ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

ΜΗΧΑΝΙΣΜΟΣ ΒΕΛΤΙΣΤΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΠΡΟΣΩΠΙΚΩΝ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΤΟ ΔΙΑΔΙΚΤΥΟ ΤΩΝ ΠΡΑΓΜΑΤΩΝ

Σοφία Κατσικέα

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
Σπαθούλας Γέωργιος
μέλος Ε.ΔΙ.Π

Λαμία, 2025

Πίνακας περιεχομένων

1. ΕΙΣΑΓΩΓΗ....4
2. Σχετικές εργασίες....6
3. Background Technologies....7
3.1 Κρυπτογράφηση....7
3.1.1 Κρυπτογράφηση συμμετρικού κλειδιού....7
3.1.2 Κρυπτογράφηση βάσει χαρακτηριστικών....8
3.2 Διαπλανητικό Σύστημα Αρχείων (IPFS)....9
4. Περιγραφή συστήματος....10
4.1 Γενική περιγραφή συστήματος....10
4.1.1 Αλληλεπίδραση των οντοτήτων με το συμφωνητικό τύπου smart contract....11
....12
4.1.2 Αλληλεπίδραση Key Authority με Gateway και Service Providers....12
4.1.3 Περιγραφή του πρωτοκόλλου από της μεριάς του Gateway....14
4.1.4 Περιγραφή του πρωτοκόλλου από μεριάς του Service Provider....15
4.2 Τεχνική περιγραφή συστήματος....17
4.2.1 Βιβλιοθήκες που χρησιμοποιήθηκαν....18
4.2.2 Ανάλυση του κώδικα του Key Authority....19
4.2.3 Ανάλυση του κώδικα του Gateway....22
4.2.4 Ανάλυση του κώδικα του Service Provider....26
4.3 Μελέτη περιφερειακών τμημάτων κώδικα που απαιτήθηκαν για την ανάπτυξη του πρωτοκόλλου στην παρούσα πτυχιακή εργασία....31
5. Πειράματα....33
6. Συμπεράσματα....36
7. Βιβλιογραφία....37

Ευρετήριο πινάκων

Εικόνα 1: Αλληλεπίδραση οντοτήτων με το συμφωνητικό τύπου smart contract....12
Εικόνα 2....12
Εικόνα 3: Αλληλεπίδραση key authority με gateway και service provider....12
Εικόνα 4: Περιγραφή πρωτοκόλου από μεριάς του gateway....14
Εικόνα 5: Περιγραφή του πρωτοκόλλου από μεριάς του service provider....17
Εικόνα 6: Βιβλιοθήκες που χρησιμοποιήθηκαν στην υλοποίηση του gateway....18
Εικόνα 7: Βιβλιοθήκες που χρησιμοποιήθηκαν στην υλοποίηση του key authority....18

Εικόνα 8: Βιβλιοθήκες που χρησιμοποιήθηκαν στην υλοποίηση του service provider.....18
Εικόνα 9: Επιλογή παραμέτρων για τον κρυπτογραφικό αλγόριθμο.....19
Εικόνα 10: Δημιουργία δημοσίου κλειδιού και master key.....20
Εικόνα 11: Δημιουργία κλάσης για αποστολή κλειδιού.....20
Εικόνα 12: Δομή if-else.....20
Εικόνα 13: Ανάγνωση συμφωνητικού.....21
Εικόνα 14: Παράδειγμα συμφωνητικού.....22
Εικόνα 15: Ρύθμιση παραμέτρου για σύνδεση με ipfs.....22
Εικόνα 16: Ρύθμιση παραμέτρων κρυπτογραφικού αλγόριθμου και παραγωγή συμμετρικού κλειδιού.....23
Εικόνα 17: Ορισμός ονόματος αρχείου.....23
Εικόνα 18: Ανάγνωση δεδομένων από το αρχείο και κρυπτογράφηση με συμμετρικό κλειδί23
Εικόνα 19: Αίτηση για δημόσιο κλειδί και μετατροπή σε object.....24
Εικόνα 20: Κρυπτογράφηση συμμετρικού κλειδιού με attribute based encryption.....24
Εικόνα 21: Μετατροπή κρυπτογραφήματος σε bytes και αποθήκευση σε αρχείο.....24
Εικόνα 22: Μετατροπή κρυπτογραφημένου συμμετρικού κλειδιού σε bytes και αποθήκευση σε αρχείο.....25
Εικόνα 23: Προσθήκη αρχείου κρυπτογραφήματος στο IPFS.....25
Εικόνα 24: Αποθήκευση αρχείου κρυπτογραφημένου συμμετρικού κλειδιού στο IPFS.....25
Εικόνα 25: Ανανέωση index αρχείου.....25
Εικόνα 26: Εισαγωγή port σε συνάρτηση για σύνδεση με το IPFS.....26
Εικόνα 27: Ορισμός url.....26
Εικόνα 28: Ρύθμιση παραμέτρων κρυπτογραφικού αλγόριθμου.....26
Εικόνα 29: Ορισμός ονόματος του service provider και αίτηση για ιδιωτικό κλειδί.....26
Εικόνα 30: Λήψη κλειδιού και μετατροπή σε unicode.....27
Εικόνα 31: Ανάγνωση του CID του index αρχείου από το συμφωνητικό.....27
Εικόνα 32: Συνάρτηση για λήψη αρχείου από IPFS.....28
Εικόνα 33: Λήψη index αρχείου.....29
Εικόνα 34: Λήψη κρυπτογραφημένου συμμετρικού κλειδιού και αποκρυπτογράφηση του…29
Εικόνα 35: ΠΑραγωγή συμμετρικού κλειδιού από το αποκρυπτογραφημένο κρυπτογραφικό στοιχείο.....30
Εικόνα 36: Λήψη αρχείου με κρυπτογραφημένα δεδομένα.....30
Εικόνα 37: Key authority στο docker compose αρχείο.....31
Εικόνα 38: Service provider στο docker compose αρχείο.....31
Εικόνα 39: IPFS node στο docker compose αρχείο.....31
Εικόνα 40: Gateway στο docker compose αρχείο.....31
Εικόνα 41: 1 δειγματοληψία ανά λεπτό (gateway).....33
Εικόνα 42: 1 δειγματοληψία ανά λεπτό (service provider).....33
Εικόνα 43: 60 δειγματοληψίες (service provider).....33
Εικόνα 44: 60 δειγματοληψίες (gateway).....33
Εικόνα 45: 1440 δειγματοληψίες (gateway).....34
Εικόνα 46: 1440 δειγματοληψίες (service provider).....34
Εικόνα 47: 24 δειγματοληψίες (gateway).....34
Εικόνα 48: 24 δειγματοληψίες (service provider).....34

1. ΕΙΣΑΓΩΓΗ

Το Διαδίκτυο των Πραγμάτων (Internet of Things, IoT) αναφέρεται σε συσκευές, γνωστές ως “έξυπνες συσκευές”, με ενσωματωμένους αισθητήρες, λογισμικό, ικανότητα επεξεργασίας και τεχνολογία που τους επιτρέπει να συνδέονται και να ανταλλάσσουν δεδομένα με άλλες συσκευές και συστήματα μέσω του διαδικτύου ή άλλα συστήματα επικοινωνίας. Οι έξυπνες συσκευές μπορούν να βρεθούν πλέον στο χέρι ενός χρήστη, με την μορφή ενός “έξυπνου” ρολογιού, στο σπίτι του, ως ένας “έξυπνος” θερμοστάτης, μια “έξυπνη” καφετέρια ή και συνθέτοντας το σύστημα ασφαλείας του σπιτιού, στο αυτοκίνητο με την μορφή μιας “έξυπνης” κάμερας ή ακόμα και στο δωμάτιο του νοσοκομείου ως “έξυπνο” κρεβάτι. Και αυτά τα παραδείγματα δεν περιέχουν καν την χρήση του Διαδικτύου των Πραγμάτων στην βιομηχανική, αγροτική και κτηνοτροφική παραγωγή, στον σχεδιασμό “έξυπνων” πόλεων, στις μεταφορές και σε πολλούς άλλους τομείς των σύγχρονων κοινωνιών.

Ο όγκος των δεδομένων που παράγονται από όλες αυτές τις συσκευές είναι τεράστιος και η κυριότητα και η ασφάλεια των δεδομένων έχει υπάρξει κεντρικό θέμα σε πολλές συζητήσεις και προβληματισμούς τόσο ηθικής, αλλά και νομικής και οικονομικής φύσεως. Η συλλογή των δεδομένων, εξάλλου, από ένα “έξυπνο” σπίτι και ένα “έξυπνο” αυτοκίνητο, δίνει την ευκαιρία σε αυτόν που έχει πρόσβαση σε αυτή να συνθέσει μια αρκετά λεπτομερή εικόνα για την καθημερινότητα και την ζωή του χρήστη αυτών των συσκευών.

Και ενώ πλέον οι “έξυπνες” συσκευές αποτελούν μέρος της καθημερινότητας πολλών πολιτών, υπάρχουν ακόμα πολλοί προβληματισμοί ως προς το πως διασφαλίζεται η ασφάλεια των χρηστών και η αποφυγή εκμετάλλευσης των δεδομένων τους με τρόπο που οι ίδιοι δεν έχουν συναινέσει. Οι χρήστες “έξυπνων” συσκευών εξαρτώνται από την νομοθεσία της εκάστοτε χώρας για να διασφαλίσουν την ιδιωτικότητα τους και την κυριότητα των δεδομένων τους, αλλά ακόμα και με νομοθετικούς κανονισμούς όπως ο Γενικός Κανονισμός για την Προστασία των Δεδομένων (GDPR), που αν και αποτέλεσε σημαντικό βήμα δεν δίνει των πλήρη έλεγχο των προσωπικών δεδομένων στους χρήστες ούτε διασφαλίζει την ανωνυμία και ιδιωτικότητα τους, οι χρήστες πρέπει να εμπιστευτούν ότι ο νόμος διασφαλίζει την ιδιωτικότητα και την ασφάλεια τους, ότι οι πάροχοι υπηρεσιών, αλλά και οι παραγωγοί των συσκευών, με τους οποίους συνεργάζονται τηρούν όλα τα μέτρα που επιβάλλουν οι νόμοι, πως δεν υπάρχουν κενά ασφαλείας και πως δεν έχουν συμφωνήσει οι ίδιοι σε κάτι που δεν θα είχαν συναινέσει αν μελετούσαν εκτενώς και κατανοούσαν σε βάθος του Όρους και τις Προϋποθέσεις της κάθε συσκευής και της κάθε υπηρεσίας, αλλά και του ισχύοντος νομοθετικού πλαισίου.

Η παρούσα πτυχιακή εργασία αναπτύχθηκε βάσει της ιδέας ότι ,πόσο μάλλον στην ψηφιακή εποχή, η προστασία της ιδιωτικότητας οφείλει να καλύπτεται και να διασφαλίζεται όχι μόνο από νομοθετικά πλαίσια, αλλά να αποτελεί δομικό στοιχείο του τρόπου σχεδιασμού και λειτουργίας του Διαδικτύου των Πραγμάτων και προτείνει ένα πρωτόκολλο αλληλεπίδρασης ανάμεσα σε χρήστες IoT συσκευών και παρόχων υπηρεσιών και επεξεργασίας δεδομένων που βασίζεται στην αρχή της κυριότητας των δεδομένων από τον χρήστη της

“έξυπνης” συσκευής και στην προστασία των δεδομένων ,και κατ’επέκτασης του χρήστη, μέσω κρυπτογράφησης και αποκεντρωμένης αποθήκευσης τους.

2. Σχετικές εργασίες

Η ιδιωτικότητα και η ασφάλεια των δεδομένων που παράγονται από IoT συσκευές αποτελεί θέμα πάνω στο οποίο βασίζονται αρκετές εργασίες και έρευνες τα τελευταία χρόνια. Η χρήση κρυπτογραφικών μεθόδων και του Διαπλανητικού Συστήματος Αρχείων (IPFS) συμπεριλαμβάνεται σε αρκετές εργασίες, χωρίς αυτό να σημαίνει πως όλα τα πρωτόκολλα και οι μέθοδοι που αναπτύσσονται είναι πανομοιότυπα. Κάθε κρυπτογραφική μέθοδος μπορεί να παρέχει διαφορετικές ιδιότητες σε ένα πρωτόκολλο και ο τρόπος χρήσης του IPFS μπορεί, επίσης, να διαφέρει.

