  if [ "$?" -ne 0 ]; then
    fatalln "configtxgen tool not found."
  fi

  local outputFileName="./channel-artifacts/${CHANNEL_NAME}.block"

  set -x
  configtxgen -profile $CHANNEL_NAME -outputBlock $outputFileName -channelID $CHANNEL_NAME

  res=$?
  { set +x; } 2>/dev/null

  verifyResult $res "Failed to generate channel configuration transaction for profile: $CHANNEL_NAME"
}
```

Εικόνα 39: Συνάρτηση Δημιουργίας του Genesis Block [67] .

Στο αρχείο πολιτικών έχουμε ορίσει το κατάλληλο προφίλ του καναλιού μας το οποίο ονομάζεται common.

```
Profiles:
  common:
    <<: *ChannelDefaults
    Orderer:
      <<: *OrdererDefaults
      OrdererType: etcdraft
      EtcdRaft:
        Consenters:
          - Host: orderer.example.com
            Port: 7050
            ClientTLSCert: ../organizations/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.crt
            ServerTLSCert: ../organizations/ordererOrganizations/example.com/orderers/orderer.example.com/tls/server.crt
      Organizations:
        - *OrdererOrg
      Capabilities: *OrdererCapabilities
    Application:
      <<: *ApplicationDefaults
      Organizations:
        - *Org1
        - *Org2
        - *Org3
      Capabilities: *ApplicationCapabilities
```

Εικόνα 40: Προφίλ καναλιού εντός του yaml αρχείου πολιτικών μας.

Οι πολιτικές οι οποίες εμπεριέχονται στα:

1. ChannelDefaults
2. OrdererDefaults
3. OrdererCapabilities
4. ApplicationDefaults
5. ApplicationCapabilities

δεν τροποποιήθηκαν από αυτές που δίνονται στο default test-network, καθώς για τους σκοπούς της εργασίας μας βασικός στόχος είναι η μελέτη των ιδιωτικών συλλογών δεδομένων χωρίς διευκολύνσεις από πολιτικές του καναλιού.

Επόμενο βήμα είναι η κλήση της συνάρτησης joinChannel η οποία θα γίνει επαναλαμβανόμενα και για τους 6 peers. Η συνάρτηση αυτή βασίζεται στην αξιοποίηση της SetGlobals ώστε να πραγματοποιηθεί το Join εκάστοτε peer.

Ένα σημείο της SetGlobals παρουσιάζεται στην εικόνα 41.

```
elif [ "$USING_ORG" == "3" ]; then
  export CORE_PEER_LOCALMSPID="Org3MSP"
  export CORE_PEER_TLS_ROOTCERT_FILE=$PEER_ORG3_CA
  export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org3.example.com/users/Admin@org3.example.com/msp
  if [ "$PEER" == "peer0" ]; then
    export CORE_PEER_ADDRESS=localhost:11051
  elif [ "$PEER" == "peer1" ]; then
    export CORE_PEER_ADDRESS=localhost:12051
  else
    errorln "Unsupported peer: $PEER for organization: $USING_ORG"
  fi
```

Εικόνα 41: Τμήμα Συνάρτησης SetGlobals.

Η συνάρτηση αυτή εξάγει τα απαραίτητα credentials κάθε peer μαζί με το Admin certificate ώστε να μπορέσει στην συνέχεια να κληθεί η εντολή:

```
peer channel join -b $BLOCKFILE >&log.txt --tls --cafile $ORDERER_CA
```

Εικόνα 42: Κλήση εντολής peer channel join.

Στο command prompt καλούμε τα παραπάνω scripts:

```
nyra@gsf test-network % ./network.sh createMainChannel
Using docker and docker-compose
Creating channels.
If network is not up, starting nodes with CLI timeout of '5' tries and CLI delay of '3' seconds and using database 'leveldb
Using docker and docker-compose
Generating channel genesis block 'common.block'
/Users/nyra/MyNetwork/test-network/../bin/configtxgen
+ configtxgen -profile common -outputBlock ./channel-artifacts/common.block -channelID common
2024-06-18 17:49:42.594 EEST 0001 INFO [common.tools.configtxgen] main -> Loading configuration
2024-06-18 17:49:42.613 EEST 0002 INFO [common.tools.configtxgen.localconfig] completeInitialization -> orderer type: etcdraft
2024-06-18 17:49:42.613 EEST 0003 INFO [common.tools.configtxgen.localconfig] completeInitialization -> Orderer.EtcdRaft.Options unset, setting to tick_interval:"500ms
k:10 heartbeat_tick:1 max_inflight_blocks:5 snapshot_interval_size:16777216
2024-06-18 17:49:42.614 EEST 0004 INFO [common.tools.configtxgen.localconfig] Load -> Loaded configuration: /Users/nyra/MyNetwork/test-network/configtx/configtx.yaml
2024-06-18 17:49:42.620 EEST 0005 INFO [common.tools.configtxgen] doOutputBlock -> Generating genesis block
2024-06-18 17:49:42.620 EEST 0006 INFO [common.tools.configtxgen] doOutputBlock -> Creating application channel genesis block
2024-06-18 17:49:42.621 EEST 0007 INFO [common.tools.configtxgen] doOutputBlock -> Writing genesis block
```

Εικόνα 43: Κάλεσμα createMainChannel από το Command Prompt.

Μόλις το κανάλι δημιουργηθεί μας εμφανίζει το παρακάτω:

```
Status: 201
{
        "name": "common",
        "url": "/participation/v1/channels/common",
        "consensusRelation": "consenter",
        "status": "active",
        "height": 1
}

Channel 'common' created
```

Εικόνα 44: Επιτυχία Δημιουργίας Καναλιού common.

Τέλος καλείται η SetGlobals με την joinChannel και αρχίζουν οι Peers να συνδέονται στο κανάλι μας.

```
Joining org1 peer0 to the channel...
setGlobals: ORG=1, PEER=peer0
Using organization 1
Attempting to join channel 'common' on peer0.org1, attempt 1
+ peer channel join -b ./channel-artifacts/common.block --tls --cafile /Users/nyra/MyNetwork/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert
.pem
+ res=0
2024-06-18 17:49:48.785 EEST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2024-06-18 17:49:48.825 EEST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
Joining org1 peer1 to the channel...
setGlobals: ORG=1, PEER=peer1
Using organization 1
Attempting to join channel 'common' on peer1.org1, attempt 1
+ peer channel join -b ./channel-artifacts/common.block --tls --cafile /Users/nyra/MyNetwork/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert
.pem
+ res=0
2024-06-18 17:49:51.926 EEST 0001 INFO [channelCmd] InitCmdFactory -> Endorser and orderer connections initialized
2024-06-18 17:49:51.971 EEST 0002 INFO [channelCmd] executeJoin -> Successfully submitted proposal to join channel
```

Εικόνα 45: Επιτυχής σύνδεση των peers του Org1 στο κανάλι common.

Αντίστοιχα συνδέονται και οι peers των Org2 και Org3 μέχρι που τελικά το δίκτυο μας αποκτάει την μορφή της ενότητας 4.5.

5.1.3 Το Chaincode του Δικτύου

Ο βασικός άξονας της παρούσας διπλωματικής σχετίζεται με το chaincode που δημιουργήσαμε. Αυτό αποτελεί την διεπαφή με την οποία ο ασθενής, το Νοσοκομείο και το Διαγνωστικό Κέντρο θα μπορούν να επικοινωνούν με το δίκτυο, ώστε να διαβάσουν και να επεξεργαστούν τα δεδομένα των ιδιωτικών συλλογών.

```go
// BloodworkContract contract for managing Bloodwork
type BloodworkContract struct {
    contractapi.Contract
}

// Bloodwork data structure
type Bloodwork struct {
    PatientID      string  `json:"patientID"`
    TestDate       string  `json:"testDate"`
    WhiteBloodCell float64 `json:"whiteBloodCell"`
    RedBloodCell   float64 `json:"redBloodCell"`
    Hemoglobin     float64 `json:"hemoglobin"`
    PlateletCount  float64 `json:"plateletCount"`
}
```

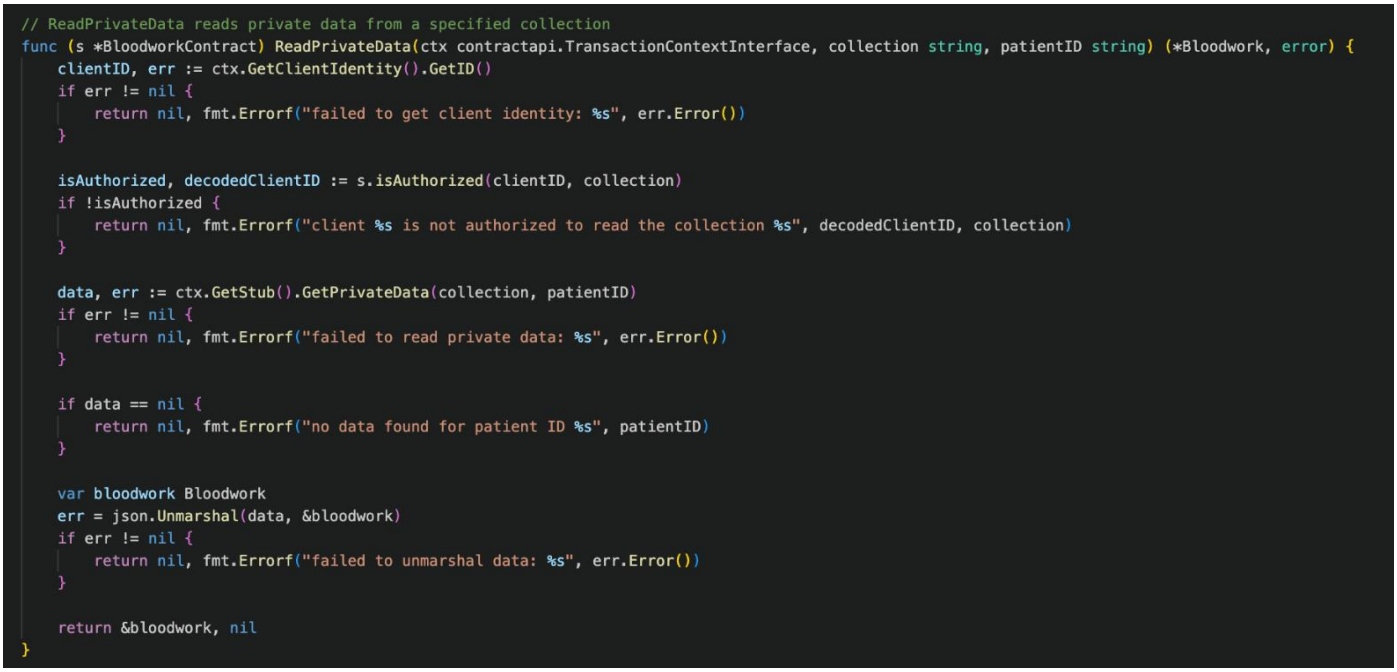
Εικόνα 46: Struct Αιματολογικών Εξετάσεων.

Για τους σκοπούς του δικτύου μας το chaincode αφορά δεδομένα αιματολογικών εξετάσεων που πραγματοποιεί ο ασθενής τόσο στο Νοσοκομείο όσο και στο Διαγνωστικό Κέντρο, κατά περίπτωση. Δημιουργήσαμε ένα βασικό struct που έχει την μορφή αιματολογικών εξετάσεων και περιέχει το ονοματεπώνυμο του ασθενούς, την ημερομηνία της εξέτασης όπως και βασικά δεδομένα των αιματολογικών εξετάσεων. Στην συνέχεια με την κατάλληλη κλήση συνάρτησης αρχικοποιούμε τις ιδιωτικές συλλογές με δεδομένα αυτής της μορφής.

Οι βασικές συναρτήσεις με τις οποίες τα επιμέρους μέλη θα διαβάζουν και θα προσθέτουν δεδομένα στις ιδιωτικές συλλογές παρουσιάζονται στις εικόνες 47 και 48.

