

Ιόνιο Πανεπιστήμιο

Τμήμα Πληροφορικής



Ανάπτυξη Εφαρμογής σε Blockchain

Αλέξανδρος-Γεώργιος Λουκάς ΑΜ : Π2017123

Επιβλέπων : - Χριστόφορος Νταντογιάν -

10 Οκτωβρίου 2021

Επιβλέπων

Χριστόφορος Νταντογιάν Επίκουρος Καθηγητής
Ιόνιο Πανεπιστήμιο

Τριμελής Επιτροπή

Χριστόφορος Νταντογιάν Επίκουρος Καθηγητής
Ιόνιο Πανεπιστήμιο

Εμμανουήλ Μάγκος Αναπληρωτής Καθηγητής Τμήματος Πληροφορικής
Ιόνιο Πανεπιστήμιο

Αγγελική Τσώχου Επίκουρη Καθηγήτρια Τμήματος Πληροφορικής
Ιόνιο Πανεπιστήμιο

Περίληψη

Η τεχνολογία blockchain γίνεται όλο και πιο γνωστή και αποδεκτή από τον κόσμο μετά την δημιουργία και την εξάπλωση του πρώτου κρυπτονομίσματος Bitcoin από τον Shatoshi Nakamoto .Στην περίπτωση του internet κανένας δεν μπορούσε να φανταστεί στις αρχές της δεκαετίας του 70 τις αλλαγές που θα προκαλούσε σε παγκόσμιο επίπεδο. Με παρόμοιο τρόπο , το Blockchain θα επιφέρει ριζοσπαστικές αλλαγές στον τρόπο με τον οποίο δομούνται οι ιστοσελίδες καθώς ένα από τα κύρια χαρακτηριστικά του είναι ο αποκεντρωτισμός , η μη ύπαρξη δηλαδή ενός κεντρικού Server ή ενός τρίτου παράγοντα. Στον οικονομικό τομέα επίσης , σήμερα το συνολικό κεφάλαιο που έχει επενδυθεί μόνο στο Bitcoin ξεπερνά τα 700 δισεκατομμύρια ευρώ. Ξεκινά δηλαδή μία νέα εποχή (Web.3) όπου υλοποιείται ήδη ο επαναπροσδιορισμός του τρόπου που υλοποιούνται οι οικονομικές συναλλαγές και του τρόπου υλοποίησης των ιντερνετικών υπηρεσιών κάθε είδους. Τα πιο σημαντικά οφέλη του blockchain που το διαφοροποιούν από τις υπάρχουσες τεχνολογίες είναι η αποκέντρωση , η αμεταβλητότητα δεδομένων και η διαφάνεια. Ήδη μετά από έρευνες , η χρήση του παρατηρείται σε τομείς όπως IoT , σε υπηρεσίες υγείας , πνευματικά δικαιώματα , εφοδιαστική αλυσίδα κ.α.

Στην εργασία αυτή , θα γίνει μία εισαγωγή αποδόμησης όλων των μηχανισμών του Blockchain και του Ethereum Blockchain και θα αναπτυχθεί μία εφαρμογή ηλεκτρονικής ψηφοφορίας πάνω στο Ethereum δίκτυο.

Πρόλογος και Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Χριστόφορο Νταντογιάν , για την άψογη συνεργασία μας , τις συμβουλές και τις κατευθύνσεις στα της εργασίας , και για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα αρκετά πρωτότυπο και σύγχρονο θέμα. Τέλος επιβάλλεται να ευχαριστήσω τους φίλους μου για την υποστήριξή τους στα δύσκολα χρόνια.

Περιεχόμενα

Περιεχόμενα

Περίληψη.....	i
Περιεχόμενα	ii
Εισαγωγή.....	1
A')Blockchain	3
A'.1)Ορισμός.....	3
A'.2) Ιστορία του Blockchain	4
A'.3)Κύρια χαρακτηριστικά.....	5
A'.4) Τρόπος Λειτουργίας.....	6
A'.5) Δομή blockchain	8
A'.6) Peer-to-Peer Network(P2P)	8
A'.7) Κόμβοι (Nodes)	9
A'.8) Ψηφιακές Υπογραφές	9
A'.9) Συναρτήσεις Κατακερματισμού.....	10
A'.10) Δομή Block	11
A'.11) Merkle Trees	12
A'.11.1) Merkle Path.....	13
A'.12) Ομοφωνία Blockchain (Consensus).....	14
A'.12.1) PoW(Proof-of-Work).....	15
A'.12.2) PoS(Proof-of-Stake).....	16
A'.14) Τύποι Blockchain.....	18
A'.14.1) Public	18
A'.14.2) Private	18
A'.14.3) Consortium.....	19
A'.15) Πιθανές επιθέσεις σε Blockchain εφαρμογές	19
A'.15.1) Η επίθεση 51%.....	20
A'.15.3) Eclipse Attack	21
A'.15.4) Double Spending Attack.....	21
A'.15.5) Liveness Attack.....	21
B')Ethereum	25
B'.1) Ορισμός.....	25
B'.2) Συναλλαγές	25
B'.3) Λογαριασμοί	27
B'.4) Smart Contracts.....	27
B'.5) Blocks.....	28
B'.6) EVM.....	29
B'.7) Gas	30

B'.8) Dapps	31
B'.9) Δίκτυα Ethereum.....	31
B'.10) Tokens.....	31
B'.10.1) ERC-20.....	32
B'.10.2)ERC-721.....	33
Γ') Βιβλιογραφική Ανασκόπηση.....	39
Γ'.1) Απαιτήσεις Συστήματος	39
Γ'.2) Έμπιστη Οντότητα	39
Γ'.3) Blockchain-Based E-Voting System	40
Γ'.4) An Evoting Protocol with Decentralization and Privacy.....	41
Γ'.5) Zerocoin.....	43
Γ'.6) ERC-20 Voting System	44
Γ'.7) Open Vote Network (New Castle University).....	45
Γ'.9) Blockchain voting system.....	46
Δ') Παρουσίαση Εφαρμογής.....	49
Δ'.1) Αρχική σελίδα	49
Δ'.2) Κατάθεση ψήφων και υποψηφίων	50
Δ'.3) Σελίδα διαχειριστή	52
Δ'.4) Αποτελέσματα.....	56
Ε') Υλοποίηση Εφαρμογής	59
Ε'.1) Σχεδιασμός	59
Ε'.2) Smart Contract.....	60
Ε'.2.1) Απαραίτητες Μεταβλητές	60
Ε'.2.2) Modifier.....	62
Ε'.2.5) Πρόταση υποψηφίων.....	63
Ε'.2.6) Δήλωση Ψήφου	63
Ε'.2.7) Καταμέτρηση ψήφων	64
Ε'.2.8) Αλλαγή φάσης ψηφοφορίας.....	65
Ε'.2.9) Τελευταίες συναρτήσεις.....	65
Ε'.3) Smart Contract Deployment.....	67
Ε'.4) Εργαλεία και Τεχνολογίες	68
Ε'.4.1) Ganache	69
Ε'.4.2) Metamask	69
Ε'.4.3) Truffle Suite	69
Ε'.4.4) Node.js.....	70
Ε'.4.5) NPM(Node Package Manager).....	71
Ε.4.5) Visual Studio Code	71