Στο άρθρο [1] αναπτύσσεται ένα σύστημα κατηγορίας ιατροπερίθαλψης με έμφαση στην παρακολούθηση χρόνιων ασθενειών μέσω από δεδομένα που προέρχονται από IoT συσκευές. Η αποθήκευση των δεδομένων γίνεται στο IPFS και με την χρήση του blockchain και των smart contracts υπάρχει διασφάλιση της ιδιωτικότητας και την επιβεβαίωση της ταυτότητας των ατόμων που έχουν πρόσβαση στα δεδομένα. Η πρόσβαση, όμως, δεν δίνεται βάσει κάποιου χρονικού περιθωρίου και η κρυπτογράφηση γίνεται μέσω ενός proxy και όχι στην μεριά του ίδιου του χρήστη, επιπλέον χρησιμοποιείται μόνο κρυπτογράφηση δημοσίου κλειδιού για την κρυπτογράφηση των δεδομένων και δίνεται μεγαλύτερη έμφαση στην λειτουργία του blockchain για την διαφύλαξη της ιδιωτικότητας.

Στο άρθρο [2] αναλύεται ένα πρωτόκολλο για διαχείριση δεδομένων από IoT συσκευές με την χρήση του IPFS και smart contracts, αλλά δεν υπάρχει ένα συμβόλαιο με τον αιτούντα πρόσβαση στα δεδομένα, αντιθέτως η οντότητα που θέλει να αποκτήσει πρόσβαση, την αιτείται απευθείας από τον κάτοχο των δεδομένων και οι “διευθύνσεις” (CID) των αρχείων που είναι αποθηκευμένα στο IPFS αποθηκεύονται όλες στο smart contract.

Στο άρθρο [3], [4] και [5] περιγράφονται πρωτόκολλα διασφάλισης των δεδομένων από IoT συσκευές με την χρήση τόσο κρυπτογράφησης, του IPFS και του blockchain, αλλά δεν γίνεται ανάλυση της υλοποίησης των πρωτοκόλλων και δίνεται μεγαλύτερη έμφαση στην χρήση του blockchain.

3. Background Technologies

Για την ανάπτυξη του πρωτοκόλλου αλληλεπίδρασης που προτείνεται σε αυτή την πτυχιακή εργασία χρησιμοποιήθηκαν δύο διαφορετικά είδη κρυπτογράφησης και ένα ομότιμο δίκτυο (Peer-to-Peer ή P2P) σχεδιασμένο για την αποθήκευση και διαμοιρασμό υπερμέσων σε ένα κατανεμημένο σύστημα αρχείων.

3.1 Κρυπτογράφηση

Τα δύο είδη κρυπτογραφίας που χρησιμοποιήθηκαν είναι η κρυπτογράφηση συμμετρικου κλειδιού (symmetric encryption) και η κρυπτογράφηση βάσει χαρακτηριστικών (Attribute Based Encryption ή ABE). Για να γίνουν κατανοητές οι δυο διαφορετικές μέθοδοι κρυπτογράφησης θα αναλυθούν σε δύο διαφορετικά υποκεφάλαια, ξεκινόντας με την κρυπτογράφηση συμμετρικού κλειδιού.

3.1.1 Κρυπτογράφηση συμμετρικού κλειδιού

Η κρυπτογράφηση συμμετρικού κλειδιού ή αλλιώς συμμετρική κρυπτογράφηση [3][4] είναι μια διαδικασία που χρησιμοποιεί ένα κρυπτογραφικό αλγόριθμο στον οποίον τόσο η κρυπτογράφηση όσο και η αποκρυπτογράφηση χρησιμοποιούν ένα και μόνο ένα κλειδί. Ένα πολύ απλό παράδειγμα για να γίνει κατανοητός ο παραπάνω αλγόριθμος είναι το εξής: δύο φίλοι θέλουν να ανταλλάξουν ένα μήνυμα μέσα στην τάξη, αλλά για παραδοθεί το μήνυμα θα πρέπει να περάσει από τα χέρια διάφορων συμμαθητών τους. Για να διατηρήσουν τα μηνύματά τους μυστικά, οι δύο φίλοι έχουν προμηθευτεί ένα κουτάκι με κλειδαριά και δύο ίδια κλειδιά. Κάθε φίλος έχει κρατήσει από ένα κλειδί το οποίο χρησιμοποιεί για να μπορέσει να ανοίξει το κουτί και να διαβάσει το μήνυμα. Σε αυτό το παράδειγμα, τα κλειδιά αντιπροσωπεύουν, φυσικά, το κοινό κρυπτογραφικό κλειδί που μοιράζονται ο αποστολέας και ο παραλήπτης και το κουτί αντιπροσωπεύει το κρυπτογραφημένο κείμενο, δηλαδή το αποτέλεσμα της κρυπτογραφικής διαδικασίας.

Από το παράδειγμα και μόνο γίνονται φανερά κάποια από τα προβλήματα ασφάλειας. Αρχικά, το κρυπτογραφικό κλειδί πρέπει να είναι γνωστό και στα δύο μέλη της συναλλαγής. Και ενώ στην παραπάνω περίπτωση των δύο φίλων ήταν εύκολο να γίνει ο διαμοιρασμός με μία συνάντηση, στην περίπτωση του διαδικτύου είναι πιο δύσκολο να γίνει μια τέτοια ανταλλαγή με ασφαλή τρόπο. Οποιαδήποτε υποκλοπή του συμμετρικού κλειδιού σημαίνει ότι κάποιος θα μπορούσε να αποκρυπτογραφήσει και να παραποιήσει τα δεδομένα που αποστέλλονται χωρίς καμία γνώση τόσο από τον αποστολέα όσο και από τον παραλήπτη. Συχνή λύση αποτελεί η χρήση ασύμμετρης κρυπτογράφησης για την ασφαλή αποστολή του συμμετρικού κλειδιού.

Στα θετικά γνωρίσματα, όμως, της συμμετρικής κρυπτογράφησης συμπεριλαμβάνονται ότι δεν καταναλώνει μεγάλη υπολογιστική ισχύ και επομένως είναι προτιμότερη για την κρυπτογράφηση μεγάλου όγκου δεδομένων και πως οι περισσότεροι αλγόριθμοι που χρησιμοποιούνται πλέον φαίνεται να

είναι, με την χρήση μεγαλύτερων κλειδιών, κβαντο-ανθεκτικοί, να αντέχουν δηλαδή σε κρυπτοαναλυτικές επιθέσεις από κβαντικούς υπολογιστές.

Υπάρχουν κυρίως δύο είδη συμμετρικής κρυπτογράφησης, οι κρυπτογραφικοί αλγόριθμοι ροής (stream ciphers), που κρυπτογραφούν ένα bit ή byte την φορά, κάτι που τους κάνει ιδανικούς για real-time επικοινωνία, και οι κρυπτογραφικοί αλγόριθμοι δέσμης (block cipher), κατά τους οποίους το αρχικό κείμενο κρυπτογραφείται σε τμήματα σταθερού μεγέθους, διασφαλίζοντας έτσι την ακεραιότητα και την ασφάλεια μεγάλου όγκου δεδομένων. Στο παρόν πρωτόκολλο χρησιμοποιείται ο αλγόριθμος AES (Advanced Encryption Standard) που αποτελεί κρυπτογραφικό αλγόριθμο δέσμης.

3.1.2 Κρυπτογράφηση βάσει χαρακτηριστικών

Η κρυπτογράφηση βάσει χαρακτριστικών (Attribute Based Encryption) είναι ένας τύπος κρυπτογράφησης ασύμμετρου κλειδιού. Στην ασύμμετρη κρυπτογράφηση, ή αλλιώς κρυπτογράφηση δημοσίου κλειδιού, σε αντίθεση με την συμμετρική κρυπτογρ/άφηση, ο χρήστης διαθέτει δύο κλειδιά, ένα ιδιωτικό και ένα δημόσιο, τα οποία συνδέονται μεταξύ τους με μια μαθηματική σχέση. Ο χρήστης κρυπτογραφεί τα δεδομένα του με το ιδιωτικό κλειδί, το οποίο πρέπει να κρατήσει ασφαλές και μυστικό, ενώ η αποκρυπτογράφηση γίνεται με το δημόσιο κλειδί, το οποίο μπορεί να διαμοιραστεί ελεύθερα χωρίς να διακινδυνεύεται η ασφάλεια της κρυπτογράφησης. [5] Στην κρυπτογράφηση με βάσει την ταυτότητα, ένα άλλο είδος κρυπτογράφησης ασύμμετρου κλειδιού, το δημόσιο κλειδί παράγεται από ένα μοναδικό χαρακτηριστικό (ταυτότητα) του χρήστη του δημοσίου κλειδιού. [6]Η κρυπτογράφηση βάσει χαρακτηριστικών γενικεύει την πρόσβαση στο δημόσιο κλειδί, παράγοντας το, όχι από την ταυτότητα του χρήστη, αλλά βάσει ενός συνόλου χαρακτηριστικών, τα οποία λειτουργούν ως προϋπόθεση για να υπάρξει πρόσβαση στα αποκρυπτογραφημένα δεδομένα.

Υπάρχουν κυρίως δύο είδη αυτής της κρυπτογράφησης, η ciphertext-policy κρυπτογράφηση (CP-ABE) και η key-policy κρυπτογράφηση (KP-ABE).

Στην CP-ABE [7] το ιδιωτικό κλειδί ενός χρήστη σχετίζεται με ένα σύνολο χαρακτηριστικών και το κρυπτογράφημα (ciphertext) έχει ενσωματωμένο ένα κανόνα πρόσβασης (access policy), ο οποίος βασίζεται στην λογική σχέση ανάμεσα σε διαφορετικά χαρακτηριστικά ενός δεδομένου συνόλου. Επομένως, ο χρήστης θα μπορέσει να αποκρυπτογραφήσει τα κρυπτογράφημα με το ιδιωτικό του κλειδί, αν και μόνο αν, το σύνολο των χαρακτηριστικών που διαθέτει ικανοποιεί την λογική σχέση του κανόνα πρόσβασης. [9]

Αντιθέτως, στην KP-ABE ο κανόνας πρόσβασης, που καθορίζει τα προνόμια πρόσβασης του χρήστη, ενσωματώνεται στο ιδιωτικό κλειδί του και τα δεδομένα κρυπτογραφούνται βάσει ενός συνόλου χαρακτηριστικών. Αν το σύνολο αυτών των χαρακτηριστικών δύναται να συνθέσει τον κανόνα πρόσβασης τότε η αποκρυπτογράφηση των δεδομένων από τον κάτοχο του ιδιωτικού κλειδιού καθίσταται δυνατή. Στο πρωτόκολλο που προτείνεται χρησιμοποιείται ένας KP-ABE αλγόριθμος. [8]

Στα πλεονεκτήματα της κρυπτογράφησης βάσει χαρακτηριστικών είναι ο μεγαλύτερος και λεπτομερής έλεγχος που παρέχει σε σχέση με το ποιος αποκτά την δυνατότητα αποκρυπτογράφησης των δεδομένων, η δυνατότητα για σύνθετους κανόνες πρόσβασης καθώς και ο δυναμικός έλεγχος της πρόσβασης,

για παράδειγμα αν σε κάποιον χρήστη αφαιρεθεί κάποιο χαρακτηριστικό και πλέον δεν ικανοποιείται ο κανόνας πρόσβασης, τότε αυτός ο χρήστης δεν μπορεί πλέον να αποκρυπτογραφήσει τα ίδια δεδομένα με πριν, χωρίς να χρειαστεί να ξαναγίνει κρυπτογράφηση αυτών των δεδομένων.
Στα μειονεκτήματα αυτής της κρυπτογράφησης καταλογίζονται ότι η γνωστοποίηση του συνόλου των χαρακτηριστικών μπορεί να διακινδυνεύσει την ιδιωτικότητα των δεδομένων, πως σε σύγκριση με παραδοσιακούς τύπους κρυπτογράφησης στην περίπτωση της κρυπτογράφησης βάσει χαρακτηριστικών υπάρχει άυξηση κατανάλωσης υπολογιστικών πόρων και άρα μείωση της ταχύτητας της διαδικασίας και τέλος, το πρόβλημα της μεσεγγύσης κλειδιού (key escrow) που απαιτεί την ανάγκη εμπιστοσύνης σε μια κεντρική αρχή που παράγει τα κλειδιά όλων των χρηστών.