```go
// PutPrivateData stores private data in a specified collection for authorized peers
func (s *BloodworkContract) PutPrivateData(ctx contractapi.TransactionContextInterface, collection string, patientID string, bloodworkJSON []byte) error {
    clientID, err := s.GetClientIdentity(ctx)
    if err != nil {
        return fmt.Errorf("failed to get client identity: %s", err.Error())
    }

    isAuthorized, decodedClientID := s.isAuthorized(clientID, collection)
    if !isAuthorized {
        return fmt.Errorf("client %s is not authorized to write to the collection %s", decodedClientID, collection)
    }

    return ctx.GetStub().PutPrivateData(collection, patientID, bloodworkJSON)
}
```

Εικόνα 47: Συνάρτηση Εγγραφής σε Ιδιωτική Συλλογή.



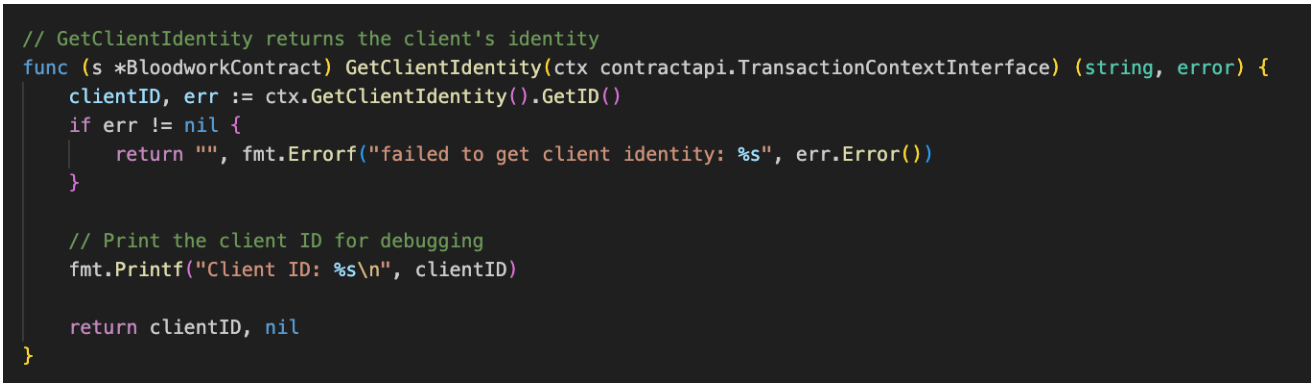
```go
// ReadPrivateData reads private data from a specified collection
func (s *BloodworkContract) ReadPrivateData(ctx contractapi.TransactionContextInterface, collection string, patientID string) (*Bloodwork, error) {
    clientID, err := ctx.GetClientIdentity().GetID()
    if err != nil {
        return nil, fmt.Errorf("failed to get client identity: %s", err.Error())
    }

    isAuthorized, decodedClientID := s.isAuthorized(clientID, collection)
    if !isAuthorized {
        return nil, fmt.Errorf("client %s is not authorized to read the collection %s", decodedClientID, collection)
    }

    data, err := ctx.GetStub().GetPrivateData(collection, patientID)
    if err != nil {
        return nil, fmt.Errorf("failed to read private data: %s", err.Error())
    }

    if data == nil {
        return nil, fmt.Errorf("no data found for patient ID %s", patientID)
    }

    var bloodwork Bloodwork
    err = json.Unmarshal(data, &bloodwork)
    if err != nil {
        return nil, fmt.Errorf("failed to unmarshal data: %s", err.Error())
    }

    return &bloodwork, nil
}
```

Εικόνα 48: Συνάρτηση Ανάγνωσης Δεδομένων Ιδιωτικής Συλλογής.

Και στις δύο περιπτώσεις έχουμε προσθέσει μια συνάρτηση που ονομάζεται isAuthorized και η οποία ελέγχει τα πιστοποιητικά των οντοτήτων που προσπαθούν να διαβάσουν δεδομένα ή να γράψουν στις ιδιωτικές συλλογές.

Αν δεν είναι εξουσιοδοτημένοι χρήστες τότε εμφανίζεται το μήνυμα ***"client (ClientID) is not authorized to write to the collection"*** για την περίπτωση της προσπάθειας εγγραφής δεδομένων ή ***"not authorized to read the collection"*** στην περίπτωση της ανάγνωσης.

Αρχικά ζητάμε το ClientID, το οποίο στην συνέχεια θα ελέγξουμε αν είναι εξουσιοδοτημένο. Αυτό πραγματοποιείται μέσω της συνάρτησης GetClientIdentity.



```go
// GetClientIdentity returns the client's identity
func (s *BloodworkContract) GetClientIdentity(ctx contractapi.TransactionContextInterface) (string, error) {
    clientID, err := ctx.GetClientIdentity().GetID()
    if err != nil {
        return "", fmt.Errorf("failed to get client identity: %s", err.Error())
    }

    // Print the client ID for debugging
    fmt.Printf("Client ID: %s\n", clientID)

    return clientID, nil
}
```

Εικόνα 49: GetClientIdentity Συνάρτηση.

Η GetClientIdentity εκμεταλλεύεται την αντίστοιχη εντολή και αποθηκεύει την τιμή που επιστρέφεται στο ClientID. Αυτή η τιμή είναι κωδικοποιημένη και συνεπώς η συνάρτηση isAuthorized αναλαμβάνει την αποκωδικοποίηση και ανάγνωση του περιεχομένου της.

```go
// / isAuthorized checks if a client is authorized to access a specific collection
func (s *BloodworkContract) isAuthorized(encodedClientID string, collection string) (bool, string) {
    // Decode the client ID
    decodedClientID, err := decodeBase64(encodedClientID)
    if err != nil {
        fmt.Printf("Failed to decode client ID: %s\n", err.Error())
        return false, ""
    }

    fmt.Printf("Checking authorization for decoded client ID: %s in collection: %s\n", decodedClientID, collection)
    authorized, exists := authorizedPeers[collection]
    if !exists {
        fmt.Println("Collection does not exist in authorizedPeers map.")
        return false, decodedClientID
    }

    for _, authClient := range authorized {
        fmt.Printf("Comparing with authorized client: %s\n", authClient)
        if authClient == decodedClientID {
            fmt.Println("Client is authorized.")
            return true, decodedClientID
        }
    }

    fmt.Println("Client is not authorized.")
    return false, decodedClientID
}
```

Εικόνα 50: Συνάρτηση isAuthorized.

Η συνάρτηση αυτή αφού πρώτα αποκωδικοποιήσει το περιεχόμενο του ClientID στην συνέχεια ελέγχει αν το ID αυτό περιέχεται στους επιτρεπόμενους χρήστες κάθε συλλογής:

- Για την **ιδιωτική συλλογή μεταξύ του Νοσοκομείου και του Ασθενούς** εξουσιοδοτημένες οντότητες είναι οι εξής : User1 του Org1 (αντιστοιχεί στην οντότητα του Ασθενούς1) και User1 του Org3 (κλινική 1 του Νοσοκομείου).

- Για την **ιδιωτική συλλογή μεταξύ του Διαγνωστικού Κέντρου και του Ασθενούς** εξουσιοδοτημένες οντότητες είναι οι εξής: User1 του Org1 (αντιστοιχεί στην οντότητα του Ασθενούς1) και User1 του Org2 (Τμήμα 1 του Διαγνωστικού Κέντρου).

Αυτό ήταν σημαντικό να υλοποιηθεί, καθώς ο τρόπος που λειτουργούν οι ιδιωτικές συλλογές δεδομένων στο Hyperledger Fabric δεν αποτρέπει οντότητες εντός ενός εξουσιοδοτημένου οργανισμού από το να διαβάσουν τα δεδομένα των συλλογών. Για παράδειγμα μια ιδιωτική συλλογή μεταξύ του Org1 και Org2 σημαίνει πως:

1. Όλες οι οντότητες εντός του Org1 και Org2 μπορούν να διαβάσουν τα δεδομένα της ιδιωτικής συλλογής.
2. Μόνο οι οντότητες του Org3 δεν γνωρίζουν το περιεχόμενο της ιδιωτικής συλλογής.

Κάτι τέτοιο θα έθετε θεμελιώδη ζητήματα στον τρόπο λειτουργίας του δικτύου καθώς τα περιεχόμενα των ιδιωτικών συλλογών είναι ευαίσθητα προσωπικά δεδομένα ασθενών. Με την παραπάνω υλοποίηση επομένως επιτεύχθηκε ο περαιτέρω έλεγχος

των ιδιωτικών συλλογών καθώς αυτές πλέον μπορούν να προσπελαστούν μόνο από εξουσιοδοτημένες οντότητες εντός των οργανισμών.

5.1.4 Δημιουργία των Private Data Collections και εγκατάσταση του Chaincode στο δίκτυο.

Έχοντας εκκινήσει το δίκτυο μας μπορούμε πλέον να εγκαταστήσουμε το chaincode που έχουμε δημιουργήσει, όπως και να ορίσουμε τις ιδιωτικές συλλογές δεδομένων.

Για την επίτευξη αυτού δημιουργήσαμε ένα script που ονομάζεται smart_contract.sh ώστε να μπορέσει να γίνει σωστά το deployment του κώδικα αλυσίδας. Το script αυτό εκμεταλλεύεται το ccutils.sh που παρέχει το test-network έπειτα από κάποιες αλλαγές που έγιναν για την υποστήριξη των δύο peers που διαθέτει κάθε οργανισμός.

Τα Private Data Collections ορίζονται σε ένα json αρχείο το οποίο επισυνάπτουμε κατά την εκτέλεση των εντολών για την προσθήκη του chaincode στο δίκτυο μας. Οι ιδιωτικές μας συλλογές παρουσιάζονται στην εικόνα 51.

```
[
 {
   "name": "HospitalPatientPrivateDataCollection",
   "policy": "OR('Org1MSP.member', 'Org3MSP.member')",
   "requiredPeerCount": 1,
   "maxPeerCount": 1,
   "blockToLive":1000000,
   "memberOnlyRead": true,
   "memberOnlyWrite": true,
   "endorsementPolicy": {
    "signaturePolicy":"OR('Org1MSP.member','Org3MSP.member')"
  }
},
 {
   "name": "TestCenterPatientPrivateDataCollection",
   "policy": "OR('Org1MSP.member', 'Org2MSP.member')",
   "requiredPeerCount": 1,
   "maxPeerCount": 1,
   "blockToLive":1000000,
   "memberOnlyRead": true,
   "memberOnlyWrite": true,
   "endorsementPolicy": {
     "signaturePolicy": "OR('Org1MSP.member', 'Org2MSP.member')"
   }
 }
]
```

Εικόνα 51: Ιδιωτικές Συλλογές Δεδομένων σε JSON μορφή.

Όπως αναφέρθηκε και προηγουμένως δημιουργούμε δύο ιδιωτικές συλλογές που ονομάζονται:

- HospitalPatientPrivateDataCollection για την ιδιωτική συλλογή μεταξύ του Ασθενούς και του Νοσοκομείου.
- TestCenterPatientPrivateDataCollection για την ιδιωτική συλλογή μεταξύ του Ασθενούς και του Διαγνωστικού Κέντρου.

Οι βασικές παράμετροι που μας ενδιαφέρουν εδώ είναι ότι οι πολιτικές είναι member Only Read και Write. Αυτό σημαίνει ότι κάποιος ο οποίος δεν συμμετέχει στις συλλογές αυτές δεν μπορεί να προσθέσει δεδομένα σε αυτές. Το Hyperledger Fabric δίνει την δυνατότητα μια οντότητα η οποία δεν ανήκει σε μια συλλογή να μπορεί να γράψει σε αυτή εφόσον δεν παραβιάζει την πολιτική έγκρισης. Κάτι τέτοιο στο σενάριο μας δεν ήταν επιθυμητό καθώς επιδιώκουμε ο ασθενής να έχει τον πρώτο λόγο σε όλα τα προσωπικά του δεδομένα. Επιπλέον, αν ο ίδιος θελήσει μελλοντικά να τα μοιραστεί με κάποια τρίτη οντότητα μπορεί να δημιουργήσει μια νέα ιδιωτική συλλογή και να τα μεταφέρει με αυτόν τον τρόπο. Περισσότερα για την μεταφορά δεδομένων μεταξύ ιδιωτικών συλλογών θα αναλυθούν σε επόμενη ενότητα.

Το script που δημιουργήσαμε για το deployment του chaincode παρουσιάζεται στην εικόνα 52.