E'.4.6) Javascript.....	72
E'.4.7) Web3.js.....	72
E'.4.8) React.js	73
E'.4.9) Solidity	73
E'.5) Front-End	73
E'.5.1) Components.....	73
E'.5.2) App.js	75
E'.5.3) Functional Components.....	75
ΣΤ') Μελλοντικές βελτιώσεις / Συμπεράσματα.....	81
Παράρτημα Ι : Κώδικας	83
Παράρτημα ΙΙ Οδηγίες Εγκατάστασης.....	89
Βιβλιογραφία.....	89

Πίνακας Εικόνων

ΕΙΚΟΝΑ 1 ΠΑΡΑΔΕΙΓΜΑ ΕΝΟΣ BLOCKCHAIN [2]	4
ΕΙΚΟΝΑ 2 : Η ΑΡΧΗ ΚΑΙ Η ΤΩΡΙΝΗ ΕΞΕΛΙΞΗ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ BLOCKCHAIN [4]..	5
ΕΙΚΟΝΑ 3 : ΚΥΚΛΟΣ ΖΩΗΣ ΜΙΑΣ ΣΥΝΑΛΛΑΓΗΣ ΣΕ ΕΝΑ BLOCKCHAIN [5]	7
ΕΙΚΟΝΑ 4 : ΔΟΜΗ BLOCKCHAIN [6].....	8
ΕΙΚΟΝΑ 5 : ΔΟΜΗ ΕΝΟΣ BLOCK ΠΗΓΗ : [11].....	12
ΕΙΚΟΝΑ 6 Η ΔΟΜΗ ΕΝΟΣ MERKLE TREE [12]	13
ΕΙΚΟΝΑ 7 : ΠΑΡΑΔΕΙΓΜΑ ΥΠΟΛΟΓΙΣΜΟΥ ΤΟΥ MERKLE PATH [12]	14