3.2 Διαπλανητικό Σύστημα Αρχείων (IPFS)

Το Διαπλανητικό Σύστημα Αρχείων (InterPlanetary File System ή IPFS) είναι ένα πρωτόκολλο για την δημιουργία ενός ομότιμου δικτύου με στόχο την αποθήκευση και την κοινή χρήση υπερμέσων σε ένα κατανεμημένο σύστημα. Το IPFS αναφέρεται επίσης σε ένα ανοιχτό και συμμετοχικό αποκεντρωμένο δίκτυο που αποτελείται από κόμβους. Οι κόμβοι αυτοί αποτελούν διασυνδεδεμένους υπολογιστές που τρέχουν λογισμικό IPFS και που χρησιμοποιούν κατανεμημένους πίνακες κατακερματισμού (Distributed Hash Table ή DHT)[10], οι οποίοι είναι ουσιαστικά ένα σύστημα αναζήτησης στο οποίο κάθε κόμβος είναι υπεύθυνος για ένα κλειδί το οποίο αντιστοιχίζεται με μια δεδομένη τιμή. Οι κόμβοι, επομένως, αποθηκεύουν δεδομένα και όταν υπάρχει ένα αίτημα για αυτά, τότε ένας κόμβος ή και πολλοί κόμβοι είναι υπεύθυνοι για την παράδοση τους. Αν υπάρχουν πολλά αιτήματα για τα ίδια δεδομένα τότε αυτά αποθηκεύονται σε πολλούς κόμβους.

Στο Διαπλανητικό Σύστημα Αρχείων τα δεδομένα διευθυνσιοδοτούνται βάσει του περιερχομένου (content addressed) τους και όχι βάσει της τοποθεσίας (location addressed) τους. Τα δεδομένα εισάγονται ως είσοδος σε μια κρυπτογραφική συνάρτηση κατακερματισμού (cryptographic hash function) η οποία δίνει ως έξοδο μια “ταυτότητα περιεχομένου” (Content Identifies ή CID) που αποτελεί μοναδική ταυτοποίηση των δεδομένων ως έχουν. Αν αλλάξει κάτι στα δεδομένα, τότε θα εκδοθεί ένα νέο, εντελώς διαφορετικό CID, κάτι που αποκαλείται ιδιότητα της αμεταβλητότητας. Αν τα ίδια δεδομένα προστεθούν από δύο κόμβους με τις ίδιες παραμέτρους τότε θα παράξουν το ίδιο CID. Αναλόγως με το μέγεθος των δεδομένων, το IPFS μπορεί να τα διαμοιράσει σε πολλαπλά τμήματα, καθέ ένα από τα οποία θα έχει το δικό του CID, διατηρώντας όμως ταυτόχρονα το αρχικό CID. Με την χρήση των CIDs, επομένως, γίνεται η αναζήτηση ενός υπερμέσου στο IPFS. Αν ο κόμβος που ερωτηθεί δεν έχει το αρχείο που αντιστοιχεί στο CID που δόθηκε, τότε θα βρει έναν άλλο προσβάσιμο κόμβο που έχει το αρχείο στο ομότιμο δίκτυο και θα το παραλάβει από αυτόν με σκοπό να το παραδώσει στον αιτούντα.

4. Περιγραφή συστήματος

4.1 Γενική περιγραφή συστήματος

Η ιδέα που αναπτύσσεται στην παρούσα πτυχιακή εργασία είναι η εξής:

Ο χρήστης έχει μια IoT συσκευή που συλλέγει δεδομένα, τα οποία στην συνέχεια κρυπτογραφούνται και προστίθενται στο IPFS. Ο χρήστης επιλέγει έναν πάροχο υπηρεσιών και συνάπτει μια συμφωνία μαζί του με την οποία συναινεί να του παρέχει πρόσβαση σε ένα ή και περισσότερα είδη δεδομένων για ένα συγκεκριμένο μόνο χρονικό διάστημα. Ο πάροχος υπηρεσιών λαμβάνει τα κρυπτογραφημένα δεδομένα από το IPFS και αν του έχει επιτραπεί η πρόσβαση μπορεί να τα αποκρυπτογραφήσει.

Υπάρχουν τρεις οντότητες οι οποίες χρησιμοποιούνται στην περιγραφή του πρωτοκόλλου αλληλεπίδρασης που προτείνεται:

1) **Gateway** : αντιπροσωπεύει τον κάτοχο-χρήστη IoT συσκευών που θέλει να μοιραστεί τα δεδομένα που συλλέγουν οι συσκευές του με κάποιον πάροχο επεξεργασίας δεδομένων.

2) **Key Authority** : αποτελεί μια έμπιστη οντότητα διαχείρισης κλειδιών κρυπτογράφησης. Κάθε gateway αντιστοιχεί και σε ένα key authority.

3) **Service Provider** : αντιπροσωπεύει τους παρόχους επεξεργασίας δεδομένων. Ένα gateway μπορεί να μοιράζεται δεδομένα με έναν, πολλούς ή και με κανέναν πάροχο.

Επιπλέον, η λειτουργία του πρωτοκόλλου βασίζεται σε δύο ακόμα βασικά στοιχεία:
- ένα κατανεμημένο σύστημα αποθήκευσης αρχείων, όπως το Διαπλανητικό Σύστημα Αρχείων (InterPlanetary File System, IPFS) καθώς και
- ένα πρωτόκολλο συναλλαγών, όπως τα smart contracts.

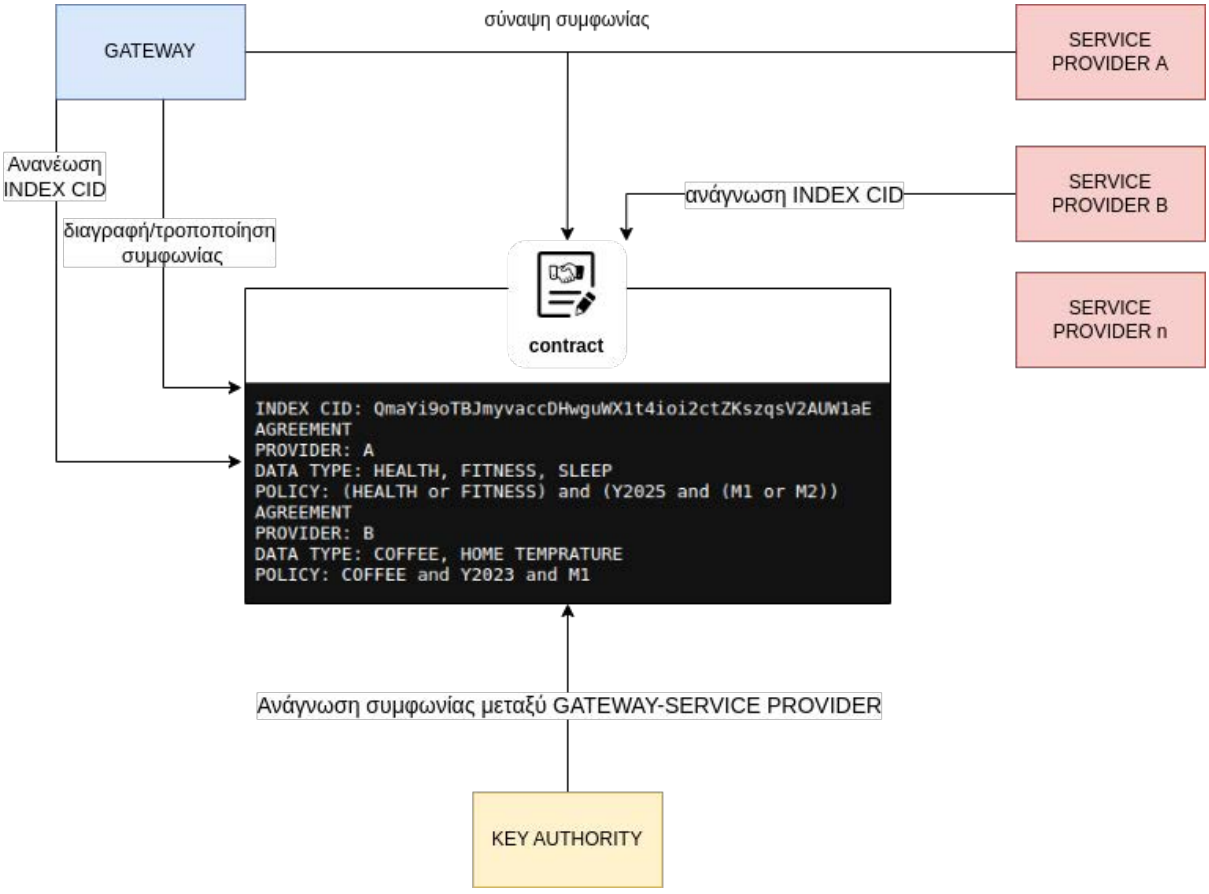
Το πρωτόκολλο που περιγράφεται βασίζεται στην δημιουργία ενός συμφωνητικού, το οποίο έχει τις εξής λειτουργικότητες:
- δίνει στον χρήστη την δυνατότητα να συνάψει μια συμφωνία με έναν πάροχο, δίνοντας έτσι πρόσβαση στα είδη δεδομένων του που επιθυμεί για όποιο χρονικό διάστημα αποφασίσει.
- δίνει στον χρήστη την δυνατότητα να τερματίσει την συμφωνία, άρα και την πρόσβαση στα δεδομένα, την χρονική στιγμή που αυτός αποφασίσει
- κάνει εύκολα προσβάσιμο το CID αναγνωριστικό ενός αρχείου στο IPFS που περιέχει τα αναγνωριστικά όλων των αρχείων με δεδομένα που έχουν προστεθεί στο IPFS.

Για να γίνει κατανοητό το πρωτόκολλο θα αναλυθεί στα εξής υποκεφάλαια:

- Αλληλεπίδραση των οντοτήτων με το συμφωνητικό τύπου smart contract
- Αλληλεπίδραση του Key Authority με το Gateway και τους Service Providers
- Περιγραφή του πρωτοκόλλου απο μεριάς του Gateway
- Περιγραφή του πρωτοκόλλου από μεριάς των Service Providers

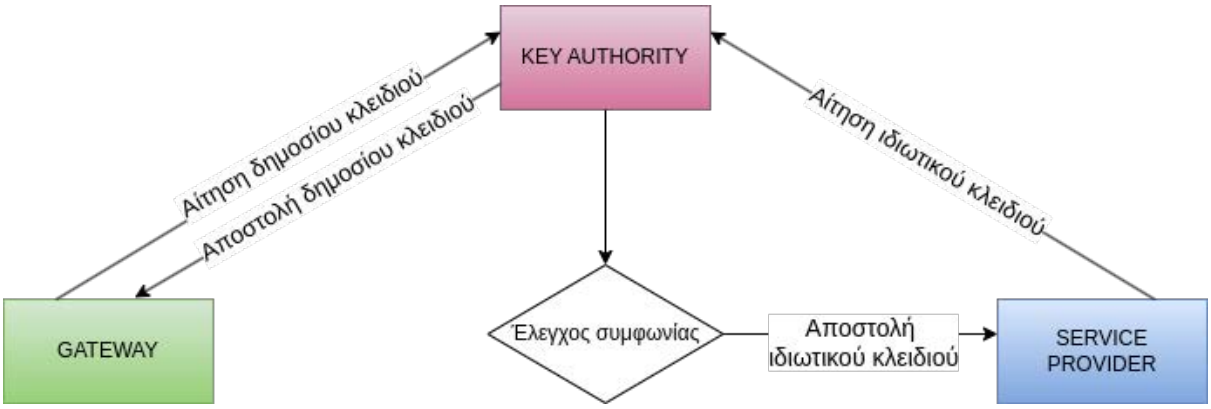
4.1.1 Αλληλεπίδραση των οντοτήτων με το συμφωνητικό τύπου smart contract

Όπως φαίνεται και στην εικόνα 1, το Gateway αλληλεπιδρά με το contract είτε για να συνάψει κάποια συμφωνία με κάποιον πάροχο είτε για να τροποποιήσει (ή και να τερματίσει) μια ήδη υπάρχουσα συμφωνία ή τέλος για να ανανεώσει το CID αναγνωριστικό για το index αρχείο.
Οι service providers αλληλεπιδρούν με το συμφωνητικό είτε για να συνάψουν κάποια συμφωνία με τον χρήστη ή για να λάβουν το CID αναγνωριστικό. Τέλος, το Key Authority αλληλεπιδρά με το συμφωνητικό, μόνο και μόνο, για να αναγνώσει τις συμφωνίες που έχουν γίνει με κάθε πάροχο.