```
if [ "$CHANNEL_NAME" = "common" ]; then
  infoln "Installing chaincode on org1 peers"
  installChaincode 1
  infoln "Install chaincode on org2 peers"
  installChaincode 2
  infoln "Install chaincode on org3 peers"
  installChaincode 3
  resolveSequence

  ## query whether the chaincode is installed
  queryInstalled 1

  ## approve the definition for org1
  approveForMyOrg 1

  ## check whether the chaincode definition is ready to be committed
  ## expect org1 to have approved and org2 and org3 not to
  checkCommitReadiness 1 "\"Org1MSP\": true" "\"Org2MSP\": false" "\"Org3MSP\": false"
  checkCommitReadiness 2 "\"Org1MSP\": true" "\"Org2MSP\": false" "\"Org3MSP\": false"
  checkCommitReadiness 3 "\"Org1MSP\": true" "\"Org2MSP\": false" "\"Org3MSP\": false"


  ## now approve also for org2
  approveForMyOrg 2

  ## check whether the chaincode definition is ready to be committed
  ## expect them both to have approved
  checkCommitReadiness 1 "\"Org1MSP\": true" "\"Org2MSP\": true" "\"Org3MSP\": false"
  checkCommitReadiness 2 "\"Org1MSP\": true" "\"Org2MSP\": true" "\"Org3MSP\": false"
  checkCommitReadiness 3 "\"Org1MSP\": true" "\"Org2MSP\": true" "\"Org3MSP\": false"

  ## now approve also for org3
  approveForMyOrg 3

  ## check whether the chaincode definition is ready to be committed
  ## expect them both to have approved
  checkCommitReadiness 1 "\"Org1MSP\": true" "\"Org2MSP\": true" "\"Org3MSP\": true"
  checkCommitReadiness 2 "\"Org1MSP\": true" "\"Org2MSP\": true" "\"Org3MSP\": true"
  checkCommitReadiness 3 "\"Org1MSP\": true" "\"Org2MSP\": true" "\"Org3MSP\": true"


  ## now that we know for sure all orgs have approved, commit the definition
  commitChaincodeDefinition 1 2 3

  ## query on all orgs to see that the definition committed successfully
  queryCommitted 1
  queryCommitted 2
  queryCommitted 3
fi
```

Εικόνα 52: Smart_contract.sh script για εγκατάσταση του chaincode στο δίκτυο.

Για το deployment του chaincode στο δίκτυο ακολουθούμε τα εξής βήματα:

1. Αρχικά πακετάρουμε το chaincode και δημιουργούμε τα Go dependencies που χρειάζονται στο δίκτυο μας.

```
infoln "Vendoring Go dependencies at $CC_SRC_PATH"
pushd $CC_SRC_PATH
GO111MODULE=on go mod vendor
popd
successln "Finished vendoring Go dependencies"
peer lifecycle chaincode package ${CC_NAME}.tar.gz --path ${CC_SRC_PATH} --lang ${CC_RUNTIME_LANGUAGE} --label ${CC_NAME}_${CC_VERSION} >&log.txt
PACKAGE_ID=$(peer lifecycle chaincode calculatepackageid ${CC_NAME}.tar.gz)
```

Εικόνα 53: Πακετάρισμα του chaincode και δημιουργία των Go dependencies.

2. Εγκαθιστούμε το πλέον πακεταρισμένο chaincode σε όλους τους peers του καναλιού μας μέσω της κλήσης της συνάρτησης installChaincode.
3. Ελέγχουμε αν έχει εγκατασταθεί σωστά το chaincode με την queryInstalled.
4. Κάνουμε αποδοχή του chaincode σε επίπεδο οργανισμού μέσω της συνάρτησης approveForMyOrg (Μόνο ένας peer του οργανισμού χρειάζεται να πραγματοποιήσει αυτή την εντολή και οι υπόλοιποι ενημερώνονται από εκείνον).
5. Μόλις γίνει το approve ελέγχουμε αν είναι έτοιμο το chaincode να γίνει commited στο δίκτυο. Αυτή η διαδικασία πραγματοποιείται βαθμιαία ως εξής: Αρχικά κάνουμε approve για τον Org1 σημείο στο οποίο μέσω της συνάρτησης checkCommitReadiness ελέγχουμε αν ο Org1 είναι έτοιμος να κάνει commit. Στο σημείο αυτό, μόνο ο Org1 πρέπει να επιστρέφει την τιμή true. Στην συνέχεια κάνουμε approve για τον Org2 και πλέον θα επιστρέφονται δύο true τιμές. Τέλος, κάνουμε approve και για τον Org3 όπου πλέον έχουμε και τα 3 approvals.
6. Κάνουμε commit το chaincode μέσω της συνάρτησης commitChaincodeDefinition και περιμένουμε για όλους τους οργανισμούς να μας επιστρέψει την τιμή VALID.
7. Ελέγχουμε αν το chaincode έχει γίνει σωστά deployed στο δίκτυο μέσω της συνάρτησης queryCommited.

Τα στάδια ελέγχου:

1. queryInstalled
2. checkCommitReadiness
3. queryCommited

δεν είναι αναγκαία για το deployment του chaincode αλλά βοηθάνε για την εύρεση πιθανών errors, που υπό άλλες συνθήκες μπορεί να μην αντιλαμβανόμασταν και προτείνεται η αξιοποίηση τους από το fabric test-network [67].

Στις επόμενες εικόνες επισυνάπτονται screenshots που αφορούν την επιτυχή εκτέλεση των παραπάνω βημάτων στο command prompt:

```
Finished vendoring Go dependencies
Installing Chaincode for all peers of the network
Installing chaincode on org1 peers
setGlobals: ORG=1, PEER=peer0
Using organization 1
+ peer lifecycle chaincode queryinstalled --output json
+ jq -r 'try (.installed_chaincodes[].package_id)'
+ grep '^bloodwork_1.0:59beccff466b20d3f8e45f7c74233aaa10f821027f4c9e4c3b4dea572c3a24c0$'
+ test 1 -ne 0
+ peer lifecycle chaincode install bloodwork.tar.gz
+ res=0
2024-06-18 20:32:55.812 EEST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nNbloodwork_1.0:59beccff466b20d3f8e45f7c
74233aaa10f821027f4c9e4c3b4dea572c3a24c0\022\rbloodwork_1.0" >
2024-06-18 20:32:55.813 EEST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: bloodwork_1.0:59beccff466b20d3f8e45f7c74233aaa10f821027f
4c9e4c3b4dea572c3a24c0
Chaincode is installed on peer0.org1
setGlobals: ORG=1, PEER=peer1
Using organization 1
+ peer lifecycle chaincode queryinstalled --output json
+ jq -r 'try (.installed_chaincodes[].package_id)'
+ grep '^bloodwork_1.0:59beccff466b20d3f8e45f7c74233aaa10f821027f4c9e4c3b4dea572c3a24c0$'
+ test 1 -ne 0
+ peer lifecycle chaincode install bloodwork.tar.gz
+ res=0
2024-06-18 20:33:43.292 EEST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nNbloodwork_1.0:59beccff466b20d3f8e45f7c
74233aaa10f821027f4c9e4c3b4dea572c3a24c0\022\rbloodwork_1.0" >
2024-06-18 20:33:43.292 EEST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: bloodwork_1.0:59beccff466b20d3f8e45f7c74233aaa10f821027f
4c9e4c3b4dea572c3a24c0
Chaincode is installed on peer1.org1
Install chaincode on org2 peers
setGlobals: ORG=2, PEER=peer0
Using organization 2
+ peer lifecycle chaincode queryinstalled --output json
+ jq -r 'try (.installed_chaincodes[].package_id)'
+ grep '^bloodwork_1.0:59beccff466b20d3f8e45f7c74233aaa10f821027f4c9e4c3b4dea572c3a24c0$'
+ test 1 -ne 0
+ peer lifecycle chaincode install bloodwork.tar.gz
+ res=0
2024-06-18 20:34:29.238 EEST 0001 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Installed remotely: response:<status:200 payload:"\nNbloodwork_1.0:59beccff466b20d3f8e45f7c
74233aaa10f821027f4c9e4c3b4dea572c3a24c0\022\rbloodwork_1.0" >
2024-06-18 20:34:29.238 EEST 0002 INFO [cli.lifecycle.chaincode] submitInstallProposal -> Chaincode code package identifier: bloodwork_1.0:59beccff466b20d3f8e45f7c74233aaa10f821027f
4c9e4c3b4dea572c3a24c0
Chaincode is installed on peer0.org2
```

Εικόνα 54: Βήμα 1 και βήμα 2, πακετάρισμα του chaincode και εγκατάσταση του σε κάθε peer.

```
Using organization 1
+ peer lifecycle chaincode approveformyorg -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /Users/nyra/MyNetwork/test-network/organizations/orderer
Organizations/example.com/tlsca/tlsca.example.com-cert.pem --channelID common --name bloodwork --version 1.0 --package-id bloodwork_1.0:59beccff466b20d3f8e45f7c74233aaa10f821027f4c9
e4c3b4dea572c3a24c0 --sequence 1 --signature-policy 'OR('\''Org1MSP.member'\'','\''Org2MSP.member'\'','\''Org3MSP.member'\'')' --collections-config /Users/nyra/MyNetwork/test-networ
k/scripts/collections_config.json
+ res=0
2024-06-18 20:36:45.986 EEST 0001 INFO [chaincodeCmd] ClientWait -> txid [c0ac7fc330483fcfa4964adc01ddec6332e705d3627318905317b1cf46b13cf7] committed with status (VALID) at localhos
t:7051
Chaincode definition approved on peer0.org1 on channel 'common'
setGlobals: ORG=1, PEER=peer0
Using organization 1
Checking the commit readiness of the chaincode definition on peer0.org1 on channel 'common'...
Attempting to check the commit readiness of the chaincode definition on peer0.org1, Retry after 3 seconds.
+ peer lifecycle chaincode checkcommitreadiness --channelID common --name bloodwork --version 1.0 --sequence 1 --signature-policy 'OR('\''Org1MSP.member'\'','\''Org2MSP.member'\'','
\''Org3MSP.member'\'')' --collections-config /Users/nyra/MyNetwork/test-network/scripts/collections_config.json --output json
+ res=0
{
        "approvals": {
                "Org1MSP": true,
                "Org2MSP": false,
                "Org3MSP": false
        }
}
```

Εικόνα 55: Βήμα 4 και 5 , Approve για τον Org1 και έλεγχος για την ετοιμότητα του commit.