EIKONA 8 : SOFT FORK [17]	17
EIKONA 9 : HARD FORK [18].....	17
EIKONA 10 : PUBLIC BLOCKCHAIN NETWORK [19]	18
EIKONA 11 : PRIVATE BLOCKCHAIN NETWORK [19]	19
EIKONA 12: CONSORTIUM BLOCKCHAIN NETWORK [19]	19
EIKONA 13 : SELF-MINING ATTACK [21].....	20
EIKONA 14 : SELF-MINING ATTACK [21]	21
EIKONA 15 : ΠΕΡΙΕΧΟΜΕΝΑ ΣΥΝΑΛΛΑΓΗΣ ETHEREUM [23]	26
EIKONA 16 : ΤΥΠΟΙ ΛΟΓΑΡΙΑΣΜΩΝ [23]	27
EIKONA 17 : BLOCK HEADER [26].....	29
EIKONA 18 : ΜΕΤΑΓΛΩΤΤΙΣΗ ΥΨΗΛΟΥ ΕΠΙΠΕΔΟΥ ΓΛΩΣΣΑΣ ΚΑΙ ΕΚΤΕΛΕΣΗ BYTECODE [27]	30
EIKONA 19 : ΔΟΜΗ ΕΚΛΟΓΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ [35]	40
EIKONA 20 : ΔΙΑΔΙΚΑΣΙΑ ΨΗΦΟΦΟΡΙΑΣ [36].....	41
EIKONA 21 TOKEN ΕΠΙΒΕΒΑΙΩΣΗΣ	42
EIKONA 22 ΨΗΦΟΔΕΛΤΙΟ	42
EIKONA 23 ΨΗΦΟΔΕΛΤΙΟ ΑΛΛΑΓΗΣ ΨΗΦΟΥ	42
EIKONA 24 ΕΦΑΡΜΟΓΗ BLIND ΣΥΝΑΡΤΗΣΗΣ.....	42
EIKONA 25 ΑΦΑΙΡΕΣΗ ΤΗΣ ΥΠΟΓΡΑΦΗΣ ΤΗΣ ΈΜΠΙΣΤΗΣ ΟΝΤΟΤΗΤΑΣ ΑΠΟ ΤΟΝ ΧΡΗΣΤΗ	42
EIKONA 26: ΠΡΟΤΕΙΝΟΜΕΝΟ ΣΥΣΤΗΜΑ ΠΗΓΗ : HTTPS://IEEEEXPLORE.IEEE.ORG/ABSTRACT/DOCUMENT/9333937	44
EIKONA 27: ΓΥΡΟΙ ΨΗΦΟΦΟΡΙΑΣ [49]	45
EIKONA 28 : ΨΗΦΟΦΟΡΙΑ [40]	46
EIKONA 29: ΗΛΕΚΤΡΟΝΙΚΟ ΣΥΣΤΗΜΑ ΨΗΦΟΦΟΡΙΑΣ [41].....	47
EIKONA 30 : ΑΡΧΙΚΗ ΣΕΛΙΔΑ.....	49
EIKONA 31 : ΣΕΛΙΔΑ ΨΗΦΟΦΟΡΟΥ	50
EIKONA 32: ΜΗΝΥΜΑ ΜΗ ΚΑΤΑΧΩΡΗΜΕΝΗΣ ΕΓΓΡΑΦΗΣ.....	50
EIKONA 33: ΜΗΝΥΜΑ ΜΗ ΚΑΤΑΧΩΡΗΜΕΝΗΣ ΕΓΓΡΑΦΗΣ.....	51
EIKONA 34: ΑΠΟΤΥΧΗΜΕΝΗ ΣΥΝΑΛΛΑΓΗ	51
EIKONA 35: ΚΑΤΑΧΩΡΗΣΗ 3 ΥΠΟΨΗΦΙΩΝ	52
EIKONA 36: ΦΑΣΗ ΤΗΣ ΨΗΦΟΦΟΡΙΑΣ	52
EIKONA 37: ΣΕΛΙΔΑ ΔΙΑΧΕΙΡΙΣΤΗ ΠΡΙΝ ΤΗΝ ΣΥΝΔΕΣΗ.....	53
EIKONA 38 : ΣΕΛΙΔΑ ΔΙΑΧΕΙΡΙΣΤΗ ΜΕΤΑ ΤΗΝ ΣΥΝΔΕΣΗ.....	53
EIKONA 39 : ΕΠΙΒΕΒΑΙΩΣΗ ΚΑΙ ΠΡΑΓΜΑΤΟΠΟΙΗΣΗ ΣΥΝΑΛΛΑΓΗΣ.....	54
EIKONA 40: ΜΗΝΥΜΑ ΕΛΕΓΧΟΥ ΕΓΓΕΓΡΑΜΜΕΝΟΥ ΨΗΦΟΦΟΡΟΥ	55
EIKONA 41: ΜΗΝΥΜΑ ΕΛΕΓΧΟΥ ΜΕ ΕΓΓΕΓΡΑΜΜΕΝΟΥ ΨΗΦΟΦΟΡΟΥ	55
EIKONA 42: ΚΟΥΜΠΙΑ ΔΙΑΧΕΙΡΙΣΤΗ	56
EIKONA 43: ΣΕΛΙΔΑ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΠΡΙΝ ΤΗΝ ΚΑΤΑΜΕΤΡΗΣΗ ΨΗΦΩΝ	56
EIKONA 44 : ΣΕΛΙΔΑ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΜΕΤΑ ΤΗΝ ΚΑΤΑΜΕΤΡΗΣΗ ΨΗΦΩΝ	57
EIKONA 45 ΜΗΧΑΝΙΣΜΟΣ ΕΦΑΡΜΟΓΗΣ.....	59
EIKONA 46: ΔΗΛΩΣΗ ΜΕΤΑΒΛΗΤΩΝ SMART CONTRACT	61
EIKONA 47 : ΔΗΛΩΣΗ ΤΩΝ MODIFIER ΤΟΥ SMART CONTRACT	62
EIKONA 48: CONSTRUCTOR.....	62
EIKONA 49: ΣΥΝΑΡΤΗΣΗ ΕΓΓΡΑΦΗΣ ΨΗΦΟΦΟΡΩΝ	63
EIKONA 50: ΣΥΝΑΡΤΗΣΗ ΚΑΤΑΘΕΣΗΣ ΥΠΟΨΗΦΙΩΝ	63
EIKONA 51: ΣΥΝΑΡΤΗΣΗ ΚΑΤΑΘΕΣΗΣ ΨΗΦΟΥ.....	64
EIKONA 52: ΣΥΝΑΡΤΗΣΗ ΚΑΤΑΜΕΤΡΗΣΗΣ ΨΗΦΩΝ.....	64
EIKONA 53: ΣΥΝΑΡΤΗΣΗ ΑΡΧΗΣ ΚΑΤΑΘΕΣΗΣ ΥΠΟΨΗΦΙΩΝ	65
EIKONA 54: ΣΥΝΑΡΤΗΣΕΙΣ ΤΕΛΟΥΣ ΚΑΤΑΘΕΣΗΣ ΥΠΟΨΗΦΙΩΝ ΚΑΙ ΑΡΧΗΣ ΨΗΦΟΦΟΡΙΑΣ.....	65
EIKONA 55: ΣΥΝΑΡΤΗΣΗ ΤΕΛΟΥΣ ΦΑΣΗΣ ΨΗΦΟΦΟΡΙΑΣ	65
EIKONA 56: ΒΟΗΘΗΤΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ	66
EIKONA 57: ΠΑΡΑΜΕΤΡΟΠΟΙΗΣΗ ΑΡΧΕΙΟΥ TRUFFLE-CONFIG.JS	67
EIKONA 58 : ΣΥΝΔΕΣΗ ΣΕ ΑΛΛΑ ΔΙΚΤΥΑ ΑΠΟ ΤΟ TRUFFLE-CONFIG.JS.....	67
EIKONA 59: ΜΕΤΑΓΛΩΤΤΙΣΗ ΤΟΥ SMART CONTRACT	67

EIKONA 60: DEPLOYMENT ΤΩΝ SMART CONTRACTS ΣΤΟ ΔΙΚΤΥΟ	68
EIKONA 61: ΌΛΑ ΤΑ ΑΡΧΕΙΑ ΤΟΥ FRONT-END	74
EIKONA 62: APP.JS COMPONENT	75
EIKONA 63: ΜΕΤΑΒΛΗΤΕΣ ΤΟΥ REGISTRATION.JS	76
EIKONA 64 : ΣΥΝΑΡΤΗΣΗ ΕΛΕΓΧΟΥ ΕΓΓΡΑΦΗΣ ΤΟΥ ΑΡΧΕΙΟΥ REGISTRATION.JS ..	77
EIKONA 65: ΣΥΝΑΡΤΗΣΗ ΕΓΓΡΑΦΗΣ ΨΗΦΟΦΟΡΩΝ ΤΟΥ ΑΡΧΕΙΟΥ REGISTRATION.JS	78
EIKONA 66: JSX .1	78
EIKONA 67: JSX.2	79