Εικόνα 1: Αλληλεπίδραση οντοτήτων με το συμφωνητικό τύπου smart contract

4.1.2 Αλληλεπίδραση Key Authority με Gateway και Service Providers



Εικόνα 3: Αλληλεπίδραση key authority με gateway και service provider

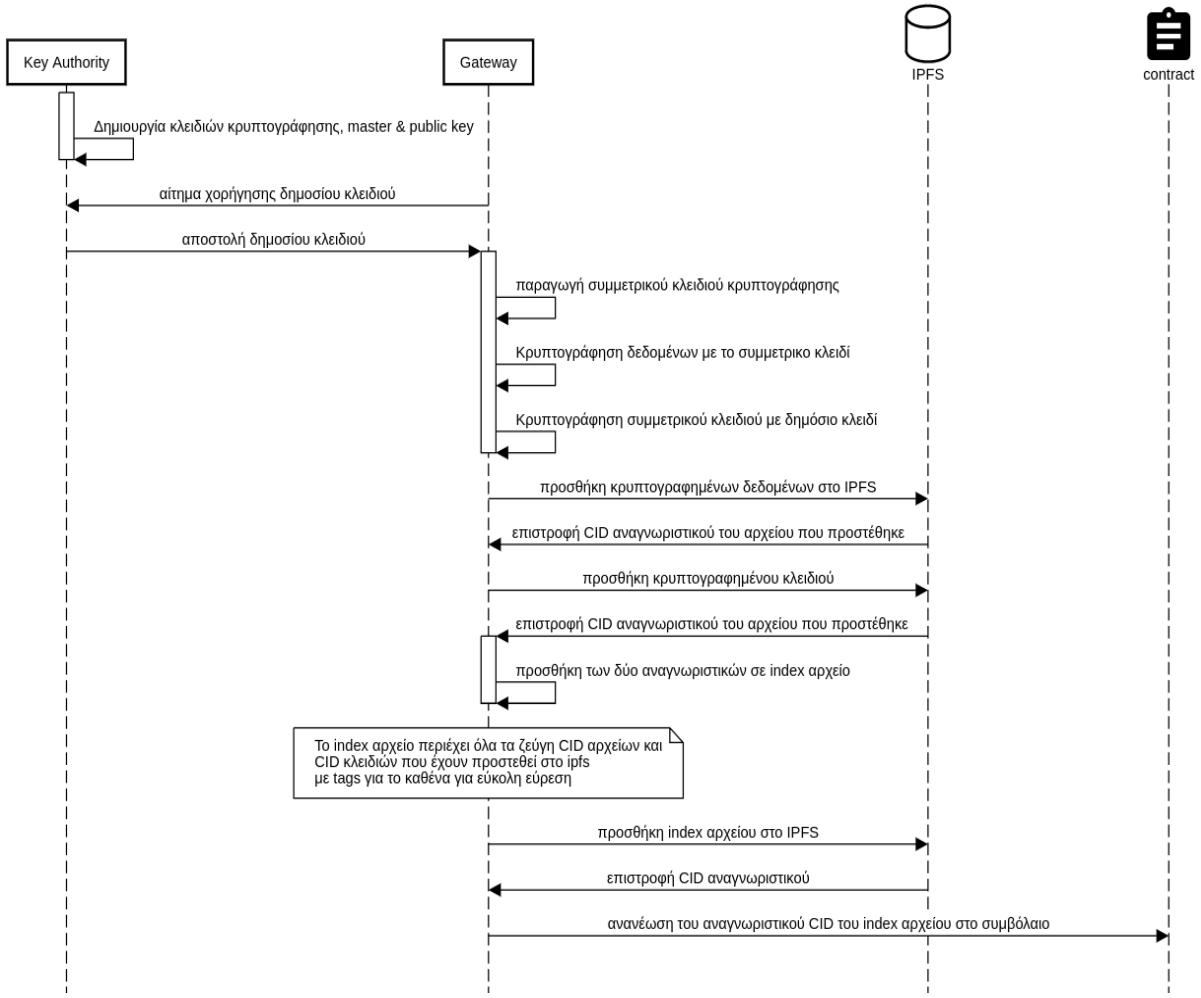
Η οντότητα Key Authority είναι αυτή η οποία παράγει ένα ζεύγος κλειδιών αποκρυπτογράφησης, δηλαδή ένα δημόσιο (public) και ένα ιδιωτικό (secret) κλειδί.

Μέτα από αίτημα του Gateway, το δημόσιο κλειδί αποστέλλεται από το Key Authority σε αυτό, ενώ για να σταλεί το ιδιωτικό κλειδί σε κάποιον service provider μετά από αίτημα του, πρέπει πρώτα να γίνει ανάγνωση από το Key Authority της συμφωνίας που έχει συνάψει αυτός με τον χρήστη, καθώς το ιδιωτικό κλειδί παράγεται βάσει της συμφωνίας αυτής.

4.1.3 Περιγραφή του πρωτοκόλλου από της μεριάς του Gateway.

Η κρυπτογράφηση που περιγράφει το πρωτόκολλο λαμβάνει χώρα στο gateway και μπορεί να χωριστεί σε δύο μέρη:

α) στην κρυπτογράφηση των δεδομένων με ένα συμμετρικό κλειδί που παράγεται στο gateway
β) στην κρυπτογράφηση του συμμετρικού κλειδιού με το δημόσιο κλειδί που αποστέλλεται από το Key Authority



Εικόνα 4: Περιγραφή πρωτοκόλου από μεριάς του gateway

Η διαδικασία ξεκινάει με την λήψη του δημοσίου κλειδιού από το gateway, το οποίο στην συνέχεια παράγει ένα συμμετρικό κλειδί με το οποίο κρυπτογραφεί τα δεδομένα που προέρχονται από IoT συσκευές. Στην συνέχεια κρυπτογραφεί το συμμετρικό κλειδί με το δημόσιο κλειδί που έλαβε από το key authority.

Έπειτα, σειρά έχει η προσθήκη του αρχείου με τα κρυπτογραφημένα δεδομένα, αλλά και το αρχείο με το κρυπτογραφημένο κλειδί στο IPFS, το οποίο επιστρέφει δυο CIDs (Content Identifiers), με τα οποία μπορεί να γίνει η αναζήτηση των αρχείων αυτών.

Προτού περιγραφεί το τελικό στάδιο, πρέπει να γίνει μια ανάλυση της δομής του index αρχείου που αναφέρεται στα παραπάνω διαγράμματα. Το index αρχείο είναι ένα αρχείο που περιέχει τα ζεύγη CIDs των αρχείων με τα

κρυπτογραφημένα δεδομένα και το κρυπτογραφημένο κλειδί που έχουν προστεθεί στο IPFS από το gateway. Κάθε ζεύγος συνοδεύεται από την ημερομηνία παραγωγής των δεδομένων, αλλά και το είδος κατηγορίας στην οποίαν ανήκουν τα δεδομένα αυτά (πχ Health data).

Κατα το τελικό στάδιο, το gateway προσθέτει στο index αρχείο το τελευταίο ζεύγος CIDs από τα αρχεία που μόλις πρόσθεσε στο IPFS, μαζί με τις απαραίτητες πληροφορίες που αναφέρθηκαν στην πάνω παράγραφο, και προσθέτει το index αρχείο στο IPFS. Τέλος, έχοντας λάβει το CID του index αρχείου, αποκτά πρόσβαση στο συμβόλαιο και ανανεώνει την τιμή του CID του index αρχείου σε αυτό.

4.1.4 Περιγραφή του πρωτοκόλλου από μεριάς του Service Provider

Αρχικά ο Service Provider κάνει αίτηση στο Key Authority για να του αποσταλεί ένα ιδιωτικό κλειδί. Το key authority αφού διαβάσει την συμφωνία που έχουν συνάψει ο χρήστης και ο service provider δημιουργεί το ιδιωτικό κλειδί βάσει αυτής. Στην συνέχεια ο πάροχος διαβάζει από το συμφωνητικό (contract) το CID του index αρχείου και το αναζητά με αυτό στο IPFS. Αφού λάβει το index αρχείο αναζητά σε αυτό, βάσει κατηγορίας δεδομένων και αναγραφόμενης ημερομηνίας, το Content Identifier (CID) του αρχείου με τα κρυπτογραφημένα δεδομένα και το CID του αρχείου με το κρυπτογραφημένο συμμετρικό κλειδί που του αντιστοιχεί. Αφού τα εντοπίσει, αναζητά με την χρήση αυτών τα αντίστοιχα αρχεία στο IPFS και τα “κατεβάζει”.

Έχοντας λάβει το ιδιωτικό κλειδί από το key authority, αποκρυπτογραφεί το συμμετρικό κλειδί και με το ,αποκρυπτογραφημένο πλέον, κλειδί αποκρυπτογραφεί τα δεδομένα.

Σε περίπτωση που δεν υπάρχει συμφωνία ανάμεσα στον πάροχο και τον κάτοχο των δεδομένων, ή αν η συμφωνία έχει λήξει, τότε το Key Authority ενημερώνει τον πάροχο ότι δεν έχει πρόσβαση πλέον στα δεδομένα και δεν του αποστέλλεται κάποιο κλειδί.

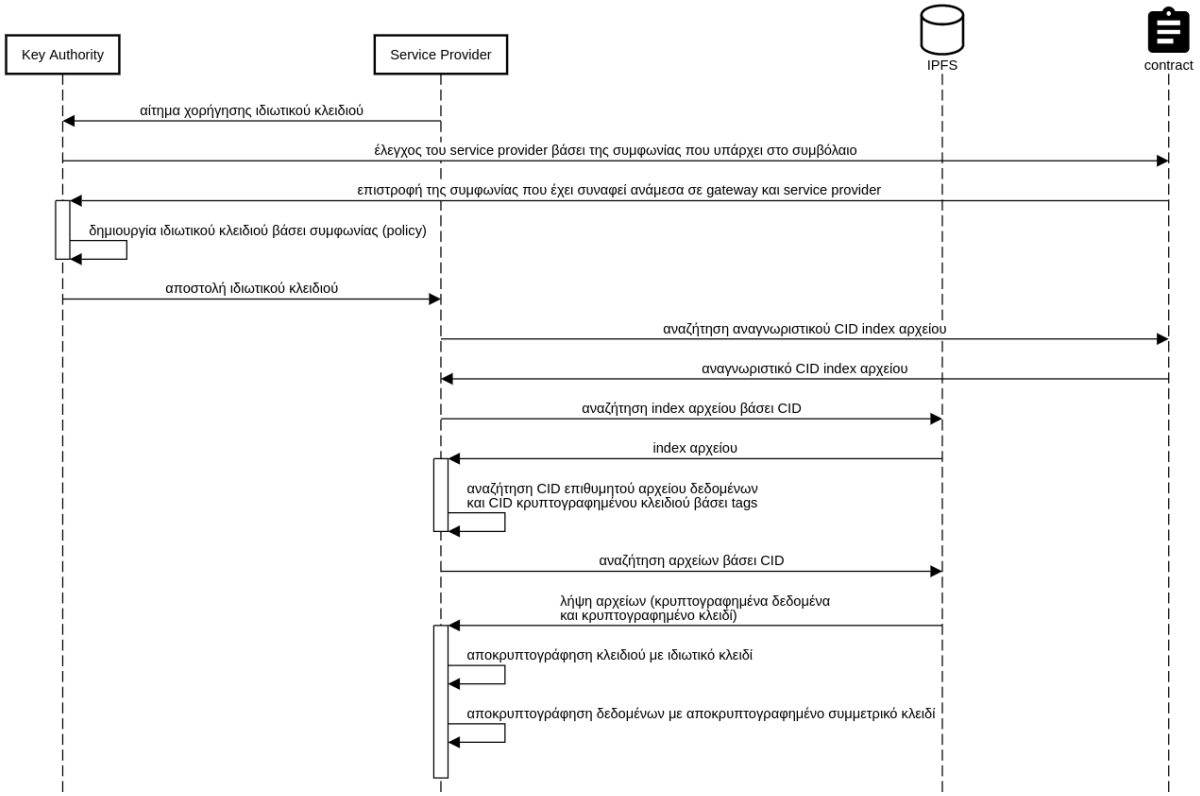
Σε περίπτωση που ο πάροχος επιχειρήσει να χρησιμοποιήσει ένα παλιό κλειδί που του είχε εκδοθεί για να αποκρυπτογραφήσει δεδομένα που έχουν παραχθεί μετά το πέρας της όποιας συμφωνίας είχε υπάρξει, τότε η αποκρυπτογράφηση του συμμετρικού κλειδιού θα αποτύχει και επομένως ο πάροχος δεν δύναται να αποκρυπτογραφήσει τα δεδομένα αυτά.