```
+ peer lifecycle chaincode commit -o localhost:7050 --ordererTLSHostnameOverride orderer.example.com --tls --cafile /Users/nyra/MyNetwork/test-network/organizations/ordererOrganizat
ions/example.com/tlsca/tlsca.example.com-cert.pem --channelID common --name bloodwork --peerAddresses localhost:7051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizati
ons/peerOrganizations/org1.example.com/tlsca/tlsca.org1.example.com-cert.pem --peerAddresses localhost:8051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerO
rganizations/org1.example.com/tlsca/tlsca.org1.example.com-cert.pem --peerAddresses localhost:9051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerOrganizati
ons/org2.example.com/tlsca/tlsca.org2.example.com-cert.pem --peerAddresses localhost:10051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerOrganizations/org2
.example.com/tlsca/tlsca.org2.example.com-cert.pem --peerAddresses localhost:11051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerOrganizations/org3.example
.com/tlsca/tlsca.org3.example.com-cert.pem --peerAddresses localhost:12051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerOrganizations/org3.example.com/tls
ca/tlsca.org3.example.com-cert.pem --version 1.0 --sequence 1 --signature-policy 'OR('\''Org1MSP.member'\'','\''Org2MSP.member'\'','\''Org3MSP.member'\'')' --collections-config /User
s/nyra/MyNetwork/test-network/scripts/collections_config.json
+ res=0
2024-06-18 20:37:50.620 EEST 0001 INFO [chaincodeCmd] ClientWait -> txid [84577d920d33acb6caccd57ff915cab35ff9f451033e7d78af8b8bb6b81a501b] committed with status (VALID) at localhos
t:8051
2024-06-18 20:37:51.014 EEST 0002 INFO [chaincodeCmd] ClientWait -> txid [84577d920d33acb6caccd57ff915cab35ff9f451033e7d78af8b8bb6b81a501b] committed with status (VALID) at localhos
t:7051
2024-06-18 20:37:51.038 EEST 0003 INFO [chaincodeCmd] ClientWait -> txid [84577d920d33acb6caccd57ff915cab35ff9f451033e7d78af8b8bb6b81a501b] committed with status (VALID) at localhos
t:10051
2024-06-18 20:37:51.054 EEST 0004 INFO [chaincodeCmd] ClientWait -> txid [84577d920d33acb6caccd57ff915cab35ff9f451033e7d78af8b8bb6b81a501b] committed with status (VALID) at localhos
t:12051
2024-06-18 20:37:51.110 EEST 0005 INFO [chaincodeCmd] ClientWait -> txid [84577d920d33acb6caccd57ff915cab35ff9f451033e7d78af8b8bb6b81a501b] committed with status (VALID) at localhos
t:9051
2024-06-18 20:37:51.166 EEST 0006 INFO [chaincodeCmd] ClientWait -> txid [84577d920d33acb6caccd57ff915cab35ff9f451033e7d78af8b8bb6b81a501b] committed with status (VALID) at localhos
t:11051
Chaincode definition committed on channel 'common'
```

Εικόνα 56: Βήμα 6, commit του chaincode στο δίκτυο για όλους τους peers.

5.1.5 Αρχικοποίηση Ιδιωτικών Συλλογών

Για την αρχικοποίηση των ιδιωτικών συλλογών έχουμε δημιουργήσει ένα script που το καλούμε μέσω της συνάρτησης **Initialize**. Το script αυτό καλεί τις δύο συναρτήσεις του chaincode οι οποίες περιέχουν τα dummy δεδομένα των ιδιωτικών συλλογών μας.

```go
// InitHospitalData initializes the HospitalPatientPrivateDataCollection with dummy data
func (s *BloodworkContract) InitHospitalData(ctx contractapi.TransactionContextInterface) error {
    clientID, err := s.GetClientIdentity(ctx)
    if err != nil {
        return fmt.Errorf("failed to get client identity: %s", err.Error())
    }

    isAuthorized, decodedClientID := s.isAuthorized(clientID, "HospitalPatientPrivateDataCollection")
    if !isAuthorized {
        return fmt.Errorf("client %s is not authorized to initialize the Patient Hospital data", decodedClientID)
    }

    fmt.Println("Initializing Hospital Patient Private Data Collection")

    hospitalBloodwork := Bloodwork{
        PatientID:      "Giannis S",
        TestDate:       "2024-04-01",
        WhiteBloodCell: 7.5,
        RedBloodCell:   5.2,
        Hemoglobin:     14.5,
        PlateletCount:  300000.0,
    }
    hospitalBloodworkJSON, err := json.Marshal(hospitalBloodwork)
    if err != nil {
        return err
    }

    return ctx.GetStub().PutPrivateData("HospitalPatientPrivateDataCollection", hospitalBloodwork.PatientID, hospitalBloodworkJSON)
}
```

Εικόνα 57: Συνάρτηση chaincode για την εισαγωγή dummy δεδομένων στην ιδιωτική συλλογή Νοσοκομείου-Ασθενούς 1.

```go
// InitTestCenterData initializes the TestCenterPrivateDataCollection with dummy data
func (s *BloodworkContract) InitTestCenterData(ctx contractapi.TransactionContextInterface) error {
    clientID, err := s.GetClientIdentity(ctx)
    if err != nil {
        return fmt.Errorf("failed to get client identity: %s", err.Error())
    }

    isAuthorized, decodedClientID := s.isAuthorized(clientID, "TestCenterPatientPrivateDataCollection")
    if !isAuthorized {
        return fmt.Errorf("client %s is not authorized to initialize the Test Center - Patient data", decodedClientID)
    }

    fmt.Println("Initializing Test Center Patient Private Data Collection")

    testCenterBloodwork := Bloodwork{
        PatientID:      "Giannis S",
        TestDate:       "2024-05-01",
        WhiteBloodCell: 8.0,
        RedBloodCell:   4.8,
        Hemoglobin:     13.9,
        PlateletCount:  310000.0,
    }
    testCenterBloodworkJSON, err := json.Marshal(testCenterBloodwork)
    if err != nil {
        return err
    }

    return ctx.GetStub().PutPrivateData("TestCenterPatientPrivateDataCollection", testCenterBloodwork.PatientID, testCenterBloodworkJSON)
}
```

Εικόνα 58: Συνάρτηση chaincode για την εισαγωγή dummy δεδομένων στην ιδιωτική συλλογή Διαγνωστικού Κέντρου - Ασθενούς 1.

Ο τρόπος για να κληθούν οι δύο συναρτήσεις του chaincode είναι μέσω της εντολής **peer chaincode invoke** του Hyperledger Fabric. Είναι σημαντικό να αναφέρουμε σε αυτό το σημείο πως:

a) Αυτή η εντολή καλείται μόνο για τα μέλη της εκάστοτε ιδιωτική συλλογής και πιο συγκεκριμένα μόνο για τον έναν από τους δύο peers του οργανισμού καθώς ο άλλος peer θα ενημερωθεί μέσω του Gossip πρωτόκολλου. Αναλυτικότερα για το δίκτυο μας:
    1. Το invoke της συνάρτησης **InitHospitalData** πραγματοποιείται για τον peer0 του Org1 και τον peer0 του Org3
    2. Το **InitTestCenterData** για τον peer0 του Org1 και τον peer0 του Org2.
    3. Οι peer1 των οργανισμών θα ενημερωθούν από το Gossip πρωτόκολλο.
b) Την εντολή αυτή την κάνουμε invoke με το **πιστοποιητικό του User1** κάθε οργανισμού. Αυτό όπως φαίνεται και στις εικόνες 57 και 58 ελέγχεται από τις συναρτήσεις που έχουν αναφερθεί στην ενότητα 5.1.3. Σε ένα πραγματικό σενάριο την εισαγωγή δεδομένων μπορεί να την κάνει μόνο ο εξουσιοδοτημένος γιατρός ή ο ίδιος ο ασθενής ο οποίος θέλει να τις στείλει στον προσωπικό του γιατρό. Επομένως επαληθεύουμε πως οι User2 δεν μπορούν να προσθέσουν δεδομένα στις ιδιωτικές συλλογές.

Τα παραπάνω ολοκληρώνονται επιτυχώς μέσω του Command Prompt ως εξής:

```
nyra@gsf test-network % ./network.sh Initialize
Using docker and docker-compose

SmartContractPeer: ORG=1, PEER=peer0
Invoking on peer0.org1 for private Data Collection of Patient-TestCenter on channel 'common'...
Attempting to Invoke on peer0.org1, Retry after 3 seconds.
+ peer chaincode invoke -o localhost:7050 -C common -n bloodwork -c '{"function":"InitTestCenterData","Args":[]}' --tls --cafile /Users/nyra/MyNetwork/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --peerAddresses localhost:7051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerOrganizations/org1.example.com/tlsca/tlsca.org1.example.com-cert.pem
+ res=0
2024-06-19 17:16:55.922 EEST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
Invoke successful on peer0.org1 on channel 'common'
SmartContractPeer: ORG=2, PEER=peer0
Invoking on peer0.org2 for private Data Collection of Patient-TestCenter on channel 'common'...
Attempting to Invoke on peer0.org2, Retry after 3 seconds.
+ peer chaincode invoke -o localhost:7050 -C common -n bloodwork -c '{"function":"InitTestCenterData","Args":[]}' --tls --cafile /Users/nyra/MyNetwork/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --peerAddresses localhost:9051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerOrganizations/org2.example.com/tlsca/tlsca.org2.example.com-cert.pem
+ res=0
2024-06-19 17:16:59.032 EEST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
Invoke successful on peer0.org2 on channel 'common'
SmartContractPeer: ORG=1, PEER=peer0
Invoking on peer0.org1 for private Data Collection of Patient-Hospital on channel 'common'...
Attempting to Invoke on peer0.org1, Retry after 3 seconds.
+ peer chaincode invoke -o localhost:7050 -C common -n bloodwork -c '{"function":"InitHospitalData","Args":[]}' --tls --cafile /Users/nyra/MyNetwork/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --peerAddresses localhost:7051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerOrganizations/org1.example.com/tlsca/tlsca.org1.example.com-cert.pem
+ res=0
2024-06-19 17:17:02.141 EEST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
Invoke successful on peer0.org1 on channel 'common'
SmartContractPeer: ORG=3, PEER=peer0
Invoking on peer0.org3 for private Data Collection of Patient-Hospital on channel 'common'...
Attempting to Invoke on peer0.org3, Retry after 3 seconds.
+ peer chaincode invoke -o localhost:7050 -C common -n bloodwork -c '{"function":"InitHospitalData","Args":[]}' --tls --cafile /Users/nyra/MyNetwork/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --peerAddresses localhost:11051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerOrganizations/org3.example.com/tlsca/tlsca.org3.example.com-cert.pem
+ res=0
2024-06-19 17:17:05.279 EEST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
Invoke successful on peer0.org3 on channel 'common'
```

Εικόνα 59: Συνάρτηση Initialize για προσθήκη dummy δεδομένων στις ιδιωτικές συλλογές.

5.2 Έλεγχος και δοκιμές ορθής λειτουργίας των ιδιωτικών συλλογών

Στην παρούσα ενότητα θα αναλύσουμε την λειτουργία των ιδιωτικών συλλογών σε σχέση με τους κανόνες και τις πολιτικές που έχουμε θεσπίσει, πραγματοποιώντας σενάρια δοκιμών και ελέγχων.

5.2.1 Σενάριο προσπέλασης ιδιωτικών συλλογών από Εγκεκριμένα μέλη

Αρχικά θα ελέγξουμε αν τα μέλη των ιδιωτικών συλλογών έχουν πρόσβαση σε αυτές. Για τον σκοπό αυτό έχουμε δημιουργήσει ένα script το οποίο ονομάζεται contract_testing.sh και το οποίο καλούμε για να τρέξει τα απαραίτητα queries.

Συνδεόμαστε μέσω του cli ως οι εξουσιοδοτημένοι users και ζητάμε πρόσβαση στην συνάρτηση ReadPrivateData του chaincode. Για να το πραγματοποιήσουμε αυτό αξιοποιούμε την συνάρτηση SmartContractPeer μέσω της οποίας γίνονται τα κατάλληλα exports:

```
SmartContractPeer() {
  local ORG=$1
  PEER=$2
  echo "SmartContractPeer: ORG=$ORG, PEER=$PEER"

  if [ "$ORG" == "1" ]; then
    export CORE_PEER_LOCALMSPID="Org1MSP"
    export CORE_PEER_TLS_ROOTCERT_FILE=$PEER_ORG1_CA
    if [ "$PEER" == "peer0" ]; then
      export CORE_PEER_ADDRESS=localhost:7051
      export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/User1@org1.example.com/msp
    elif [ "$PEER" == "peer1" ]; then
      export CORE_PEER_ADDRESS=localhost:8051
      export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org1.example.com/users/User2@org1.example.com/msp
    else
      errorln "Unsupported peer: $PEER for organization: $ORG"
    fi
  elif [ "$ORG" == "2" ]; then
    export CORE_PEER_LOCALMSPID="Org2MSP"
    export CORE_PEER_TLS_ROOTCERT_FILE=$PEER_ORG2_CA
    if [ "$PEER" == "peer0" ]; then
      export CORE_PEER_ADDRESS=localhost:9051
      export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/User1@org2.example.com/msp
    elif [ "$PEER" == "peer1" ]; then
      export CORE_PEER_ADDRESS=localhost:10051
      export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org2.example.com/users/User2@org2.example.com/msp
    else
      errorln "Unsupported peer: $PEER for organization: $ORG"
    fi

  elif [ "$ORG" == "3" ]; then
    export CORE_PEER_LOCALMSPID="Org3MSP"
    export CORE_PEER_TLS_ROOTCERT_FILE=$PEER_ORG3_CA
    if [ "$PEER" == "peer0" ]; then
      export CORE_PEER_ADDRESS=localhost:11051
      export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org3.example.com/users/User1@org3.example.com/msp
    elif [ "$PEER" == "peer1" ]; then
      export CORE_PEER_ADDRESS=localhost:12051
      export CORE_PEER_MSPCONFIGPATH=${PWD}/organizations/peerOrganizations/org3.example.com/users/User2@org3.example.com/msp
    else
      errorln "Unsupported peer: $PEER for organization: $ORG"
    fi
  else
    errorln "Unsupported organization: $ORG"
  fi
```

Εικόνα 60: Συνάρτηση SmartContractPeer για export των κατάλληλων παραμέτρων.

Αναλυτικότερα, ορίζουμε πως ο User1 κάθε οργανισμού θα αξιοποιεί τον peer0 ενώ ο User2 θα αξιοποιεί τον peer1. Το σημαντικό σημείο αυτής της συνάρτησης είναι το CORE_PEER_MSPCONFIGPATH που κάνει export το πιστοποιητικό του User και όχι κάποιου Admin ή του ίδιου του peer. Η επιλογή των peers μεταξύ των Users έγινε τυχαία και θα μπορούσε να είναι ακόμα και ο ίδιος peer και για τους δύο Users. Η προσθήκη περισσότερων peers όπως και ο διαχωρισμός τους μεταξύ των Users διευκολύνει την διαχείριση παράλληλων αιτημάτων.

Η λογική αυτής της επιλογής είναι ότι η οντότητα που μας ενδιαφέρει είναι ο User που θα εισέρχεται στην εφαρμογή με τα κρυπτογραφικά στοιχεία του. Ο peer απλώς εκτελεί την λογική του δικτύου μας και διατηρεί το ledger και τις ιδιωτικές συλλογές.

Κάνουμε λοιπόν τα εξής queries:

```
CHANNEL_NAME="common"
SmartContractPeer 1 "peer0"
infoln "Quering on peer0.org1 used by user1 (Ασθενής 1) for private Collection of Test Center-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["TestCenterPatientPrivateDataCollection", "Giannis S"]}'
SmartContractPeer 2 "peer0"
infoln "Quering on peer0.org2 used by user1 (Τμήμα 1) for private Collection of Test Center-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["TestCenterPatientPrivateDataCollection", "Giannis S"]}'
SmartContractPeer 1 "peer0"
infoln "Quering on peer0.org1 used by user1 (Ασθενής 1) for private Collection of Hospital-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["HospitalPatientPrivateDataCollection", "Giannis S"]}'
SmartContractPeer 3 "peer0"
infoln "Quering on peer0.org1 used by user1 (Κλινική 1) for private Collection of Hospital-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["HospitalPatientPrivateDataCollection", "Giannis S"]}'
```

Εικόνα 61: Chaincode Queries από τους εγκεκριμένους χρήστες.

Αξιοποιούμε τα credentials του user1 του Org1 (Ασθενής 1) όπως και αυτά του user1 του Org2 (Τμήμα 1 Διαγνωστικού Κέντρου) και ζητάμε την ανάγνωση των δεδομένων της ιδιωτικής συλλογής που μοιράζονται μεταξύ τους.

Αντίστοιχα πράττουμε και για την δεύτερη συλλογή που μοιράζεται ο Ασθενής 1 με την Κλινική 1 του Νοσοκομείου

Στο command prompt μας επιστρέφονται σωστά τα δεδομένα που έχουμε αποθηκεύσει στις ιδιωτικές συλλογές.

```
SmartContractPeer: ORG=1, PEER=peer0
Quering on peer0.org1 used by user1 (Ασθενής 1) for private Collection of Test Center-Patient on channel 'common'...
{"patientID":"Giannis S","testDate":"2024-05-01","whiteBloodCell":8,"redBloodCell":4.8,"hemoglobin":13.9,"plateletCount":310000}
SmartContractPeer: ORG=2, PEER=peer0
Quering on peer0.org2 used by user1 (Τμήμα 1) for private Collection of Test Center-Patient on channel 'common'...
{"patientID":"Giannis S","testDate":"2024-05-01","whiteBloodCell":8,"redBloodCell":4.8,"hemoglobin":13.9,"plateletCount":310000}
SmartContractPeer: ORG=1, PEER=peer0
Quering on peer0.org1 used by user1 (Ασθενής 1) for private Collection of Hospital-Patient on channel 'common'...
{"patientID":"Giannis S","testDate":"2024-04-01","whiteBloodCell":7.5,"redBloodCell":5.2,"hemoglobin":14.5,"plateletCount":300000}
SmartContractPeer: ORG=3, PEER=peer0
Quering on peer0.org1 used by user1 (Κλινική 1) for private Collection of Hospital-Patient on channel 'common'...
{"patientID":"Giannis S","testDate":"2024-04-01","whiteBloodCell":7.5,"redBloodCell":5.2,"hemoglobin":14.5,"plateletCount":300000}
```

Εικόνα 62: Επιστροφή Δεδομένων μετά την κλήση των queries.

Ο Ασθενής 1 βλέπει και τις δύο συλλογές οι οποίες διατηρούν διαφορετικές εξετάσεις που έχει πραγματοποιήσει.

5.2.2 Σενάριο προσπέλασης ιδιωτικών συλλογών από Μη Εγκεκριμένα μέλη

Σε αυτό το σενάριο θα εκτελέσουμε τα ίδια queries αλλά αυτή την φορά από μη εξουσιοδοτημένα μέλη και πιο συγκεκριμένα:

- Τον Ασθενή 2 (User2) του Org1 και το Τμήμα 2(User2) του Διαγνωστικού Κέντρου για την πρώτη ιδιωτική συλλογή.
- Τον Ασθενή 2 (User2) του Org1 και την Κλινική 2User2) του Νοσοκομείου για την δεύτερη ιδιωτική συλλογή.
- Την Κλινική 1 του Νοσοκομείου για την ιδιωτική συλλογή που δεν συμμετέχει.
- Το Τμήμα 1 του Διαγνωστικού Κέντρου για την ιδιωτική συλλογή που δεν συμμετέχει.

Τα παραπάνω παρουσιάζονται στην εικόνα 63.

```
CHANNEL_NAME="common"
SmartContractPeer 1 "peer1"
infoln "Quering on peer1.org1 used by user2 (Ασθενής 2) for private Collection of Test Center-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["TestCenterPatientPrivateDataCollection", "Giannis S"]}'
SmartContractPeer 2 "peer1"
infoln "Quering on peer1.org2 used by user2 (Τμήμα 2) for private Collection of Test Center-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["TestCenterPatientPrivateDataCollection", "Giannis S"]}'
SmartContractPeer 1 "peer1"
infoln "Quering on peer1.org1 used by user2 (Ασθενής 2) for private Collection of Hospital-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["HospitalPatientPrivateDataCollection", "Giannis S"]}'
SmartContractPeer 3 "peer1"
infoln "Quering on peer0.org1 used by user2 (Κλινική 2) for private Collection of Hospital-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["HospitalPatientPrivateDataCollection", "Giannis S"]}'
SmartContractPeer 3 "peer0"
infoln "Quering on peer0.org3 used by user1 (Κλινική 1) for private Collection of Test Center-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["TestCenterPatientPrivateDataCollection", "Giannis S"]}'
SmartContractPeer 2 "peer0"
infoln "Quering on peer0.org2 used by user1 (Τμήμα 1) for private Collection of Hospital-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["HospitalPatientPrivateDataCollection", "Giannis S"]}'
```

Εικόνα 63: Queries για μη εξουσιοδοτημένους χρήστες.

Στο command prompt αυτά τα queries μας επιστρέφουν τα εξής:

```
SmartContractPeer: ORG=1, PEER=peer1
Quering on peer1.org1 used by user2 (Ασθενής 2) for private Collection of Test Center-Patient on channel 'common'...
Error: endorsement failure during query. response: status:500 message:"client x509::CN=User2@org1.example.com,OU=client,L=San Francisco,ST=California,C=US::CN=ca.org1.example.com,O=
org1.example.com,L=San Francisco,ST=California,C=US is not authorized to read the collection TestCenterPatientPrivateDataCollection"
SmartContractPeer: ORG=2, PEER=peer1
Quering on peer1.org2 used by user2 (Τμήμα 2) for private Collection of Test Center-Patient on channel 'common'...
Error: endorsement failure during query. response: status:500 message:"client x509::CN=User2@org2.example.com,OU=client,L=San Francisco,ST=California,C=US::CN=ca.org2.example.com,O=
org2.example.com,L=San Francisco,ST=California,C=US is not authorized to read the collection TestCenterPatientPrivateDataCollection"
SmartContractPeer: ORG=1, PEER=peer1
Quering on peer1.org1 used by user2 (Ασθενής 2) for private Collection of Hospital-Patient on channel 'common'...
Error: endorsement failure during query. response: status:500 message:"client x509::CN=User2@org1.example.com,OU=client,L=San Francisco,ST=California,C=US::CN=ca.org1.example.com,O=
org1.example.com,L=San Francisco,ST=California,C=US is not authorized to read the collection HospitalPatientPrivateDataCollection"
SmartContractPeer: ORG=3, PEER=peer1
Quering on peer0.org1 used by user2 (Κλινική 2) for private Collection of Hospital-Patient on channel 'common'...
Error: endorsement failure during query. response: status:500 message:"client x509::CN=User2@org3.example.com,OU=client,L=San Francisco,ST=California,C=US::CN=ca.org3.example.com,O=
org3.example.com,L=San Francisco,ST=California,C=US is not authorized to read the collection HospitalPatientPrivateDataCollection"
SmartContractPeer: ORG=3, PEER=peer0
Quering on peer0.org3 used by user1 (Κλινική 1) for private Collection of Test Center-Patient on channel 'common'...
Error: endorsement failure during query. response: status:500 message:"client x509::CN=User1@org3.example.com,OU=client,L=San Francisco,ST=California,C=US::CN=ca.org3.example.com,O=
org3.example.com,L=San Francisco,ST=California,C=US is not authorized to read the collection TestCenterPatientPrivateDataCollection"
SmartContractPeer: ORG=2, PEER=peer0
Quering on peer0.org2 used by user1 (Τμήμα 1) for private Collection of Hospital-Patient on channel 'common'...
Error: endorsement failure during query. response: status:500 message:"client x509::CN=User1@org2.example.com,OU=client,L=San Francisco,ST=California,C=US::CN=ca.org2.example.com,O=
org2.example.com,L=San Francisco,ST=California,C=US is not authorized to read the collection HospitalPatientPrivateDataCollection"
Quering chaincode failed
```

Εικόνα 64: Αποτυχία προσπέλασης δεδομένων από μη εξουσιοδοτημένους χρήστες.

Παρατηρούμε, ότι όλοι οι μη εξουσιοδοτημένοι χρήστες δεν κατάφεραν να προσπελάσουν τις ιδιωτικές συλλογές δεδομένων ακόμα και αν ανήκουν στον ίδιο οργανισμό.

5.3 Υλοποίηση σεναρίου μεταφοράς εξέτασης ασθενούς από Διαγνωστικό Κέντρο σε Νοσοκομείο

Όπως αναφέρθηκε στην ενότητα 4.5, ένας βασικός άξονας ενδιαφέροντος είναι η ασφαλής μεταφορά αποτελεσμάτων διαγνωστικών εξετάσεων του ασθενούς, όταν αυτές διεκπεραιώνονται, μέσω κάποιου παραπεμπτικού, σε εξωτερική Δομή Υγείας διαφορετική από αυτή του προσωπικού του γιατρού. Στο chaincode που δημιουργήσαμε έχουμε συμπεριλάβει μια κατάλληλη συνάρτηση για αυτό τον σκοπό και η οποία παρουσιάζεται στην εικόνα 65.

```go
// TransferBloodwork transfers bloodwork data from one private collection to another
func (s *BloodworkContract) TransferBloodwork(ctx contractapi.TransactionContextInterface, fromCollection string, toCollection string, patientID string) error {

    // Get client identity
    clientID, err := s.GetClientIdentity(ctx)
    if err != nil {
        return fmt.Errorf("failed to get client identity: %s", err.Error())
    }
    // Check authorization for the source collection
    isAuthorizedFrom, decodedClientID := s.isAuthorized(clientID, fromCollection)
    if !isAuthorizedFrom {
        return fmt.Errorf("client %s is not authorized to read from collection %s", decodedClientID, fromCollection)
    }
    // Verify the bloodwork data exists in the 'from' collection
    existingData, err := ctx.GetStub().GetPrivateData(fromCollection, patientID)
    if err != nil {
        return fmt.Errorf("failed to read private data from collection %s: %s", fromCollection, err.Error())
    }
    if existingData == nil {
        return fmt.Errorf("no data found for patient ID %s in collection %s", patientID, fromCollection)
    }
    // Unmarshal the existing data
    var bloodwork Bloodwork
    err = json.Unmarshal(existingData, &bloodwork)
    if err != nil {
        return fmt.Errorf("failed to unmarshal existing bloodwork data: %s", err.Error())
    }
    // Prepare the bloodwork data as Json Format
    bloodworkJSON, err := json.Marshal(bloodwork)
    if err != nil {
        return fmt.Errorf("failed to marshal bloodwork data: %s", err.Error())
    }
    isAuthorizedTo, _ := s.isAuthorized(clientID, toCollection)
    if !isAuthorizedTo {
        return fmt.Errorf("client %s is not authorized to write to collection %s", decodedClientID, toCollection)
    }

    fmt.Println("Authorization checks passed")
    // Write the bloodwork data to the destination collection
    err = ctx.GetStub().PutPrivateData(toCollection, patientID, bloodworkJSON)
    if err != nil {
        return fmt.Errorf("failed to put private data to collection %s: %s", toCollection, err.Error())
    }
    // Optionally, delete the data from the source collection
    err = ctx.GetStub().DelPrivateData(fromCollection, patientID)
    if err != nil {
        return fmt.Errorf("failed to delete private data from collection %s: %s", fromCollection, err.Error())
    }

    return nil
}
```

Εικόνα 65: Συνάρτηση TransferBloodwork για μεταφορά αποτελεσμάτων από Διαγνωστικό Κέντρο σε Νοσοκομείο.

Στόχος ήταν να δημιουργηθεί μια ασφαλής μέθοδος όπου ο ασθενής θα είναι αυτός που έχει τον απόλυτο έλεγχο των δεδομένων του. Για την επίτευξη αυτού, δημιουργήσαμε δύο σημεία ελέγχου στην συνάρτηση μεταφοράς των εξετάσεων. Το πρώτο ελέγχει αν ο User έχει εγκριθεί να διαβάσει τα δεδομένα της πρώτης συλλογής και το δεύτερο αν έχει εγκριθεί για να γράψει στα δεδομένα της δεύτερης συλλογής. Επιπλέον, έχει προστεθεί μια εντολή διαγραφής των δεδομένων από την πρώτη συλλογή όταν αυτά μεταφερθούν. Με αυτό τον τρόπο ο ασθενής μπορεί να μεταφέρει τα δεδομένα του με ασφάλεια από την ιδιωτική συλλογή που μοιράζεται με το Διαγνωστικό Κέντρο σε αυτή που μοιράζεται με το Νοσοκομείο. Στην συνέχεια διαγράφει τα δεδομένα από την συλλογή με το Διαγνωστικό Κέντρο και το οποίο πλέον δεν έχει πρόσβαση σε αυτά.

Η κλήση αυτής της συνάρτησης απαιτεί ένα chaincode invoke από τον Ασθενή 1 δηλαδή τον User1 του Org1 και παρουσιάζεται στην εικόνα 66.

```
SmartContractPeer 1 "peer0"
infoln "Initializing the transfer of the bloodwork test from the TestCenter Collection to the Hospital one on channel '$CHANNEL_NAME'..."
rc=1
COUNTER=1
# continue to poll
# we either get a successful response, or reach MAX RETRY
while [ $rc -ne 0 -a $COUNTER -lt $MAX_RETRY ]; do
    sleep $DELAY
    infoln "Attempting to Invoke on peer0.org1 Retry after $DELAY seconds."
    set -x
    peer chaincode invoke -o localhost:7050 -C $CHANNEL_NAME -n ${CC_NAME} -c '{"function":"TransferBloodwork","Args":["TestCenterPatientPrivateDataCollection", "HospitalPatientPrivateDataCollection", "Giannis S"]}'
    res=$?
    { set +x; } 2>/dev/null
    let rc=$res
    COUNTER=$(expr $COUNTER + 1)
done
cat log.txt
if test $rc -eq 0; then
    successln "Invoke successful on peer0.org${ORG} on channel '$CHANNEL_NAME'"
else
    fatalln "After $MAX_RETRY attempts, Invoke result on peer0.org${ORG} is INVALID!"
fi
```

Εικόνα 66: Κλήση συνάρτησης μεταφοράς δεδομένων ως Ασθενής 1.

Στο command prompt καλούμε το αντίστοιχο script και βλέπουμε την επιτυχή ολοκλήρωση της μεταφοράς των στοιχείων

```
nyra@gsf test-network % ./network.sh Testing
Using docker and docker-compose

SmartContractPeer: ORG=1, PEER=peer0
Initializing the transfer of the bloodwork test from the TestCenter Collection to the Hospital one on channel 'common'...
Attempting to Invoke on peer0.org1 Retry after 3 seconds.
+ peer chaincode invoke -o localhost:7050 -C common -n bloodwork -c '{"function":"TransferBloodwork","Args":["TestCenterPatientPrivateDataCollection", "HospitalPatientPrivateDataCol
lection", "Giannis S"]}' --tls --cafile /Users/nyra/MyNetwork/test-network/organizations/ordererOrganizations/example.com/tlsca/tlsca.example.com-cert.pem --peerAddresses localhost:
7051 --tlsRootCertFiles /Users/nyra/MyNetwork/test-network/organizations/peerOrganizations/org1.example.com/tlsca/tlsca.org1.example.com-cert.pem
+ res=0
2024-06-19 22:28:13.035 EEST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200
Invoke successful on peer0.org on channel 'common'
```

Εικόνα 67: Επιτυχής μεταφορά εξετάσεων ασθενούς.

Τελευταίο βήμα είναι (α) να ελέγξουμε αν διαγράφηκαν επιτυχώς οι εξετάσεις του ασθενούς από την ιδιωτική συλλογή που διατηρεί με το Διαγνωστικό κέντρο και (β) αν η Κλινική 1 του Νοσοκομείου μπορεί να δει τις νέες εξετάσεις του ασθενούς.

Για να κάνουμε αυτούς τους ελέγχους πραγματοποιούμε δύο queries τα οποία παρουσιάζονται στην εικόνα 68.

- Το πρώτο ως User1 του Διαγνωστικού Κέντρου.
- Το δεύτερο ως User1 του Νοσοκομείου.

```
CHANNEL_NAME="common"
SmartContractPeer 2 "peer0"
infoln "Quering on peer0.org1 used by user1 (Τμήμα 1) for private Collection of Test Center-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["TestCenterPatientPrivateDataCollection", "Giannis S"]}'
SmartContractPeer 3 "peer0"
infoln "Quering on peer0.org3 used by user1 (Κλινική 1) for private Collection of Hospital-Patient on channel '$CHANNEL_NAME'..."
peer chaincode query -C common -n bloodwork -c '{"function":"ReadPrivateData","Args":["HospitalPatientPrivateDataCollection", "Giannis S"]}'
```

Εικόνα 68: Queries για έλεγχο σωστής μεταφοράς δεδομένων.

Η εκτέλεση τους πραγματοποιείται στο command prompt και παρουσιάζεται στην εικόνα 69.