Εισαγωγή

Σε αυτή την εργασία πραγματοποιήθηκε η ανάπτυξη μιας αποκεντρωμένης εφαρμογής ηλεκτρονικής ψηφοφορίας πάνω σε ένα Ethereum δίκτυο. Η χρήση της τεχνολογίας έχει γίνει πλέον συνηθισμένη για να βοηθήσει στην κάλυψη των ανθρώπινων αναγκών. Η αυξανόμενη χρήση της τεχνολογίας έχει φέρει νέες προκλήσεις στη διαδικασία της δημοκρατίας, καθώς οι περισσότεροι άνθρωποι σήμερα δεν εμπιστεύονται τις κυβερνήσεις τους, καθιστώντας τις εκλογές πολύ σημαντικές στη σύγχρονη δημοκρατία . Οι εκλογές έχουν μεγάλη δύναμη στον καθορισμό της τύχης ενός έθνους ή μιας οργάνωσης. Με την ανάπτυξη της τεχνολογίας, η χρήση της τεχνολογίας για την αντιμετώπιση των προβλημάτων που παρουσιάζονται γίνεται σημαντική, καθώς και οι περιπλοκές της διαδικασίας συλλογής. Η ασφάλεια είναι και θα είναι πάντα η μεγαλύτερη ανησυχία για ένα σύστημα ηλεκτρονικής ψηφοφορίας. Δεν θα πρέπει να υπάρχει σύστημα ηλεκτρονικής ψηφοφορίας για την ασφάλεια των δεδομένων και θα πρέπει να είναι σε θέση να αντέχει σε πιθανές επιθέσεις .Ανέκαθεν υπήρχαν ευπάθειες στον παραδοσιακό τρόπο διεξαγωγής των εκλογών .Μερικά πιθανά σενάρια απάτης είναι η δημιουργία ψεύτικης λίστας με τα άτομα που έχουν το δικαίωμα να ψηφίσουν , ψεύτικες ταυτότητες αλλά και παραποίηση των εκλογικών αποτελεσμάτων. Η δημιουργία ενός ασφαλούς και έγκριτου εκλογικού συστήματος αποτελεί βασική προτεραιότητα σε πολλές σε παγκόσμια κλίμακα καθώς δεν τα φαινόμενα εκλογικής απάτης είναι πολύ πιθανό να συμβεί. Στην σημερινή εποχή όπου οι υπηρεσίες παρέχονται ηλεκτρονικά , υπάρχουν χώρες όπου έχει δοκιμαστεί στην πράξη επιτυχημένα η χρήση συστημάτων ηλεκτρονικής ψηφοφορίας . Η Εσθονία είναι μία χώρα που πρωτοπόρησε το 2003 και το σύστημά της χρησιμοποιείται ακόμα. Έκτοτε η υλοποίηση και η χρήση συστήματος ηλεκτρονικής ψηφοφορίας έχει εφαρμοστεί από την Ελβετία , την Ολλανδία , την Νορβηγία κ.α. Παρόλα αυτά , η διείσδυση του διαδικτύου σε πολλές πτυχές τις καθημερινότητας δεν συνεπάγεται σε απόλυτη ασφάλεια. Μέχρι τώρα όλες σχεδόν οι ηλεκτρονικές εφαρμογές λειτουργούν σε ένα κεντρικό Server όπου τα δεδομένα τους αποθηκεύονται σε βάσεις δεδομένων. Τα υπάρχοντα συστήματα ηλεκτρονικής ψηφοφορίας υλοποιούνται στη βάση αυτού του μοντέλου. Η εξάρτηση του μοντέλου από έναν κεντρικό Server και βάσεων δεδομένων , καθιστά εφικτή μία επίθεση DDoS(Distributed Denial of Service) .Παρόλα λοιπόν τα πλεονεκτήματα σε σχέση με τον παραδοσιακό τρόπο ψηφοφορίας , δημιουργούνται άλλου τύπου ευπάθειες σε τέτοιου είδους εφαρμογές καθώς πολλές είναι οι επιθέσεις που μπορούν να διεξαχθούν εις βάρος τους. Η τεχνολογία του Blockchain φαίνεται ικανή να καλύψει πολλά από τα προβλήματα και κενά ασφαλείας που μπορούν να δημιουργηθούν σε μία ηλεκτρονική ψηφοφορία . Πρόκειται για ένα κατακεντρωμένο δημόσιο κατάστιχο , στο οποίο καταγράφονται όλες οι συναλλαγές μεταξύ χρηστών που επιβεβαιώνονται από τους κόμβους του δικτύου χωρίς ενδιάμεσες οντότητες. Τα δεδομένα που περιέχει είναι αδύνατο να τροποποιηθούν από την στιγμή που καταγράφονται , είναι κατακεντρωμένα και ασφαλή

, συνεπώς παρέχει υψηλού επιπέδου διαθεσιμότητα , ακεραιότητα και επαληθευσιμότητα. Χάρης σε αυτά τα χαρακτηριστικά ,μπορεί να αποτελέσει μία εναλλακτική σε όλες τις υλοποιήσεις ηλεκτρονικών εκλογών.

Στο πρώτο κεφάλαιο αναλύονται εκτενέστερα και περιγράφονται οι βασικές πτυχές της τεχνολογίας blockchain αλλά και πιθανές επιθέσεις και ευπάθειες που μπορούν να δημιουργηθούν. Στο δεύτερο κεφάλαιο περιγράφεται συγκεκριμένα το Ethereum Blockchain αφού πάνω σε αυτό υλοποιήθηκε η εφαρμογή. Στο τρίτο κεφάλαιο γίνεται μία σύντομη βιβλιογραφική έρευνα για τις υπάρχουσες υλοποιήσεις εφαρμογών ηλεκτρονικής ψηφοφορίας σε blockchain. Στο τέταρτο κεφάλαιο γίνεται η παρουσίαση της εφαρμογής. Στο πέμπτο κεφάλαιο περιγράφονται οι τεχνολογίες που χρησιμοποιήθηκαν. Τέλος στο έκτο κεφάλαιο καταγράφονται μελλοντικές πιθανές βελτιώσεις και τα συμπεράσματα.

Κεφάλαιο Α΄

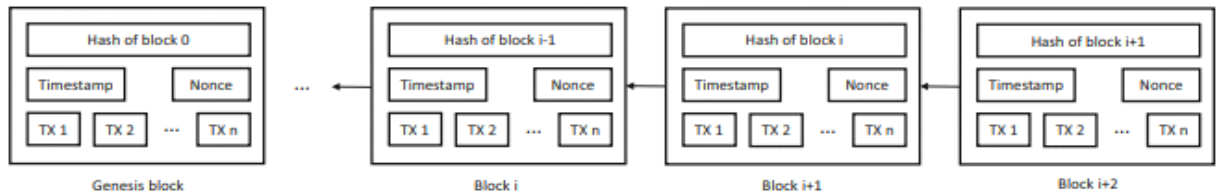
Α΄)Blockchain

Σε αυτό το κεφάλαιο γίνεται μία ανάλυση για όλες τις τεχνολογίες που δομούν ένα Blockchain με περιεκτικό τρόπο αλλά και στα προβλήματα ασφαλείας που μπορεί να αντιμετωπίσει.