Σε περίπτωση που ο πάροχος έχει κάποια συμφωνία με τον χρήστη, η οποία δεν έχει λήξει, αλλά προσπαθήσει να χρησιμοποιήσει το κλειδί που του έχει εκδοθεί για να αποκρυπτογραφήσει δεδομένα κατηγορίας εκτός αυτής που έχει συμφωνηθεί, δηλαδή αν του έχει επιτραπεί πρόσβαση μόνο σε δεδομένα κατηγορίας “oxygen level” και προσπαθήσει να αποκτήσει αποκρυπτογραφήσει δεδομένα κατηγορίας “body temprature”, τότε η αποκρυπτογράφηση του συμμετρικού κλειδιού, και επομένως των δεδομένων, θα αποτύχει.

Τέλος, αν ο πάροχος επιχειρήσει να χρησιμοποιήσει κλειδί που του έχει εκδοθεί κατά την διάρκεια της συμφωνίας για να αποκρυπτογραφήσει παλιότερα

δεδομένα στα οποία δεν του έχει δοθεί πρόσβαση, τότε θα αποτύχει. Υπάρχει η δυνατότητα όμως, μέσω συμφωνίας να του δοθεί πρόσβαση σε δεδομένα που παρήχθησαν πριν αυτής.

Σε κάθε περίπτωση, επιτυχής αποκρυπτογράφηση συμμετρικού κλειδιού, και επομένως δεδομένων, είναι δυνατή μόνο όταν υπάρχει (ή έχει υπάρξει) συμφωνία ανάμεσα στον πάροχο και τον χρήστη που δηλώνει ότι τα δεδομένα αυτής της κατηγορίας και ημερομηνίας είναι προσβάσιμα από τον πάροχο.



Εικόνα 5: Περιγραφή του πρωτοκόλλου από μεριάς του service provider

4.2 Τεχνική περιγραφή συστήματος

Με την ολοκλήρωση της ποιοτικής περιγραφής της λειτουργίας του πρωτοκόλλου, στο παρόν υποκεφάλαιο θα ακολουθήσει η τεχνική περιγραφή, δηλαδή η περιγραφή του κώδικα που συνθέτει το πρωτόκολλο. Για την διευκόλυνση της κατανόησης της ανάλυσης, το υποκεφάλαιο θα χωριστεί στα εξής πέντε διαφορετικά τμήματα:
1) βιβλιοθήκες που χρησιμοποιήθηκαν
2) ανάλυση του κώδικα του Key Authority
3) ανάλυση του κώδικα του Gateway
4) ανάλυση του κώδικα του Service Provider
5) μελέτη περιφερειακών τμημάτων κώδικα που απαιτήθηκαν για την ανάπτυξη του πρωτοκόλλου στην παρούσα πτυχιακή εργασία

4.2.1 Βιβλιοθήκες που χρησιμοποιήθηκαν

Gateway > code > gateway.py > ...

```python
import pickle
from charm.core.engine.util import objectToBytes,bytesToObject
import base64
import serpent
import charm.toolbox.symcrypto
from charm.toolbox.symcrypto import SymmetricCryptoAbstraction
from charm.toolbox.pairinggroup import PairingGroup,GT,extract_key
import requests, json
from flask import Flask
from flask_cors import CORS
import ipfsApi
from charm.schemes.abenc.abenc_lsw08 import KPabe
from charm.toolbox.pairinggroup import PairingGroup, GT, deserialize, serialize
import ipfshttpclient
import os
import time
```

Εικόνα 6: Βιβλιοθήκες που χρησιμοποιήθηκαν στην υλοποίηση του gateway

Key_Authority > code > key_authority.py > debug

```python
import pickle
from charm.core.engine.util import objectToBytes,bytesToObject, serializeObject
import base64
import serpent
from flask_restful import Resource, Api, reqparse
from flask import Flask
from charm.toolbox.pairinggroup import PairingGroup, GT
from charm.schemes.abenc.abenc_lsw08 import KPabe
import time, json
import ipfsApi
import os
```

Εικόνα 7: Βιβλιοθήκες που χρησιμοποιήθηκαν στην υλοποίηση του key authority

Service_Provider > code > service_provider.py > debug

```python
import pickle
from charm.core.engine.util import objectToBytes,bytesToObject
from charm.toolbox.pairinggroup import PairingGroup, GT, deserialize, serialize
import base64
import serpent
import requests, json
from charm.schemes.abenc.abenc_lsw08 import KPabe
from charm.toolbox.pairinggroup import PairingGroup, GT
from charm.schemes.abenc.abenc_lsw08 import KPabe
from charm.toolbox.symcrypto import SymmetricCryptoAbstraction
from charm.toolbox.pairinggroup import PairingGroup,GT,extract_key
import ipfsApi
import ipfshttpclient
import os
import time
```

Εικόνα 8: Βιβλιοθήκες που χρησιμοποιήθηκαν στην υλοποίηση του service provider

CHARM

Η βιβλιοθήκη charm είναι ένα framework για την ανάπτυξη κρυπτοσυστημάτων σε high level language. Καθώς το πρωτότυπο αλληλεπίδρασης κάνει χρήση τόσο κρυπτογράφησης συμμετρικού κλειδιού, όσο και κρυπτογράφησης βάσει χαρακτηριστικών, η βιβλιθήκη αυτό έδωσε την δυνατότητα για την γρήγορη και ασφαλή ανάπτυξη του.

Από την βιβλιοθήκη charm χρησιμοποιήθηκαν:

- το scheme abenc_lsw08 που είναι ένας αλγόριθμος KP-ABE και δίνει την δυνατότητα παραγωγής δημοσίου και ιδιωτικού κλειδιού, κρυπτογράφησης και αποκρυπτογράφησης
- η βιβλιοθήκη SymmetricCryptoAbstraction για την χρήση κρυπτογράφησης συμμετρικού κλειδιού
- η βιβλιοθήκη PairingGroup, που απαιτείται για το scheme abenc_lsw08 που αποτελεί κρυπτογράφηση βάσει σύζευξης. Η σύζευξη αυτή γίνεται ανάμεσα σε στοιχεία δύο κρυπτογραφικών ομάδων σε μια τρίτη ομάδα με χαρτογράφηση. Η βιβλιοθήκη PairinggGroup παρέχει τα δομικά στοιχεία για την επιλογή του μαθηματικού συνόλου βάσει του οποίου συντίθεται ο αλγόριθμος κρυπτογράφησης που χρησιμοποιείται.

IpfsApi & ipfshttpclient

Οι δύο βιβλιοθήκες χρησιμοποιήθηκαν για την σύνδεση του gateway και του service provider με έναν ipfs κόμβο.

FLASK

Το Flask είναι ένα web framework που χρησιμοποείται για την ανάπτυξη web applications στην python. Στην παρούσα εργασία χρησιμοποιείται στο key authority για την δημιουργία ενός πλαισίου στο οποίο το key authority μπορεί να δέχεται και να απαντάει σε αιτήματα

serpent & pickle

Το serpent και το pickle είναι βιβλιοθήκες για την σειροποίηση δεδομένων για την πιο έυκολη μεταφορά τους. Στην εργασία αυτή χρησιμοποείται για την διευκόλυνση αποστολής αντικειμένων.

4.2.2 Ανάλυση του κώδικα του Key Authority

```
26  #create necessary objects
27  groupObj = PairingGroup('MNT224')
28  kpabe = KPabe(groupObj)
```

Εικόνα 9: Επιλογή παραμέτρων για τον κρυπτογραφικό αλγόριθμο

Αρχικά, επιλέγονται οι παράμετροι βάσει των οποίων θα λειτουργήσει ο κρυπτογραφικός αλγόριθμος. Το όρισμα που δίνεται στην συνάρτηση PairingGroup αντιστοιχεί σε μια ασύμμετρη καμπύλη με base field 224bits και επιστρέφει ένα μαθηματικό σύνολο το οποίο δίνεται ως όρισμα στην συνάρτηση Kpabe (συνάρτηση του αλγορίθμου abenc_lsw08), αρχικοποιώντας έτσι τις παραμέτρους λειτουργίας του.

```
#get public and master key, setup policy and create secret key
(pk, mk) = kpabe.setup()
```

Εικόνα 10: Δημιουργία δημοσίου κλειδιού και master key

Στην συνέχεια δημιουργούνται το δημόσιο κλειδί και το master key από το οποίο θα δημιουργηθούν τα ιδιωτικά κλειδιά των παρόχων.

```
class endpoint(Resource):
    def post(self):
        args = parser.parse_args()  #get arguments
        name = args['name']
        signature = args['signature']
```

Εικόνα 11: Δημιουργία κλάσης για αποστολή κλειδιού

Στην παραπάνω εικόνα φαίνεται η δημιουργία της κλάσης για την αποστολή κλειδιού σε αιτούντα από το Key Authority. Δίνονται τα αναγνωριστικά του gateway ή του παρόχου που κάνει το αίτημα. Η προσθήκη της “υπογραφής” αποτελεί κομμάτι μελλοντικής ανάπτυξης του κώδικα, για αυτό σε αυτό το σημείο παραμένει απλά ως “signature”.

```
if valid:
    #send private key to service provider
    return {'key': sk_final}
elif permission:
    #send public key to gateway
    return {'key': a}
else:
    #the entity requesting a key is not recognised
    return {'key': '1'}
```

Εικόνα 12: Δομή if-else

Βάσει της δομής if-else αποστέλλεται από το Key Authority ή το ιδιωτικό κλειδί στον πάροχο υπηρεσιών ή το δημόσιο κλειδί στο gateway ή η ειδοποίηση ότι πάροχος που αιτείται κλειδί δεν έχει κάποια υπάρχουσα συμφωνία με το

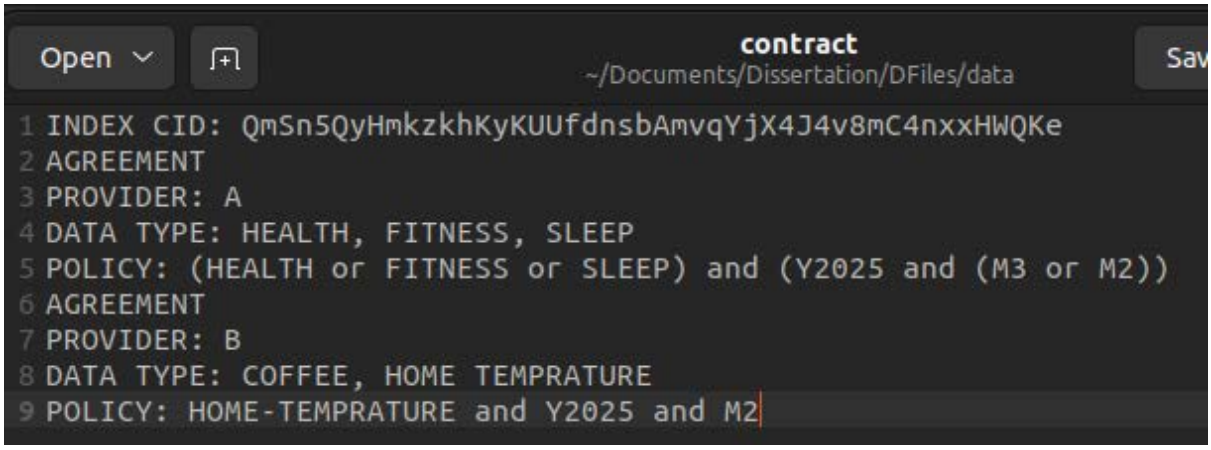
gateway. Η επιλογή γίνεται βάσει του παρακάτω κώδικα που προηγείται της δομής if-else.

```
try:
    with open(contractPath, 'r') as contract:
        #find if the entity making the request is a service provider who has a deal with the gateway
        for line in contract:
            if('AGREEMENT' in line):
                provider =  contract.readline().replace('PROVIDER: ', '').strip()
                print(provider)
                if name == provider:
                    contract.readline()
                    policyProvider = contract.readline().replace('POLICY: ', '').strip()
                    #private key generation
                    mykey= kpabe.keygen(pk,mk, policyProvider)
                    #convert key to bytes and then to unicode
                    sk = objectToBytes(mykey,groupObj)
                    sk_final = sk.decode('ASCII')
                    valid = True
                    break
                else:
                    contract.readline()
                    contract.readline()
except FileNotFoundError:
    printf("File not found!\n")
```

Εικόνα 13: Ανάγνωση συμφωνητικού

Αν το όνομα που έχει δοθεί από τον αιτούντα κλειδί είναι “gateway” τότε στην μεταβλητή permission δίνεται αληθής τιμή και επομένως βάσει της δομής if-else θα αποσταλεί το δημόσιο κλειδί στο gateway.