```
nyra@gsf test-network % ./network.sh Testing
Using docker and docker-compose

SmartContractPeer: ORG=2, PEER=peer0
Quering on peer0.org1 used by user1 (Τμήμα 1) for private Collection of Test Center-Patient on channel 'common'...
Error: endorsement failure during query. response: status:500 message:"no data found for patient ID Giannis S"
SmartContractPeer: ORG=3, PEER=peer0
Quering on peer0.org3 used by user1 (Κλινική 1) for private Collection of Hospital-Patient on channel 'common'...
{"patientID":"Giannis S","testDate":"2024-05-01","whiteBloodCell":8,"redBloodCell":4.8,"hemoglobin":13.9,"plateletCount":310000}
```

Εικόνα 69: Επιτυχής Εκτέλεση μεταφοράς προσωπικών εξετάσεων ασθενούς.

Παρατηρούμε πως το τμήμα 1 του Διαγνωστικού Κέντρου δεν μπορεί να δει πλέον τις εξετάσεις του Ασθενούς 1 και επιστρέφει μήνυμα ***"no data found for patient ID Giannis S"***. Αντίστοιχα η κλινική 1 του Νοσοκομείου έχει τις νέες εξετάσεις τους ασθενούς που πραγματοποιήθηκαν 01-05-2024.

Μέρος ΙΙΙ

Επίλογος

Κεφάλαιο 6

Επίλογος

6.1 Συζήτηση – Συμπεράσματα

Η παρούσα διπλωματική αποσκοπούσε στην εύρεση μιας λειτουργικής προσέγγισης έναντι των σημερινών πρακτικών που ισχύουν στον Τομέα Υγείας στην Ελλάδα αξιοποιώντας την τεχνολογία του blockchain. Στο πλαίσιο αυτό, προτάθηκε η δημιουργία ενός αποκεντρωμένου δικτύου στο οποίο θα συμμετέχει κάθε είδους Δομή Υγείας που μπορεί να αφορά Δημόσια ή Ιδιωτικά Νοσοκομεία, Κέντρα Υγείας, Ιδιώτες Γιατρούς κ.ο.κ με στόχο την μεταφορά αποτελεσμάτων εξετάσεων ενός ασθενούς.

Για τον σκοπό αυτό, δημιουργήθηκε ένα παράδειγμα χρήσης αξιοποιώντας το Hyperledger Fabric. Το παράδειγμα αυτό στόχευε στην μελέτη του τρόπου που θα μπορούσε να δημιουργηθεί ένα ενοποιημένο δίκτυο στο οποίο θα υπάρχουν επιπρόσθετα πεδία ιδιωτικότητας αξιοποιώντας τις ιδιωτικές συλλογές δεδομένων.

Σε συνέχεια βιβλιογραφικής έρευνας και πρακτικών δοκιμών διαπιστώθηκε ότι βασικό πρόβλημα αποτελεί η αδυναμία του Hyperledger Fabric να διαχωρίζει εσωτερικά την δομή εκάστοτε οργανισμού ώστε να μην επιτρέπεται σε όλα τα μέλη του να έχουν πρόσβαση στα δεδομένα των ιδιωτικών συλλογών. Ακόμα και στην περίπτωση που επιλέγαμε να διαχωρίσουμε την οντότητα του ασθενούς, ώστε καθένας να αποτελεί αυτούσιο μέλος του δικτύου, το συγκεκριμένο πρόβλημα δεν θα είχε επιλυθεί. Πιο συγκεκριμένα, λόγω του GDPR και της ισχύουσας ευρωπαϊκής και εθνικής νομοθεσίας, ήταν σημαντικό ακόμα και η οντότητα του Νοσοκομείου ή του Διαγνωστικού Κέντρου να μην γνωρίζει επακριβώς τα δεδομένα του ασθενούς, παρά μόνο οι εξουσιοδοτημένες κλινικές και τμήματα που αφορούν τον ασθενή.

Για την αντιμετώπιση του συγκεκριμένου προβλήματος η ιδέα μας ήταν να δημιουργήσουμε τους κατάλληλους κανόνες ελέγχου στο έξυπνο συμβόλαιο που αναπτύξαμε για τις ιδιωτικές συλλογές. Προσθέτοντας τις παραμέτρους αυθεντικοποίησης του εκάστοτε χρήστη ώστε να μπορέσει να προσπελάσει τα δεδομένα, δόθηκε η δυνατότητα δημιουργίας ενός ασφαλούς περιβάλλοντος και μιας λειτουργικής λύσης στα μέχρι σήμερα προβλήματα που αντιμετωπίζει η μεταφορά των αποτελεσμάτων ιατρικών εξετάσεων του ασθενούς. Συνεπώς, παρατηρούμε πως το chaincode και οι παράμετροι που θέτουμε σε αυτό παίζουν κομβικό ρόλο στην σωστή λειτουργία του δικτύου μας.

Στο σημείο αυτό αξίζει να σημειωθεί, πως ένα ακόμα σημαντικό κομμάτι του Hyperledger Fabric είναι οι πολλές δυνατότητες παραμετροποίησης που προσφέρει στις επιμέρους πολιτικές τόσο των καναλιών όσο και τον συμμετεχόντων οργανισμών. Μέσω του ορισμού των κατάλληλων κανόνων όπως των απαραίτητων υπογραφών που χρειάζονται για την εκτέλεση του chaincode και των επιμέρους

λειτουργιών, μας δίνεται η δυνατότητα δημιουργίας ακόμα περισσότερων κανόνων ώστε ο ασθενής να έχει τον πρώτο λόγο στα προσωπικά του δεδομένα.

Παραδείγματος χάριν, θα μπορούσαμε να θέσουμε μια πολιτική AND στις ιδιωτικές συλλογές έτσι ώστε σε κάθε βήμα να απαιτείται η υπογραφή και των δύο οντοτήτων αναφορικά με την επεξεργασία και ανάγνωση των δεδομένων ή θα μπορούσε να οριστεί μια πολιτική καναλιού όπου για κάθε συναλλαγή να απαιτείται τουλάχιστον μια υπογραφή από την οντότητα του ασθενούς κ.ο.κ.

Συμπερασματικά, διαπιστώσαμε ότι η αξιοποίηση της συγκεκριμένης λύσης προσφέρει σημαντικά συγκριτικά πλεονεκτήματα. Διακρίνεται από ασφάλεια, ακεραιότητα και κυρίως επεκτασιμότητα και ως εκ τούτου στο μέλλον μπορεί να αξιοποιηθεί σε πρακτικές εφαρμογές του τομέα Υγείας σε μια ευρεία γκάμα σχετικών υπηρεσιών (ηλεκτρονικός φάκελος ασθενούς, ηλεκτρονική συνταγογράφηση κ.λ.π.).

6.2 Μελλοντικές Επεκτάσεις

Στην συγκεκριμένη ενότητα θα συζητηθούν μελλοντικές επεκτάσεις που θα μπορούσαν να αξιοποιηθούν για την βελτίωση της αποδοτικότητας του δικτύου που υλοποιήθηκε, καθώς και να εισάγουν νέες λειτουργίες και τεχνολογίες που θα εξασφαλίσουν την περαιτέρω ανάπτυξη του έργου.

Αρχικά, μια σημαντική μελλοντική επέκταση είναι η δημιουργία μιας αυτοματοποιημένης διαδικασίας η οποία θα αναλαμβάνει την ανάπτυξη των έξυπνων συμβολαίων μεταξύ του ασθενούς και της Δομής Υγείας. Πιο συγκεκριμένα, ο τρόπος που λειτουργεί το δίκτυο είναι ότι κάθε ασθενής συμφωνεί με κάποιο τμήμα ή κλινική μιας Δομής Υγείας να δημιουργήσουν μια ιδιωτική συλλογή με στόχο την ανταλλαγή δεδομένων. Αρκεί επομένως να συμφωνήσουν και να υπογράψουν στο έξυπνο συμβόλαιο ποια είναι τα αυθεντικοποιημένα μέλη τα οποία θα ελέγχονται μέσω της συνάρτησης isAuthorized που αναλύθηκε στο κεφάλαιο 5. Οι συναρτήσεις που υλοποιούν την επιχειρηματική λογική για την ανάγνωση, εγγραφή και μεταφορά των δεδομένων έχουν υλοποιηθεί και παρουσιάζονται στην ενότητα 5.1.3 του chaincode. Συνεπώς, πολύ εύκολα μπορεί να επεκταθεί ο τρόπος ανάπτυξης νέων έξυπνων συμβολαίων για την δημιουργία ιδιωτικών συλλογών.

Επόμενη σημαντική επέκταση, είναι η δημιουργία της κατάλληλης διεπαφής την οποία θα χρησιμοποιούν ο ασθενής και οι Δομές Υγείας. Στο σημείο αυτό, μπορούμε να δημιουργήσουμε ένα φιλικό και εύχρηστο περιβάλλον το οποίο θα μπορεί να χρησιμοποιηθεί από όλους τους χρήστες χωρίς να απαιτεί ειδικότερες γνώσεις σχετικά με την τεχνολογία που αξιοποιεί το δίκτυο ώστε όλα τα εμπλεκόμενα μέρη να μπορούν να διαχειριστούν τα δεδομένα που τους αφορούν με τρόπο εύκολο και κατανοητό.

Στο εισαγωγικό κεφάλαιο δόθηκε μια μικρή ανάλυση του Cloud Computing και πως αυτό μπορεί να αξιοποιηθεί συνδυαστικά με την τεχνολογία του blockchain για να παρέχει υπηρεσίες μέσω διαδικτύου με ασφάλεια και αποδοτικότητα. Κάτι τέτοιο, θα

ήταν πολύ χρήσιμο και για την δικιά μας εφαρμογή ώστε να είναι διαθέσιμη σε ευρεία κλίμακα και να δίνεται σε όλους η δυνατότητα αξιοποίησης της.

Τέλος, μια σημαντική μελλοντική επέκταση που πρέπει να διερευνηθεί είναι η αξιοποίηση ασφαλών βάσεων δεδομένων εκτός του blockchain ή και η αξιοποίηση του IPFS πρωτοκόλλου για κατανεμημένη και ασφαλή αποθήκευση των δεδομένων. Πιο συγκεκριμένα, όπως προτάθηκε από τους Nikos Kapsoulis κ.α. στο ερευνητικό άρθρο «*Know Your Customer (KYC) Implementation with Smart Contracts on a Privacy-Oriented Decentralized Architecture*» το 2020 [2], θα μπορούσε να αξιοποιηθεί το αποκεντρωμένο πρωτόκολλο IPFS για ασφαλέστερη διαχείριση των προσωπικών δεδομένων. Αυτό καθίσταται εφικτό μέσω της δημιουργίας πολιτικών μιας διαδρομής, που στην δικιά μας περίπτωση θα αποτρέπουν την ανάγνωση των προσωπικών δεδομένων του ασθενούς από μη επικυρωμένες οντότητες, προσφέροντας έτσι μια συνδυαστική λύση στην διαχείριση δεδομένων στον τομέα της Υγείας.

Βιβλιογραφία

[1] C. V. N. U. B. Murthy, M. L. Shri, S. Kadry και S. Lim, «Blockchain Based Cloud Computing: Architecture and Research Challenges,» *IEEE Access,* τόμ. 8, pp. 205190-205205, 2020.

[2] N. Kapsoulis , A. Psychas , G. Palaiokrassas , A. Marinakis, A. Litke και T. Varvarigou, «Know Your Customer (KYC) Implementation with Smart Contracts on a Privacy-Oriented Decentralized Architecture,» *Future Internet,* 2020.

[3] A. Hasselgren, K. Kralevska, D. Gligoroski, S. A. Pedersen και A. Faxvaag, «Blockchain in healthcare and health sciences—A scoping review,» *International Journal of Medical Informatics,* τόμ. 134, 2020.

[4] A. A. Monrat, O. Schelén και K. Andersson, «A Survey of Blockchain From the Perspectives of Applications, Challenges, and Opportunities,» τόμ. 7, pp. 117134 - 117151, 19 August 2019.

[5] S. Nakamoto, «A Peer-to-Peer Electronic Cash System,» *Decentralized Business Review,* 2008.

[6] M. Wang, M. Duan και J. Zhu, «Research on the Security Criteria of Hash Functions in the Blockchain,» p. 9, 2018.

[7] S. Saraji, Blockchain and Sustainable Energy. In: Sustainable Oil and Gas Using Blockchain. Lecture Notes in Energy, Springer, Cham, 2023.

[8] Z. Zheng, S. Xie, H. Dai, X. Chen και H. Wang, «An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,» *2017 IEEE International Congress on Big Data (BigData Congress),* pp. 557-564, 2017.

[9] V. Buterin, «A next-generation smart contract and decentralized application platform,» *White Paper,* p. 37, 2014.

[10] Y. Liu, Y. Hei, T. Xu και J. Liu, «An Evaluation of Uncle Block Mechanism Effect on Ethereum Selfish and Stubborn Mining Combined With an Eclipse Attack,» *IEEE Access,* τόμ. 8, pp. 17489-17499, 2020.

[11] B. Wu, «Analysis of Ethereum Ghost Protocol under Blockchain Framework,» *Highlights in Science, Engineering and Technology 60,* pp. 121-127, 2023.

[12] S. S. Sarmah, «Understanding blockchain technology.,» *Computer Science and Engineering,* pp. 23-29, 2018.

[13] H. Chen, M. Pendleton, L. Njilla και S. Xu, «A Survey on Ethereum Systems Security: Vulnerabilities, Attacks, and Defenses,» *ACM Computing Surveys,* τόμ.

53, αρ. 3, pp. 1-43, 2020.

[14] C. Wang, H. Jiang, J. Zeng, M. Yu, Q. Huang και Z. Zuo, «A review of blockchain layered architecture and technology application research.,» *Wuhan University Journal of Natural Sciences,* τόμ. 26, αρ. 5, pp. 415-428, 2021.