Α΄.1)Ορισμός

Το Blockchain είναι μία τεχνολογία η οποία πρωτοεμφανίστηκε το 2009 με την δημοσίευση ενός άρθρου, αγνώστου μέχρι σήμερα προέλευσης με τον τίτλο «a peer to peer electronic cash system» με την υπογραφή Shatoshi Nakamoto [1] . Είναι η βασική τεχνολογία πίσω από το Bitcoin , το οποίο αποτέλεσε την πρώτη επιτυχημένη πρακτική εφαρμογή του και από όλα τα άλλα κρυπτονομίσματα στην αγορά και έχει πολλά υποσχόμενες δυνατότητες πέρα από τα ψηφιακά νομίσματα. Πρόκειται για μία κατακευματισμένη βάση δεδομένων γνωστή ως ledger , στην οποία αποθηκεύεται πληροφορία με την μορφή συναλλαγών , η οποία είναι κατακευματισμένη με έναν αποκεντρωτικό τρόπο σε όλες τις υπολογιστικές συσκευές που συγκροτούν ένα peer-to-peer δίκτυο. Οι συναλλαγές διαδίδονται σε όλο το δίκτυο , επικυρώνονται από τους χρήστες αυτού του δικτύου , ελέγχονται για την εγκυρότητά τους και έπειτα συγκεντρώνονται στα μπλοκ. Κάθε μπλοκ συνδέεται με το προηγούμενό του σχηματίζοντας έτσι μία αλυσίδα μέχρι το πρώτο μπλοκ που δημοσιεύτηκε. Η αλυσίδα των μπλοκ αναφέρεται στο γεγονός ότι κάθε μπλοκ αναφέρεται με έναν κρυπτογραφικό τρόπο στο αμέσως προηγούμενο μπλοκ. Τα δεδομένα σε κάθε μπλοκ είναι ανέφικτο να τροποποιηθούν χωρίς την αλλαγή όλων των προηγούμενων μπλοκ , το οποίο απαιτεί την συναίνεση όλου του δικτύου. Το blockchain επεκτείνεται με κάθε επιπλέον μπλοκ και ως εκ τούτου αντιπροσωπεύει ένα πλήρες «βιβλίο» του ιστορικού συναλλαγών . Η διαδικασία της δημοσίευσης του επόμενου μπλοκ λέγεται mining και πραγματοποιείται μέσα από έναν διαγωνισμό μεταξύ πολλών κόμβων(miners) . Αυτό πραγματοποιείται με την εφαρμογή ενός αλγορίθμου συναίνεσης , διασφαλίζοντας ότι όλες οι συναλλαγές στο καινούργιο μπλοκ είναι έγκυρες και έτσι όλοι οι κόμβοι διαθέτουν ένα πανομοιότυπο αντίγραφο του blockchain. Οι πιο δημοφιλείς αλγόριθμοι που χρησιμοποιούνται από τα περισσότερα κρυπτονομίσματα είναι ο Proof-of-Work (PoW) και ο Proof-of-Stake(PoS). Τα Κύρια και σημαντικότερα χαρακτηριστικά είναι η αμεταβλητότητα των καταγραφών , η αποκέντρωση , η ακεραιότητα των δεδομένων , η δυνατότητα δημιουργίας κρυπτονομισμάτων , η μεταφορά αξιών μεταξύ ομότιμα συνδεδεμένων χρηστών και η ανωνυμία καθώς οι πραγματικές ταυτότητες των χρηστών παραμένουν κρυφές.

Μερικοί κλάδοι που έχουν επηρεαστεί καταλυτικά από αυτήν την επαναστατική τεχνολογία είναι η οικονομία, το IoT(Internet of Things) , η ιατρική , εφοδιαστική αλυσίδα , επικοινωνία κ.α.

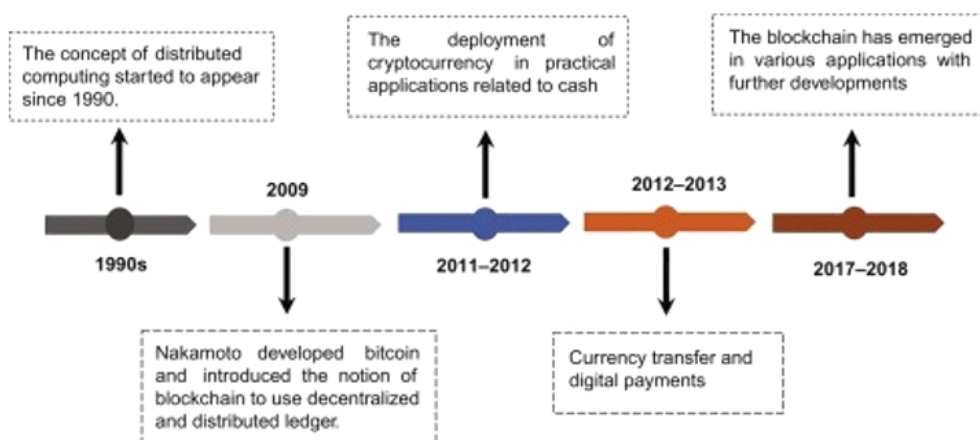


Εικόνα 1 Παράδειγμα ενός blockchain [2]

Α'.2) Ιστορία του Blockchain

Η ιδέα πίσω από την τεχνολογία blockchain περιγράφηκε ήδη το 1991, όταν οι ερευνητές Stuart Haber και W. Scott Stornetta εισήγαγαν μια υπολογιστικά πρακτική λύση για ψηφιακά έγγραφα με χρονική σφραγίδα, έτσι ώστε να μην μπορούν να αναχαιτιστούν ή να παραβιαστούν. Το 1998, ένας διάσημος επιστήμονας υπολογιστών Nick Szabo άρχισε να εργάζεται πάνω σε μια ιδέα για ένα αποκεντρωμένο νόμισμα, το οποίο ονόμασε «Bit Gold». Δημοσίευσε ένα άρθρο όπου πρότεινε μια αλυσίδα κατακερματισμού με χρονική σφραγίδα (blockchain) που μπορεί να λύσει το πρόβλημα των διπλών δαπανών (double-spending) χρησιμοποιώντας hashcash, μια πρώιμη έκδοση του συστήματος Proof-of-Work με σκοπό τον περιορισμό των ανεπιθύμητων μηνυμάτων σε φόρουμ και μηνύματα ηλεκτρονικού ταχυδρομείου. Το 2000, ο Stefan Konst πρότεινε την ιδέα των κρυπτογραφικά ασφαλισμένων αλυσίδων σε μια ερευνητική εργασία με τίτλο "Secure Log Files Based on Cryptographically Concatenated Entries". Παρουσίασε ένα μοντέλο όπου οι καταχωρήσεις στην αλυσίδα μπορούν να εντοπιστούν από την δημιουργία της για να αποδείξουν την αυθεντικότητα, το ίδιο μοντέλο που βλέπουμε στο Bitcoin και σε άλλα κρυπτονομίσματα σήμερα. Τον Οκτώβριο του 2008, ένα άγνωστο άτομο ή ομάδα ατόμων , που αυτοπροσδιορίστηκε ως Satoshi Nakamoto δημοσίευσε ένα άρθρο , εισάγοντας τον κόσμο στο θέμα των blockchains , περιγράφοντας τα οφέλη ενός ηλεκτρονικού συστήματος μετρητών που ονομάζεται Bitcoin. Μέχρι σήμερα, το Bitcoin και άλλα κρυπτονομίσματα είναι αναμφισβήτητα η πιο συχνά αναγνωρισμένη περίπτωση χρήσης του blockchain και, ως εκ τούτου, αποτελούν την κατάλληλη βάση για την εξήγηση των αρχών λειτουργίας της τεχνολογίας. Το Bitcoin ήταν μόνο η αρχή από πολλές εφαρμογές blockchain . Το 2013 ένας προγραμματιστής ο Vitalik Buterin ξεκίνησε την ανάπτυξη ενός συστήματος το οποίο ονομάστηκε Ethereum , βασισμένου στην τεχνολογία