Αν δεν είναι ο αιτών το gateway, τότε το Key Authority θα ανοίξει το αρχείο στο οποίο καταγράφεται το συμβόλαιο και θα ψάξει αν το όνομα που έχει δοθεί ταυτίζεται με κάποιον από τους παρόχους που έχουν συμφωνίες με τον χρήστη. Αν ο αιτών έχει συμφωνία τότε εντοπίζεται το policy (δηλαδή οι όροι της συμφωνίας) και βάσει αυτού, του δημόσιου κλειδιού και το master key γίνεται παραγωγή του ιδιωτικού κλειδιού (mykey=kpabe.keygen(pk, mk, policyProvider) , το οποίο στην συνέχεια μετατρέπεται σε bytes (sk=objectToBytes(mykey, groupObj) και τέλος σε unicode (sk_final = sk.decode(‘ASCII’)). Τέλος, η μεταβλητή valid λαμβάνει αληθή τιμή και έτσι μέσα από την δομή if-else θα επιστραφεί στον αιτούντα πάροχο το ιδιωτικό κλειδί που παρήχθη βάσει του policy του.



Εικόνα 14: Παράδειγμα συμφωνητικού

Το συμβόλαιο, που για την ανάπτυξη του πρωτοκόλλου στην εργασία αναπαριστάται σε ένα από txt αρχείο, έχει την μορφή που απεικονίζεται στην εικόνα. Η λέξη “AGREEMENT” σηματοδοτεί την αρχή μιας συμφωνίας, έπειτα από την οποία δηλώνεται το όνομα του παρόχου, οι τύποι των δεδομένων στους οποίους του παρέχεται πρόσβαση και η συμφωνία (policy) που έχει κάνει με τον χρήστη.

4.2.3 Ανάλυση του κώδικα του Gateway

```
api = ipfsApi.Client('ipfs_node', 5001)

url_index = None

url =  "http://keyAuthority:5000/keyauthority"
```

Εικόνα 15: Ρύθμιση παραμέτρου για σύνδεση με ipfs

Για να γίνει δυνατή η σύνδεση με IPFS κόμβο καλείται η συνάρτηση ipfsApi.Client και για να γίνει η σύνδεση με το Key Authority ορίζεται ένα url στο οποίο θα στέλνονται τα αιτήματα μέσω του Flask.

```
#create necessary objects
groupObj = PairingGroup('MNT224')
kpabe = KPabe(groupObj)

#the random group element is used fron the symmetric key extraction
group_element = groupObj.random(GT)

#producing symmetric key
a_key = SymmetricCryptoAbstraction(extract_key(group_element))
```

Εικόνα 16: Ρύθμιση παραμέτρων κρυπτογραφικού αλγόριθμου και παραγωγή συμμετρικού κλειδιού

Στην αρχή του κώδικα για το gateway ορίζονται οι παράμετροι με τον ίδιο τρόπο, όπως το Key Authority, δίνοντας ως όρισμα την ασύμμετρη καμπύλη MNT224 και διαλέγεται ένα τυχαίο μαθηματικό σύνολο που θ χρησιμοποιηθεί για την παραγωγή του συμμετρικού κλειδιού, η οποία γίνεται στην συνέχεια με την χρήση της συνάρτησης SymmetricCryptoAbstraction(extract_key(group_element)).

```
51
52  #specifying which file is to be encrypted and uploaded
53  #CHANGE TITLE TO CHANGE FILE
54  title = "Y2025_M2_D2_HEALTH"
55  filename = os.path.join(relative_path, title)
56
```

Εικόνα 17: Ορισμός ονόματος αρχείου

Στην συνέχεια ορίζεται το όνομα του αρχείου που θα κρυπτογραφηθεί και θα προστεθεί στο IPFS.

```
#read data
with open(filename) as file:
    plaintext = file.read()

#encrypt with symmetric key
ciphertxt = a_key.encrypt(plaintext)
```

Εικόνα 18: Ανάγνωση δεδομένων από το αρχείο και κρυπτογράφηση με συμμετρικό κλειδί

Έπεται η ανάγνωση των δεδομένων από το αρχείο στην μεταβλητή plaintext με την συνάρτηση file.read() και η κρυπτογράφηση τους με το συμμετρικό κλειδί (a_key) με την συνάρτηση a_key.encrypt(plaintext). Τα κρυπτογραφημένα δεδομένα δίνονται ως τιμή στην μεταβλητή ciphertxt.

```
#request public key from key authority
response = requests.post(url, json = param).json()

a = response['key']

#in case public key request is denied
if a == '1':
    exit(1)

a = a.encode("ascii")

#get the master public key
master_public_key = bytesToObject(a,groupObj)
```

Εικόνα 19: Αίτηση για δημόσιο κλειδί και μετατροπή σε object

```
77
78  #encryption of the symmetric key with specific attributes using the public key
79  attributes = title.split("_")
80  enc_element = kpabe.encrypt(master_public_key, group_element, attributes)
81
```

Εικόνα 20: Κρυπτογράφηση συμμετρικού κλειδιού με attribute based encryption

Στην συνέχεια αποστέλλεται ένα αίτημα για το δημόσιο κλειδί στο Key Authority με την συνάρτηση requests.post(url, json = param).json() και επιλέγεται το κλειδί από την απάντηση (a = response[‘key]). Σε περίπτωση που το αίτημα δεν γίνει δεκτό, τότε ο κώδικας τερματίζει.

Γίνεται μετατροπή του κλειδιού σε Object με την συνάρτηση bytesToObject και όρισμα μαθηματικό σύνολο που επιλέχθηκε στην αρχή του κώδικα. Έτσι δημιουργείται το master public key που χρησιμοποιείται στην συνέχεια για την κρυπτογράφηση του συμμετρικού κλειδιού ως όρισμα στην συνάρτησης kpabe.encrypt. Για την κρυπτογράφηση του συμμετρικού κλειδιού χρησιμοποιούνται επίσης τα χαρακτηριστικά των δεδομένων που κρυπτογραφήθηκαν, τα οποία εξάγονται από τον τίτλο του αρχείου των δεδομένων (attributes = title.split(“_”)).

```
#change ciphertext object to bytes
ct = objectToBytes(ciphertxt, groupObj)

completeName = os.path.join(relative_path, "ciphertext.pickle")

#save ciphertext to file
with open(completeName, 'wb') as handle:
    pickle.dump(ct, handle)
```

Εικόνα 21: Μετατροπή κρυπτογραφήματος σε bytes και αποθήκευση σε αρχείο

Έπειτα γίνεται μετατροπή των κρυπτογραφημένων δεδομένων σε bytes για να μπορέσουν με αυτή την μορφή να αποθηκευτούν σε ένα αρχείο με την χρήση της συνάρτησης pickle.dump.

```
#change encrypted symmetric key to bytes
ek = objectToBytes(enc_element, groupObj)

#save encrypted symmetric key in file
encKeyName = os.path.join(relative_path, "encrypted_element.pickle")
with open(encKeyName, 'wb') as handle:
    pickle.dump(ek, handle)
```

Εικόνα 22: Μετατροπή κρυπτογραφημένου συμμετρικού κλειδιού σε bytes και αποθήκευση σε αρχείο

Η ίδια διαδικασία ακολουθείται για το κρυπτογραφημένο συμμετρικό κλειδί που αποθηκεύεται σε αρχείο.

```
#add file with ciphertext to ipfs
res = api.add(completeName)
```

Εικόνα 23: Προσθήκη αρχείου κρυπτογραφήματος στο IPFS

```
#add encrypted symmetric key to ipfs
keyRes = api.add(encKeyName)
```

Εικόνα 24: Αποθήκευση αρχείου κρυπτογραφημένου συμμετρικού κλειδιού στο IPFS

Τα δύο αρχεία προστίθενται στο IPFS με την συνάρτηση api.add που επιστρέφει ένα σύνολο δεδομένων σχετικά με την επιτυχή πρόσθεση του κάθε αρχείου.

```
#add CIDs of encrypted files along with tags to index file
url_index_file = os.path.join(relative_path, "url_index.txt")
if url_index == None:
    #open index file
    index_file = open(url_index_file, "a+")
    #add cid of encrypted data file
    hash = [d['Hash'] for d in res if d['Name'] == 'data/ciphertext.pickle']
    #add cid of encrypted symmetric key
    keyHash = [d['Hash'] for d in keyRes if d['Name'] == 'data/encrypted_element.pickle']
    #add tags
    index_file.write("Category: " + str(attributes[3]) + " Date: " + str(attributes[0]) + " "
    + str(attributes[1]) + " " + str(attributes[2]) + "\nCID: " + str(hash[0]) + "\nKey Hash: " + str(keyHash[0]) + "\n")
    index_file.close()
    #add index file to ipfs
    res = api.add(url_index_file)
```

Εικόνα 25: Ανανέωση index αρχείου

Από τα τελευταία τμήματα του κώδικα αποτελεί η ανανέωση του index αρχείου που περιλαμβάνει τα στοιχεία όλως των αρχείων που έχουν προστεθεί στο IPFS. Από το σύνολο δεδομένων (res) που επέστρεψε η συνάρτηση api.add όταν προστέθηκε το αρχείο με τα κρυπτογραφημένα δεδομένα, επιλέγεται το hash (hash = [d['Hash'] for d in res if d['Name'] == 'data/ciphertext.pickle'] που δίνει το CID του αρχείου και η ίδια διαδικασία ακολουθείται για την εύρεση του CID του αρχείου με το κρυπτογραφημένο συμμετρικό κλειδί.

Στην συνέχεια, γράφεται στο index αρχείο η κατηγορία των κρυπτογραφημένων δεδομένων που προστέθηκαν, η ημερομηνία παραγωγής τους, το CID του αρχείου με τα κρυπτογραφημένα δεδομένα και το CID του αρχείου με το κρυπτογραφημένο κλειδί. Αφού γίνει η ανανέωση του index αρχείου με την προσθήκη αυτή, τότε και αυτό με την σειρά του προστίθεται στο IPFS με την χρήση της συνάρτησης api.add.