[15] S. Mollajafari και K. Bechkoum, «Blockchain Technology and Related Security Risks: Towards a Seven-Layer Perspective and Taxonomy,» *Sustainability,* 2023.

[16] B. Lashkari και P. Musilek, «A Comprehensive Review of Blockchain,» *IEEE Access,* τόμ. 9, pp. 43620-43652, 2021.

[17] L. M. Bach, B. Mihaljevic και M. Zagar, «Comparative analysis of blockchain consensus algorithms,» *41st International Convention on Information and Communication Technology, Electronics and Microelectronics* , pp. 1545-1550, 2018.

[18] S. Zhou, K. Li, L. Xiao, J. Cai, W. Liang και A. Castiglione, «A Systematic Review of Consensus Mechanisms in Blockchain,» *Mathematics,* 2023.

[19] L. Lamport, R. Shostak και M. Pease, «The Byzantine Generals Problem,» *ACM Transcations on Programming Languages and Systems,* τόμ. 4, αρ. 3, pp. 382-461, 1982.

[20] K. Lei, Q. Zhang, L. Xu και Z. Qi, «Reputation-Based Byzantine Fault-Tolerance for Consortium Blockchain,» *IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS),* pp. 604-611, 2018.

[21] C. Dwork και M. Naor, «Pricing via processing or combatting junk mail,» *12th Annual International Cryptology Conference,* p. 139–147, 1992.

[22] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen και E. Dutkiewicz, «Proof-of-Stake Consensus Mechanisms for Future Blockchain Networks: Fundamentals, Applications and Opportunities,» *IEEE Access,* τόμ. 7, pp. 85727-85745, 2019.

[23] S. King και S. Nadal, «Ppcoin: Peer-to-peer crypto-currency with proof-of-stake,» *Self-Publ. Pap.,* 2012.

[24] M. Xie, J. Liu, S. Chen και M. Lin, «A survey on blockchain consensus mechanism: research overview, current advances and future directions,» *International Journal of Intelligent Computing and Cybernetics,* τόμ. 16, αρ. 2, pp. 314-340, 2023.

[25] S. AR και B. G. Banik, «A Comprehensive Study of Blockchain Services: Future of Cryptography,» *(IJACSA) International Journal of Advanced Computer Science and Applications,* τόμ. 11, αρ. 10, pp. 279-288, 2020.

[26] S. Zhai, Y. Yang, J. Li, C. Qiu και J. Zhao, «Research on the Application of Cryptography on the Blockchain,» *Journal of Physics: Conference Series,* τόμ. 1168, αρ. 3, 2019.

[27] M. Raikwar, D. Gligoroski και K. Kralevska, «SoK of Used Cryptography in Blockchain,» *IEEE Access,* τόμ. 7, pp. 148550-148575, 2019.

[28] Y. Bakos και H. Halaburda, «Permissioned vs Permissionless Blockchain Platforms: Tradeoffs in Trust and Performance,» *NYU Stern School of Business working paper,* 2021.

[29] P. K. Paul, P. S. Aithal, R. Saavedra και S. Ghosh, «Blockchain Technology and Its Types—A Short Review,» *International Journal of Applied Science and Engineering (IJASE),* pp. 189-200, 2021.

[30] Y. Tang, J. Yan, C. Chakraborty και Y. Sun, «edera: A Permissionless and Scalable Hybrid Blockchain Consensus Algorithm in Multiaccess Edge Computing for IoT,» *IEEE Internet of Things Journal,* τόμ. 10, αρ. 24, pp. 21187-21202, 2023.

[31] S. Suratkar, M. Shirole και S. Bhirud, «Cryptocurrency Wallet: A Review,» *4th International Conference on Computer, Communication and Signal Processing (ICCCSP),* pp. 1-7, 2020.

[32] S. Houy, P. Schmid και A. Bartek, «Security Aspects of Cryptocurrency Wallets—A Systematic Literature Review,» *ACM Computing Surveys,* pp. 1-31, 2023.

[33] E. Nowroozi, S. Seyedshoari, Y. Mekdad, E. Savaş και M. Conti, «Cryptocurrency Wallets: Assessment and Security,» σε *Blockchain for Cybersecurity in Cyber-Physical Systems. Advances in Information Security,* 2023.

[34] S. Jani, «An Overview of Ethereum & Its Comparison with Bitcoin,» *International Journal of Scientific & Engineering Research,* τόμ. 10, αρ. 8, 2018.

[35] Z. Wang, H. Jin, W. Dai, K.-K. R. Choo και D. Zou, «Ethereum smart contract security research: survey and future research opportunities,» *Frontiers of Computer Science 15,* 2021.

[36] T. Hu, X. Liu, T. Chen, X. Zhang, X. Huang, W. Niu, J. Lu, K. Zhou και Y. Liu, «Transaction-based classification and detection approach for Ethereum smart contract.,» *Information Processing & Management ,* τόμ. 58, αρ. 2, 2021.

[37] V. Dhillon, D. Metcalf και M. Hooper, Blockchain enabled applications, Berkeley: Apress, 2017.

[38] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur και H.-n. Lee, «Systematic Review of Security Vulnerabilities in Ethereum Blockchain Smart Contract,» *IEEE*

Access, τόμ. 10, pp. 6605-6621, 2022.

[39] S. N. Khan, F. Loukil, C. Ghedira-Guegan, E. Benkhelifa και A. Bani-Hani, «Blockchain smart contracts: Applications, challenges, and future trends,» *Peer-to-peer Networking and Applications,* τόμ. 14, pp. 2901-2925, 2021.

[40] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng και M. Imran, «An overview on smart contracts: Challenges, advances and platforms,» *Future Generation Computer Systems,* τόμ. 105, pp. 475-491, 2020.

[41] S. Aggarwal και N. Kumar, «Chapter Sixteen - Hyperledger,» σε *Advances in Computers,* τόμ. 121, Elsevier, 2021, pp. 323-343.

[42] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco και J. Yellick, «Hyperledger fabric: a distributed operating system for permissioned blockchains.,» *Proceedings of the thirteenth EuroSys conference.,* pp. 1-15, 2018.

[43] K. Marhane, F. Taif και A. Namir, «Secure Sharing of university Data Using Hyperledger Fabric and IPFS system,» *Procedia Computer Science,* τόμ. 224, pp. 163-168, 2023.

[44] X. Zhao, S. Wang, Y. Zhang και Y. Wang, «Attribute-based access control scheme for data sharing on hyperledger fabric,» *Journal of Information Security and Applications,* τόμ. 67, 2022.

[45] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu και A. V. Vasilakos, «Latency performance modeling and analysis for hyperledger fabric blockchain network,» *Information Processing & Management,* τόμ. 58, αρ. 1, 2021.

[46] S. Kakei, Y. Shiraishi, M. Mohri, T. Nakamura, M. Hashimoto και S. Saito, «Cross-Certification Towards Distributed Authentication Infrastructure: A Case of Hyperledger Fabric,» *IEEE Access,* τόμ. 8, pp. 135742-135757, 2020.

[47] L. Foschini, A. Gavagna, G. Martuscelli και R. Montanari, «Hyperledger Fabric Blockchain: Chaincode Performance Analysis,» *ICC 2020 - 2020 IEEE International Conference on Communications (ICC),* pp. 1-6, 2020.

[48] Y. Manevich, A. Barger και Y. Tock, «Service Discovery for Hyperledger Fabric,» *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems (DEBS '18),* p. 226–229, 2018.

[49] S. Zhang, E. Zhou, B. Pi, J. Sun, K. Yamashita και Y. Nomura, «A Solution for the Risk of Non-deterministic Transactions in Hyperledger Fabric,» *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC),* pp. 253-261, 2019.

[50] «Hyperledger Fabric Docs,» [Ηλεκτρονικό]. Available: https://hyperledger-fabric.readthedocs.io.

[51] S. Shalaby, A. A. Abdellatif, A. Al-Ali, A. Mohamed, A. Erbad και M. Guizani, «Performance Evaluation of Hyperledger Fabric,» *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT),* pp. 608-613, 2020.

[52] E. Androulaki, A. De Caro, M. Neugschwandtner και A. Sorniotti, «Endorsement in Hyperledger Fabric,» *2019 IEEE International Conference on Blockchain (Blockchain),* pp. 510-519, 2019.

[53] M. Soelman, V. Andrikopoulos, J. A. Pérez, V. Theodosiadis, K. Goense και A. Rutjes, «Hyperledger Fabric: Evaluating Endorsement Policy Strategies in Supply Chains,» *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS),* pp. 145-152, 2020.

[54] J. Sousa, A. Bessani και M. Vukolic, «A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform,» *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN),* pp. 51-58, 2018.

[55] M. Zand, X. B. Wu και M. A. Morris, Hands-On Smart Contract Development with Hyperledger Fabric V2, O'Reilly Media, Inc., 2021.

[56] G. Saldamli, C. Upadhyay, D. Jadhav, R. Shrishrimal και B. Patil, «Improved gossip protocol for blockchain applications,» *Cluster Computing 25,* p. 1915–1926 , 2022.

[57] N. Berendea, H. Mercier, E. Onica και E. Rivière, «Fair and Efficient Gossip in Hyperledger Fabric,» *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS),* pp. 190-200, 2020.

[58] H. H. Pajooh, M. Rashid, F. Alam και S. Demidenko, «Hyperledger Fabric Blockchain for Securing the Edge Internet of Things,» *Sensors 2021,* 2021.

[59] S. Wang, M. Yang, Y. Zhang, Y. Luo, T. Ge, X. Fu και W. Zhao, «On Private Data Collection of Hyperledger Fabric,» *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS),* pp. 819-829, 2021.

[60] «ΦΕΚ 5940/Β/21-11-2022,» [Ηλεκτρονικό]. Available: https://www.e-nomothesia.gr/kat-ygeia/kya-oik-65011-2022.html.

[61] «ΦΕΚ-6973 Β/11-12-23,» [Ηλεκτρονικό]. Available: https://istrikala.gr/wp-content/uploads/2023/12/6973b-23-%CE%A6%CE%95%CE%9A.pdf.

[62] docker.docs, «docs.docker.com,» [Ηλεκτρονικό]. Available: https://docs.docker.com/desktop/.

[63] «IBM,» [Ηλεκτρονικό]. Available: https://www.ibm.com/topics/docker.

[64] Docker, «docker.com,» [Ηλεκτρονικό]. Available: https://www.docker.com/products/docker-desktop/.

[65] «Hyperledger Fabric Docs,» [Ηλεκτρονικό]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.5/prereqs.html.

[66] «Hyperledger Fabric Docs,» [Ηλεκτρονικό]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.5/install.html.

[67] «Hyperledger Fabric Samples Github,» [Ηλεκτρονικό]. Available: https://github.com/hyperledger/fabric-samples.

[68] «Hyperledger Fabric Docs,» [Ηλεκτρονικό]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.5/install.html.

[69] «Hyperledger Fabric Docs,» [Ηλεκτρονικό]. Available: https://hyperledger-fabric.readthedocs.io/en/release-2.5/commands/configtxgen.html.