blockchain που συνδύαζε μία τεχνολογία προγραμματιστικών λειτουργιών ,τα smart contracts [3] .



Εικόνα 2 : Η αρχή και η τωρινή εξέλιξη της τεχνολογίας Blockchain [4]

Α'.3)Κύρια χαρακτηριστικά

- **Αποκεντρωμένο.** Το βασικό χαρακτηριστικό του blockchain, είναι ότι δεν χρειάζεται να βασίζεται σε μια έμπιστη κεντρική οντότητα πχ. Τράπεζα . Τα δεδομένα μπορούν να καταγραφούν, να αποθηκευτούν και να ενημερωθούν κατανεμημένα σε πολλαπλά συστήματα.
- **Διαφάνεια .** Η καταγραφή των δεδομένων ανά σύστημα blockchain είναι διαφανής σε κάθε κόμβο και είναι επίσης διαφανές για την ενημέρωση των δεδομένων, γι 'αυτό μπορεί κάποιος να εμπιστευτεί το blockchain.
- **Open-source.** Τα περισσότερα συστήματα blockchain είναι ανοιχτά σε όλους, οι εγγραφές μπορούν να ελεγχθούν δημόσια και οποιοσδήποτε μπορεί επίσης να χρησιμοποιήσει την τεχνολογία για να δημιουργήσει οποιαδήποτε εφαρμογή επιθυμεί.
- **Αυτονομία.** Λόγω της βάσης συναίνεσης, κάθε κόμβος στο σύστημα blockchain μπορεί να μεταφέρει ή να ενημερώσει δεδομένα με ασφάλεια. Η ιδέα είναι να εμπιστευτείτε ένα μόνο άτομο στο ολόκληρο το σύστημα και κανείς δεν μπορεί να παρέμβει σε αυτό.
- **Αμετάβλητο .** Οι καταχωρήσεις θα διατηρούνται για πάντα, και δεν μπορούν να τροποποιηθούν παρά μόνο στην «απίθανη» περίπτωση που κάποιος

αποκτήσει τον έλεγχο υπολογιστικής δύναμης περισσότερο από το 51% όλων των κόμβων ταυτόχρονα.

- **Ανωνυμία.** Οι τεχνολογίες blockchain έλυσαν το πρόβλημα εμπιστοσύνης μεταξύ κόμβου σε κόμβου, οπότε η μεταφορά δεδομένων ή ακόμη και η συναλλαγή μπορεί να είναι ανώνυμη με την μόνη απαίτηση να είναι η διεύθυνση blockchain του κόμβου.
- **Εμπιστοσύνη :** Νέα πληροφορία μπορεί να εισαχθεί στο blockchain μόνο όταν η πλειονότητα των χρηστών του δικτύου δώσουν την έγκρισή τους , αφού λάβουν ικανοποιητική απόδειξη ότι οι πληροφορίες που διαβιβάζονται κρυπτογραφικά είναι αληθείς. Η επαλήθευση ταυτότητας των πληροφοριών γίνεται σε σύντομα χρονικά διαστήματα και οι ενημερωμένες πληροφορίες αποθηκεύονται στο blockchain και είναι διαθέσιμες σε όλους τους ομότιμους συμμετέχοντες του δικτύου.
- **Ουσιώδεις βελτιώσεις:** Η τεχνολογία blockchain μπορεί να επιφέρει ουσιώδεις βελτιώσεις στην εξοικονόμηση κόστους αλλά και χρόνου κατά τη διαδικασία μεταφοράς χρημάτων ή κεφαλαίων καθώς οι συναλλαγές είναι πιθανές οποιαδήποτε χρονική στιγμή και δεν απαιτούν την διαμεσολάβηση κανενός παράγοντα ή επιτροπής για την επαλήθευση της εγκυρότητας των καταγραφών/εγγράφων.

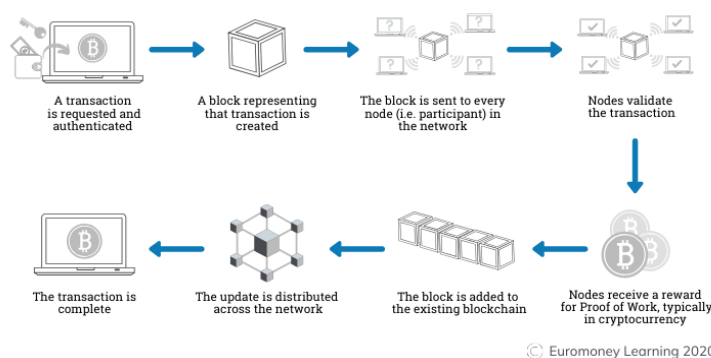
Α΄.4) Τρόπος Λειτουργίας

Η τεχνολογία του blockchain είναι ένα αποκεντρωμένο και κατανεμημένο «δημόσιο βιβλίο» που κατασκευάζεται γύρω από ένα P2P (peer-to-peer) δίκτυο. Αυτό το σύστημα μπορεί να μοιραστεί ανοιχτά μεταξύ των χρηστών του για να σχηματίσει ένα αμετάβλητο αρχείο συναλλαγών καθώς κάθε ένας έχει ένα αντίγραφο αυτού. Κάθε φορά που προστίθεται μια συναλλαγή, τα νέα δεδομένα σχηματίζουν ένα νέο μπλοκ στο τέλος της αλυσίδας . Κάθε έγκυρη συναλλαγή που πραγματοποιείται σε ένα blockchain περνά από τα ίδια βήματα ανεξάρτητα από το αν χρησιμοποιείται για οικονομικές συναλλαγές ή παρακολούθηση προϊόντων. Ο κύκλος των συναλλαγών μπορεί να παρουσιαστεί σε τέσσερα διαφορετικά, συνεχόμενα στάδια:

- Γίνεται εγγραφή για κάθε μία συναλλαγή. Αυτή η εγγραφή, η οποία περιέχει ορισμένα στοιχεία των ατόμων που πραγματοποιούν τη συναλλαγή,

πιστοποιείται με την ψηφιακή υπογραφή του καθενός με την χρήση του ιδιωτικού κλειδιού.