4.2.4 Ανάλυση του κώδικα του Service Provider

```
api = ipfsApi.Client('ipfs_node', 5001)
```

Εικόνα 26: Εισαγωγή port σε συνάρτηση για σύνδεση με το IPFS

```
url =  "http://keyAuthority:5000/keyauthority"
```

Εικόνα 27: Ορισμός url

Για να γίνει η σύνδεση με τον IPFS κόμβο καλείται η συνάρτηση ipfsApi.Client και ορίζεται το url που θα χρησιμοποιηθεί για να σταλούν αιτήματα στο Key Authority.

```
#create necessary objects
groupObj = PairingGroup('MNT224')
kpabe = KPabe(groupObj)
```

Εικόνα 28: Ρύθμιση παραμέτρων κρυπτογραφικού αλγόριθμου

Στην αρχή του κώδικα για το gateway ορίζονται οι παράμετροι με τον ίδιο τρόπο, όπως το Key Authority, δίνοντας ως όρισμα την ασύμμετρη καμπύλη MNT224.

```
#CHANGE name TO CHANGE SERVICE PROVIDER
#specify name of service provider and provde signature (future work: signature authentication)
param = {'name': 'A', 'signature': 'signature'}

#request private key from key authority
response = requests.post(url, json = param).json()
```

Εικόνα 29: Ορισμός ονόματος του service provider και αίτηση για ιδιωτικό κλειδί

Ορίζεται το όνομα του service provider και με την συνάρτηση requests.post γίνεται αίτημα το key authority για ιδιωτικό κλειδί.

```
k = response['key']

#in case this service provider doesn't have any kind of agreement with gateway
if k == '1':
    print("This provider has not been granted access!\n")
    exit(1)

key = k.encode("ascii")

#change private key from bytes to object
sk = bytesToObject(key,groupObj)
```

Εικόνα 30: Λήψη κλειδιού και μετατροπή σε unicode

Γίνεται λήψη του κλειδιού από το Key Authority και γίνεται μετατροπή σε unicode, ενώ στην συνέχεια γίνεται μετατροπή από bytes σε αντικείμενο με την χρήση του μαθηματικού συνόλου που επιλέγει στην αρχή (bytesToObject).

```
#find index file CID from contract
with open('data/contract', 'r') as contract:
    hash = contract.readline().replace("INDEX CID: ", "")
    hash = hash.strip()
```

Εικόνα 31: Ανάγνωση του CID του index αρχείου από το συμφωνητικό

Στην συνέχεια, ανοίγεται το αρχείο με το συμβόλαιο και γίνεται ανάγνωση του index αρχείου.

```python
#function to download file from ipfs
def get_from_ipfs(cid, mode):
    url = f'http://172.18.0.1:5001/api/v0/cat'
    params = {
        'arg': cid
    }

    try:
        response = requests.post(url, params=params)
        if response.status_code == 200:
            #get index file from ipfs
            if mode == 0:
                with open("response.txt", "w") as f:
                    f.write(response.text)
            #get encrypted data file or encrypted symmetric key file from ipfs
            if mode == 1:
                with open("encodedText.pickle", "wb") as f:
                    f.write(response.content)
            return None
        else:
            print(f"Error: {response.status_code}")
            return None

    except Exception as e:
        print(f"Error retrieving from IPFS: {e}")
        return None
```

Εικόνα 32: Συνάρτηση για λήψη αρχείου από IPFS

Στην εικόνα φαίνεται η συνάρτηση που καλείται για την λήψη αρχείων από το IPFS με ορίσματα το CID και με mode που ορίζει αν θα γίνει λήψη του index αρχείου (mode == 0) ή κάποιο αρχείο με κρυπτογραφημένο περιεχόμενο (mode == 1).

Ορίζεται το url για την σύνδεση με IPFS κόμβο και αποστέλλεται το αίτημα με την συνάρτηση requests.post

Το αρχείο που αποστέλλεται ως απάντηση από το IPFS αποθηκεύεται είτε ως αρχείο κειμένου, στην περίπτωση του index αρχείου, είτε ώς byte αρχείο, στην περίπτωση αρχείου με κρυπτογραφημένα δεδομένα.

Σε περίπτωση που η λήψη του αρχείου από το IPFS αποτύχει, κάτι που θα συμβεί αν το CID είναι λανθασμένο, τότε ο κώδικας θα τερματίσει.

```
#get index file from ipfs
get_from_ipfs(hash, 0)

cid = 0
keyCid = 0

try:
    with open("response.txt", "r+") as f:
        check = 0
        for line in f:
            #IF YOU WANT A DIFFERENT CATEGORY /TAGS- CHANGE HERE
            #search for CIDs according to tags
            if('Category: HEALTH Date: Y2025 M2 D2\n' == line):
                cid = f.readline()
                cid = cid.replace('CID: ', '')
                cid = cid.strip()
                keyCid = f.readline()
                keyCid = keyCid.replace('Key Hash: ', '')
                keyCid = keyCid.strip()
                check = 1
                break
            else:
                f.readline()
                f.readline()
        if check == 0:
            print("No such file found")
            exit(0)
except FileNotFoundError:
    #failed getting index file from ipfs
    print("The file does not exist.")
    exit(0)
```

Εικόνα 33: Λήψη index αρχείου

Καλείται η συνάρτηση get_from_ipfs για την λήψη του index αρχείου. Ανοίγονται το ληφθέν αρχείο, γίνεται αναζήτηση βάσει χαρακτηριστικών του επιθυμητού αρχείου (εδω: Κατηγορια: HEALTH και Ημερομηνία 2/2/2025). Αν το αρχείό βρεθεί στον κατάλογο τότε δίνεται η τιμή 1 στην μεταβλητή check. Αν στο τέλος της αναζήτησης η τιμή του check είναι 0, τότε το αρχείο δεν βρέθηκε και τυπώνεται το ανάλογο μήνυμα και ο κώδικας τερματίζει.

Αν έχει αποτύχει η λήψη του index αρχείου από το IPFS τότε τυπώνεται το αντίστοιχο μήνυμα και ο κώδικας τερματίζει.

```
#get encrypted symmetric key file from ipfs
get_from_ipfs(keyCid, 1)

with open('encodedText.pickle', 'rb') as handle:
    a = pickle.load(handle)

encGroupElement = serpent.tobytes(a)
encGroupElement = bytesToObject(encGroupElement,groupObj)

#decrypt group element and extract symmetric key
group_element = kpabe.decrypt(encGroupElement, sk)
```

Εικόνα 34: Λήψη κρυπτογραφημένου συμμετρικού κλειδιού και αποκρυπτογράφηση του

Στην συνέχεια καλείται η ίδια συνάρτηση με τιμή ορίσματος mode: 1 και cid το cid του αρχείου με το κρυπτογραφημένο συμμετρικό κλειδί για να γίνει λήψη του αρχείου αυτού.

Όταν ολοκληρωθεί η λήψη, γίνεται ανάγνωση του αρχείου και το κρυπτογραφημένο κρυπτογραφικό στοιχείο μετατρέπεται πρώτα σε bytes (serpent.tobytes(a)) και μετά σε object.

Γίνεται αποκρυπτογράφηση του συμμετρικού κλειδιού με την συνάρτηση kpabe.decrypt που παίρνει ως ορίσματα το κρυπτογραφημένο συμμετρικό κλειδί και το ιδιωτικό κλειδί που δόθηκε από το Key Authority.

```
try:
    #produce symmetric key from group element
    a_key = SymmetricCryptoAbstraction(extract_key(group_element))
except:
    #in case of error the symmetric key extraction failed because of failed decryption
    print("You have not been granted access to this data or you are attempting to use the wrong key!")
    exit(0)
```

Εικόνα 35: ΠΑραγωγή συμμετρικού κλειδιού από το αποκρυπτογραφημένο κρυπτογραφικό στοιχείο

Με την χρήση της συνάρτησης SymmetricCryptoAbstraction γίνεται παραγωγή του συμμετρικού κλειδιού από το αποκρυπτογραφημένο κρυπτογραφικό στοιχείο group_element. Σε περίπτωση που η αποκρυπτογράφηση δεν έχει γίνει σωστά, τότε θα προκύψει error κατά την παραγωγή του συμμετρικού κλειδιού και θα τυπωθεί μήνυμα που θα ενημερώνει το πάροχο ότι προσπάθησε να αποκτήσει πρόσβαση σε δεδομένα που δεν έχει δικαίωμα και ο κώδικας θα τερματίσει.

```
#get encrypted data file from ipfs
get_from_ipfs(cid, 1)

with open('encodedText.pickle', 'rb') as handle:
    ciphertext = pickle.load(handle)

#ciphertext from bytes to object
ciphertext = bytesToObject(ciphertext, groupObj)

#decrypt ciphertect using decrypted symmetric key
plaintext = a_key.decrypt(ciphertext)

#print original data
print(str(plaintext, encoding = 'utf-8'))
```

Εικόνα 36: Λήψη αρχείου με κρυπτογραφημένα δεδομένα

Τέλος καλείται η συνάρτηση get_from_ipfs με όρισμα το cid του αρχείου με τα κρυπτογραφημένα δεδομένα. Όταν ολοκληρωθεί η λήψη του αρχείου, τότε γίνεται ανάγνωση αυτού με την συνάρτηση pickle.load και μετατροπή των

δεδομένων από bytes σε object. Στην συνέχεια τα δεδομένα αποκρυπτογράφονται με την συνάρτηση a_key.decrypt που παίρνει ως όρισμα τα κρυπτογραφημένα δεδομένα και τυπώνεται το τελικό αποτέλεσμα.

4.3 Μελέτη περιφερειακών τμημάτων κώδικα που απαιτήθηκαν για την ανάπτυξη του πρωτοκόλλου στην παρούσα πτυχιακή εργασία

```
key_authority:

  build: ./Key_Authority

  container_name: keyAuthority

  expose:
    - "5050"
    - "5000"

  volumes:
    - type: bind
      source: ./Key_Authority/code
      target: /opt/app
    - type: bind
      source: ./data
      target: /opt/app/data
```

Εικόνα 37: Key authority στο docker compose αρχείο

```
service_provider:

  build: ./Service_Provider
  container_name: serviceProvider
  stdin_open: true
  tty: true
  volumes:
    - type: bind
      source: ./data
      target: /opt/app/data
    - type: bind
      source: ./Service_Provider/code
      target: /opt/app
```

Εικόνα 38: Service provider στο docker compose αρχείο

```
gateway:

  build: ./Gateway
  container_name: gateway
  expose:
    - "5000"
  volumes:
    - ipfs_data:/data/ipfs
    - type: bind
      source: ./Gateway/code
      target: /opt/app
    - type: bind
      source: ./data
      target: /opt/app/data
```

Εικόνα 40: Gateway στο docker compose αρχείο

```
ipfs_node:

  image: ipfs/go-ipfs
  container_name: ipfs
  ports:
    - "4001:4001"  # IPFS Swarm Port
    - "5001:5001"  # IPFS API Port
    - "8080:8080"  # IPFS Gateway Port
  volumes:
    - ipfs_data:/data/ipfs
  command: ["daemon", "--migrate=true"]
```

Εικόνα 39: IPFS node στο docker compose αρχείο

Λόγω των απαιτήσεων της βιβλιοθήκης charm έγινε deployment του κώδικα του πρωτοκόλλου σε docker containers.

Στις εικόνες φαίνονται τμήματα του yaml αρχείου docker- compose. Υπάρχουν 4 services: ένα για το gateway, ένα για τον service provider, ένα για το key authority και ένα για έναν ipfs κόμβο.

5. Πειράματα

Το εξής πείραμα σχεδιάστηκε με βασικό ερώτημα πώς αλλάζει ο χρόνος διαπεραίωσης της διαδικασίας αν οι δειγματοληψίες από τις IoT συσκευές διαφέρουν χρονικά. Θέτονται λοιπόν ως προς εξέταση τα εξής σενάρια:

α) Γίνεται δειγματοληψία κάθε ένα λεπτό. Τα αρχεία που κρυπτογραφούνται και προστίθενται περιέχουν τα δεδομένα ενός λεπτού (1 δειγματοληψία)
β) Γίνεται δειγματοληψία κάθε ένα λεπτό. Τα αρχεία που κρυπτογραφούνται και προστίθενται περιέχουν δεδομένα μίας ώρας (60 δειγματοληψίες)
γ) Γίνεται δειγματοληψία κάθε ένα λεπτό. Τα αρχεία που κρυπτογραφούνται και προστίθενται περιέχουν δεδομένα μίας ημέρας (1440 δειγματοληψίες)
δ) Γίνεται δειγματοληψία κάθε μία ώρα. Τα αρχεία που κρυπτογραφούνται και προστίθενται περιέχουν δεδομένα μίας ημέρας (24 δειγματοληψίες)

α) 1 δειγματοληψία

```
gateway             | Timer (Gateway): 1.7688907959964126
```

Εικόνα 41: 1 δειγματοληψία ανά λεπτό (gateway)

```
serviceProvider     | Timer (Service Provider): 0.39189336900017224
```

Εικόνα 42: 1 δειγματοληψία ανά λεπτό (service provider)

Ο χρόνος κρυπτογράφησης και προσθήκης στο IPFS είναι 1.7688907959964126 δευτερόλεπτα. Ο συνολικός χρόνος μέσα σε μια ημέρα είναι 2547,20275 δευτερόλεπτα ή 42.4533792 λεπτά
Ο χρόνος λήψης αρχείων από το IPFS και αποκρυπτογράφησης είναι 0.39189336900017225 δευτερόλεπτα. Ο συνολικός χρόνος μέσα σε μια ημέρα είναι 564.32645136 δευτερόλεπτα ή 0.1567573476 λεπτά