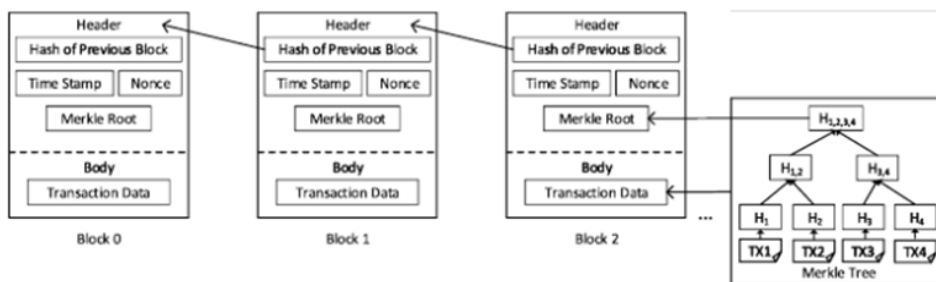
- Η συναλλαγή διαδίδεται σε όλους τους κόμβους του P2P δικτύου. Κάθε συναλλαγή επαληθεύεται για να διασφαλιστεί η εγκυρότητά της. Αυτή η διαδικασία επαλήθευσης ολοκληρώνεται από τους υπολογιστές/κόμβους (nodes) που είναι συνδεδεμένοι στο δίκτυο, καθένας από τους οποίους ελέγχει για να διασφαλίσει ότι η συναλλαγή είναι νόμιμη. Επειδή πρόκειται για μια αποκεντρωμένη διαδικασία, σημαίνει ότι κάθε κόμβος στο δίκτυο πρέπει να συμφωνήσει για να ολοκληρωθεί.
- Μόλις επαληθευτούν οι συναλλαγές, κάθε συναλλαγή προστίθεται σε ένα μπλοκ από τους miners. Στην περίπτωση που μία συναλλαγή δεν είναι έγκυρη δεν θα συμπεριληφθεί στο επόμενο μπλοκ και δεν θα πραγματοποιηθεί. Στο σημείο αυτό οι miners ανταγωνίζονται μεταξύ τους για το ποιος θα βρει πρώτος την λύση σε έναν κρυπτογραφικό γρίφο. Ο νικητής αυτού του «διαγωνισμού» θα ανταμειφτεί με κρυπτονομίσματα. Τα μπλοκ περιέχουν συναλλαγές, μεταδεδομένα και το καθένα είναι μοναδικό. Για κάθε μπλοκ υπολογίζεται η τιμή κατακερματισμού της κεφαλίδας του, η οποία ταυτόχρονα το προσδιορίζει μοναδικά και συμπεριλαμβάνεται στο επόμενο μπλοκ σχηματίζοντας μία αλυσίδα. Το γεγονός αυτό διασφαλίζει την ακεραιότητα των δεδομένων που περιέχει ένα μπλοκ καθώς υποδεικνύει ότι δεν έχουν τροποποιηθεί από τότε που καταγράφηκαν. Έπειτα το καινούργιο μπλοκ διαδίδεται στο δίκτυο και οι υπόλοιποι miners το αποδέχονται μόνο αν όλες οι συναλλαγές μέσα στο μπλοκ και το ίδιο είναι έγκυρα.
- Με την ολοκλήρωση της παραπάνω διαδικασίας, το μπλοκ προστίθεται στο τέλος του blockchain. Αυτό μας οδηγεί στο τέλος της διαδικασίας δημιουργίας και της επαλήθευσης σε ένα blockchain καθώς και στην ολοκλήρωση της συναλλαγής. Μόλις ολοκληρωθεί ένα μπλοκ, σύντομα θα ακολουθήσει ένα άλλο μπλοκ - συνήθως μέσα σε λίγα λεπτά.



Εικόνα 3 : Κύκλος ζωής μιας συναλλαγής σε ένα Blockchain [5]

Α'.5) Δομή blockchain

Μία δομή δεδομένων blockchain είναι μια διατεταγμένη, συνδεδεμένη λίστα μπλοκ συναλλαγών. Τα μπλοκ συνδέονται "προς τα πίσω" και το καθένα αναφέρεται στο προηγούμενό του μπλοκ μέσω ενός δείκτη hash. Ένας τέτοιος δείκτης αποτελείται από την τιμή κατακερματισμού του μπλοκ στο οποίο δείχνει. Αυτή η τιμή κατακερματισμού βοηθάει στην επαλήθευση της ακεραιότητας των δεδομένων του προηγούμενου μπλοκ. Τα block είναι παρόμοια με τους κόμβους μιας συνδεδεμένης λίστας (linked-list). Το πρώτο μπλοκ της αλυσίδας ονομάζεται "genesis" μπλοκ και δεν έχει πατρικό μπλοκ. Η κύρια διαφορά μεταξύ ενός blockchain και μιας συνδεδεμένης λίστας είναι ότι οι δείκτες σε ένα blockchain είναι κρυπτογραφικά ασφαλείς μιας και είναι αδύνατη η τροποποίησή τους χωρίς να γίνει αντιληπτή. Αντίθετα, οι δείκτες σε μια συνδεδεμένη λίστα μπορούν να τροποποιηθούν ανά πάσα στιγμή χωρίς να επηρεαστεί η ακεραιότητα των δεδομένων. Οι ασφαλείς δείκτες δημιουργούν τάξη σε όλα τα μπλοκ και καθιστούν αποτελεσματικά το blockchain μια δομή δεδομένων όπου νέα δεδομένα μπορούν να προστεθούν μόνο με νέα μπλοκ.