β) 60 δειγματοληψίες

```
serviceProvider     | Timer (Service Provider): 0.3571212549941265
```

Εικόνα 43: 60 δειγματοληψίες (service provider)

```
gateway             | Timer (Gateway): 1.7855870010025683
```

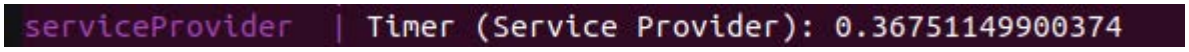
Εικόνα 44: 60 δειγματοληψίες (gateway)

Ο χρόνος κρυπτογράφησης και προσθήκης στο IPFS είναι 1.7855870010025683 δευτερόλεπτα. Ο συνολικός χρόνος μέσα σε μια ημέρα είναι 42.8540880241 ή 0.7142348004 λεπτά

Ο χρόνος λήψης αρχείων από το IPFS και αποκρυπτογράφησης είναι 0.3571212549941265 δευτερόλεπτα. Ο συνολικός χρόνος μέσα σε μια ημέρα είναι 8.57091011986 δευτερόλεπτα ή 0.14284850199 λεπτά

γ) 1440 δειγματοληψίες

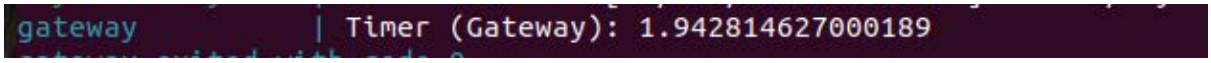
Εικόνα 45: 1440 δειγματοληψίες (gateway)



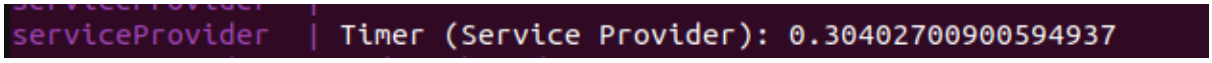
Εικόνα 46: 1440 δειγματοληψίες (service provider)

Ο χρόνος κρυπτογράφησης και προσθήκης στο IPFS είναι 1.9860007910028799 δευτερόλεπτα ή 0.03310001318 λεπτά
Ο χρόνος λήψης αρχείων από το IPFS και αποκρυπτογράφησης είναι 0.36751149900374 δευτερόλεπτα ή 0.00612519165 λεπτά.

δ) 24 δειγματοληψίες



Εικόνα 47: 24 δειγματοληψίες (gateway)



Εικόνα 48: 24 δειγματοληψίες (service provider)

Ο χρόνος κρυπτογράφησης και προσθήκης στο IPFS είναι 1.942814627000189 δευτερόλεπτα. Ο συνολικός χρόνος μέσα σε μια ημέρα είναι 46.627551048 ή 0.7771258508 λεπτά
Ο χρόνος λήψης αρχείων από το IPFS και αποκρυπτογράφησης είναι 0.30402700900594937 δευτερόλεπτα. Ο συνολικός χρόνος μέσα σε μια ημέρα είναι 7.29664821614 δευτερόλεπτα ή 0.1216108036 λεπτά.

Η χρονική διαφορά για το gateway ανάμεσα:

- στην 1 δειγματοληψία και στις 60, είναι ~0.02δευτερόλεπτα, αλλά σε βάθος ημέρας, η διαφορά αυξάνεται κατά 41.7391443996 λεπτά.

- στις 60 δειγματοληψίες και στις 1440 είναι ~0.2 δευτερόλεπτα και σε βάθος ημέρας η διαφορά ανέρχεται σε 0.68113478722 δευτερόλεπτα

Η χρονική διαφορά για το service provider ανάμεσα:

- στην 1 δειγματοληψία και στις 60, είναι ~0.034δευτερόλεπτα, αλλά σε βάθος ημέρας, η διαφορά αυξάνεται κατά 555.75554124 δευτερόλεπτα

- στις 60 δειγματοληψίες και στις 1440 είναι ~0.01 δευτερόλεπτα και σε βάθος ημέρας η διαφορά ανέρχεται σε 6.58490932886 δευτερόλεπτα.

Από το παραπάνω πείραμα εξάγονται τα εξής συμπεράσματα:

1) η διαφορά ανάμεσα στην κρυπτογράφηση και προσθήκη στο ipfs αρχείων με 1 και με 60 δειγματοληψίες είναι μικρή, αλλά ανάγκη για κρυπτογράφηση και προσθήκη αρχείων με μεγαλύτερο ρυθμό μέσα στην μέρα στην περίπτωση του σεναρίου αρχείων με μία δειγματοληψία, αυξάνει κατά πολύ τον χρόνο της διαδικασίας. Στην περίπτωση των αρχείων με 60 και 1440 δειγματοληψίες, η διαφορά στην κρυπτογράφηση και την αποκρυπτογράφηση είναι μεγαλύτερη, αλλά η διαφορά σε βάθος ημέρας είναι κατά πολύ μικρότερη.

2) η διαφορά ανάμερα στην λήψη αρχείων από το ipfs και στην αποκρυπτογράφηση με 1 δειγματοληψία και με 60 δειγματοληψίες είναι μεγαλύτερη από ότι η διαφορά ανάμεσα σε αρχεία με 60 δειγματοληψίες και 1440 δειγματοληψίες, αλλά σε βάθος ημέρας η πρώτη αυξάνεται κατά πολύ σε σύγκριση με την δεύτερη.

Επομένως, ευσταθεί και υποστηρίζεται η υπόθεση ότι το πρωτόκολλο αλληλεπίδρασης, με τον τρόπο που έχει υλοποιηθεί για την παρούσα πτυχιακή εργασία, είναι περισσότερα κατάλληλο για συσκευές που δεν απαιτούν real time επεξεργασία δεδομένων, αλλά βάσει των σεναρίων στα οποία έγιναν πειράματα, η επεξεργασία δεδομένων ανά μια ώρα είναι εφικτή, ενώ η επεξεργασία δεδομένων μια φορά την ημέρα είναι το σενάριο που απαιτεί τον λιγότερο χρόνο, τόσο από μεριάς του gateway όσο και από μεριάς του service provider.

6. Συμπεράσματα

Το πρωτόκολλο αλληλεπίδρασης που προτείνεται λειτουργεί ικανοποιητικά φτιάχνοντας ένα πλαίσιο στο οποίο τα δεδομένα κρυπτογραφούνται, προστατεύοντας έτσι την ιδιωτικότητα του χρήστη, ενώ ταυτόχρονα του δίνει την δυνατότητα να κάνει συμφωνίες με παρόχους υπηρεσιών βάσει των χρονικών ορίων που επιθυμεί ο ίδιος, τα οποία μπορεί να τροποποιήσει άμεσα. Τα κρυπτογραφημένα δεδομένα αποθηκεύονται σε ένα ομότιμο δίκτυο, μειώνοντας έτσι ακόμη περισσότερο τον κίνδυνο διαρροής τους λόγω ενός security breach.

Οι περιορισμοί του πρωτοκόλλου έγκεινται κυρίως στην αδυναμία επεξεργασία δεδομένων σε πραγματικό χρόνο και στον τρόπο αποθήκευσης των δεδομένων, καθώς χρησιμοποιώντας το IPFS η μοναδική εγγύηση για την μακροχρόνια αποθήκευση τους είναι η επιλογή να γίνουν pin, που απαιτεί είτε την επιβάρυνση σε αποθηκευτικό χώρο του ίδιου του χρήστη ή την συνδρομή σε κάποια υπηρεσία, που διατρέχει τον κίνδυνο να καταργήσει εν μέρει την ανωνυμία και την ασφάλεια του πρωτοκόλλου κατά διαρροής δεδομένων. Οι περιορισμοί που υπάρχουν είναι , κατά βάσει, περιορισμοί που υπάρχουν εγγενώς στα πλαίσια και τις μεθόδους που συνθέτουν το πρωτόκολλο. Το πρωτόκολλο που προτείνεται δεν αποτελεί αναγκαστικά μια λύση για την παρούσα φάση της τεχνολογικής εποχής, αλλά, με την πάροδο της τεχνολογίας και την εξέλιξη των τμημάτων που συνθέτουν και εξυπηρετούν το πρωτόκολλο αυτό, αποτελεί μια βάση πάνω στην οποία μπορεί να αναπτυχθεί ένα πλαίσιο και ένα πρωτόκολλο που θα καλύπτει τις ανάγκες της σύγχρονης εποχής.

Σε μελλοντικό χρόνο, το πρωτόκολλο θα μπορούσε να αναπτυχθεί έτσι ώστε το συμβόλαιο ανάμεσα στον χρήστη και στον πάροχο υπηρεσιών να διαπεραιώνεται μέσω ενός smart contract διασφαλίζοντας έτσι την ακεραιότητα του συμβολαίου και με την χρήση ενός blockchain λογαριασμού να δίνει πλήρη ανωνυμία στον χρήστη, που θα μπορεί πλέον να συνάπτει συμφωνίες και να πληρώνει τους παρόχους χωρίς να δίνει τα προσωπικά του στοιχεία.

Το πρωτόκολλο αλληλεπίδρασης, επομένως, που αναπτύχθηκε αποτελεί μια επαρκή και λειτουργική βάση για την ανάπτυξη ενός εναλλακτικού μοντέλου διαχείρισης δεδομένων από IoT συσκευές που δίνει έμφαση στην ιδιωτικότητα, την ασφάλεια και την κυριότητα ,από τον χρήστη, των δεδομένων που παράγονται από τις συσκευές IoT.

7. Βιβλιογραφία

[1] Azbeg, Kebira & Ouchetto, Ouaïl & Jai Andaloussi, Said. (2022). BlockMedCare: A healthcare system based on IoT, Blockchain and IPFS for data management security. Egyptian Informatics Journal. 23. 10.1016/j.eij.2022.02.004.

[2] Huang, Tse-Yang & Chen, Yu-Chi & Hsieh, Tsung-Chen & Chang, Huan-Chi & Chang, Chih-Chieh. (2023). A Secure and IoT-Enabled Data Sharing System Based on IPFS and IOTA Blockchain. 50-57. 10.1145/3625078.3625085.

[3] BhavinKumar Kothari, Shakthi Mudaliar and Sabestin Nadar, Prof. Mrinal Khadse (2020), "Securing IoT with Blockchain" ISBN:978-1-7281-4685-0

[4] Muhammad Salek Ali, Koustabh Dolui, Fabio Antonelli, IoT Data Privacy via Blockchains and IPFS

[5] João Paulo de Brito Gonçalves, Gabriel Spelta, Rodolfo da Silva Villaça, Roberta Lima Gomes (2022), IoT Data Storage on a Blockchain Using Smart Contracts and IPFS, 2022 IEEE International Conference on Blockchain (Blockchain),

DOI 10.1109/Blockchain55522.2022.00078

[6] Annie Badman, Matthew Kosinksi (2024), "What is symmetric encryption?" [https://www.ibm.com/think/topics/symmetric-encryption]

[7] Delfs, Hans; Knebl, Helmut (2007). "Symmetric-key encryption". *Introduction to cryptography: principles and applications*. Springer. ISBN .

[8] Stallings, William (3 May 1990). *Cryptography and Network Security: Principles and Practice*. Prentice Hall. p. 165. ISBN .

[9] Shamir, A. (1985). Identity-Based Cryptosystems and Signature Schemes. In: Blakley, G.R., Chaum, D. (eds) Advances in Cryptology. CRYPTO 1984. Lecture Notes in Computer Science, vol 196. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39568-7_5

[10] J. Bethencourt, A. Sahai and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," *2007 IEEE Symposium on Security and Privacy (SP '07)*, Berkeley, CA, USA, 2007, pp. 321-334, doi: 10.1109/SP.2007.11.

[11] Allison Lewko, Amit Sahai and Brent Waters "Revocation Systems with Very Small Private Keys", Large Universe Construction, IEEE S&P 2010

[12] Vipul Goyal, Omkant Pandey, Amit Sahai and Brent Waters, Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data *ACM CCS (2006)*

[13] Klaus Wehrle, Stefan Götz, Simon Rieche, January 2005, Lecture Notes in Computer Science 3485:79-93 DOI:10.1007/11530657_7