Εικόνα 4 : Δομή Blockchain [6]

Α'.6) Peer-to-Peer Network(P2P)

Σε ένα δίκτυο peer-to-peer ένα σύνολο υπολογιστών είναι συνδεδεμένοι μεταξύ τους με ίδια δικαιώματα και ευθύνες. Οι πόροι του δικτύου μοιράζονται ισοδύναμα πχ φάκελοι, αποθηκευτικός χώρος, υπολογιστική δύναμη, εύρος κ.α. Πληροφορίες και δεδομένα που διαθέτει ένας κόμβος είναι προσβάσιμα για ανάγνωση από όλους τους άλλους. Κάθε συσκευή σε ένα τέτοιο δίκτυο κατέχει τον ρόλο του client και του server και αναφέρεται ως "peer". Κύρια χαρακτηριστικά του είναι η προσαρμοστικότητα καθώς καινούργιοι κόμβοι μπορούν να εισέλθουν στο δίκτυο με εύκολο τρόπο, υψηλή απόδοση σε αντίθεση με το κλασικό σύστημα client-server όπου όσο μεγαλύτερος ο αριθμός των πελατών τόσο πιο χαμηλή η απόδοση, αξιοπιστία αφού άμα σταματήσει η λειτουργία ενός κόμβου το υπόλοιπο δίκτυο λειτουργεί κανονικά. Τέλος αποτρέπει η συμφόρηση διότι η κίνηση κατανέμεται σε πολλούς υπολογιστές [7].

Α'.7) Κόμβοι (Nodes)

Σε ένα δίκτυο blockchain οι συναλλαγές επικυρώνονται από μία κοινότητα χρηστών (nodes) και έπειτα καταγράφονται στα μπλοκ. Οι χρήστες του πραγματοποιούν συναλλαγές μεταξύ τους πάνω σε ένα κατανεμημένο P2P δίκτυο ανταλλάσσοντας μηνύματα.

Οι κόμβοι αποτελούν ένα ζωτικής σημασίας μέρος για ένα οικοσύστημα blockchain καθώς είναι εκείνοι που πραγματοποιούν την πλειονότητα των αλλαγών. Αφού το δίκτυο είναι ένα αποκεντρωμένο peer-to-peer δίκτυο κάθε ένας κόμβος λειτουργεί σαν client-server. Το γεγονός ότι τα P2P δίκτυα δεν βασίζονται σε κεντρικό διακομιστή τα καθιστά πιο ανθεκτικά. Το blockchain έχει σχεδιαστεί να είναι ένα αποκεντρωμένο σύστημα. Κάθε κόμβος σε ένα blockchain δίκτυο έχει την ικανότητα να αντιγράψει το κατανεμημένο βιβλίο (ledger) , το οποίο ενημερώνεται για οποιαδήποτε αλλαγή/προσθήκη γίνει μέσω ενός μηχανισμού ομοφωνίας (blockchain consensus). Ένα από τα βασικότερα έργα που καλούνται να επιτελέσουν είναι η επεξεργασία των συναλλαγών. Οποιοσδήποτε είναι συνδεδεμένος στο δίκτυο του blockchain είναι υποχρεωμένος να διαδώσει οποιαδήποτε συναλλαγή λάβει από άλλους κόμβους σε όλο το δίκτυο. Ένας από τους κόμβους του δικτύου είναι υποχρεωμένος (καθώς δεν υπάρχει ένας κεντρικός-κυρίαρχος) να συλλέξει όλες τις συναλλαγές από το κοινό κατανεμημένο βιβλίο , να φτιάξει το νέο μπλοκ με βάσει αυτές και έπειτα να το διαδώσει στο υπόλοιπο δίκτυο. Υπάρχουν διαφορετικοί τύποι κόμβων. Οι full-nodes , οι οποίοι αποθηκεύουν όλη την πληροφορία που υπάρχει στο blockchain , μαζί με τις κεφαλίδες όλων των μπλοκ και τις συναλλαγές. Ένας κόμβος που είναι miner ανήκει σε αυτή την κατηγορία. Ένας lightweight-node ή αλλιώς SPV(Simple Payment Verification) αποθηκεύει μόνο τις κεφαλίδες των μπλοκ , οι οποίες περιέχουν πληροφορίες για ένα συγκεκριμένο προηγούμενο μπλοκ . Επίσης βασίζονται στους full-nodes για την παροχή πληροφορίας. Το κύριο έργο τους είναι απλά να διαδίδουν συναλλαγές αλλά και να τις επαληθεύουν . Στο σημείο αυτό παρατηρείται μία ευπάθεια καθώς τέτοιου είδους κόμβοι βασίζονται σε άλλους για την απόκτηση πληροφοριών και έτσι κάποιος επιτιθέμενος μπορεί να διαδώσει ψεύτικες πληροφορίες [8] .

Α'.8) Ψηφιακές Υπογραφές

Σε ένα τέτοιο σύστημα η ταυτότητα του κάθε χρήστη καταγράφεται με ένα ζεύγος από δημόσιο-ιδιωτικό κλειδί που είναι μαθηματικά συνδεδεμένα μεταξύ τους χάρη στην ασύμμετρη κρυπτογράφηση. Στην πραγματικότητα μόνο το δημόσιο κλειδί του καθενός αποκαλύπτεται στους άλλους χρήστες. Η δημιουργία της ψηφιακής υπογραφής επιτυγχάνεται αρχικά με ένα one-way hash των δεδομένων. Ο αλγόριθμος της υπογραφής θα κρυπτογραφήσει το hash value με την χρήση του ιδιωτικού

κλειδιού και έτσι ορίζεται η ψηφιακή υπογραφή. Από την πλευρά του παραλήπτη, λαμβάνει τα δεδομένα με την ψηφιακή υπογραφή και κάνει χρήση του αλγορίθμου επαλήθευσης ο οποίος παίρνει ως είσοδο το δημόσιο κλειδί του αποστολέα, την ψηφιακή υπογραφή και παράγει ένα αποτέλεσμα. Ο παραλήπτης εφαρμόζει την ίδια συνάρτηση κατακερματισμού στα δεδομένα και συγκρίνει το αποτέλεσμα αυτής της ενέργειας με την έξοδο του αλγορίθμου επαλήθευσης. Η ψηφιακή υπογραφή θεωρείται έγκυρη στην περίπτωση που τα 2 παραπάνω έχουν την ίδια τιμή. Αυτές οι υπογεγραμμένες συναλλαγές διαδίδονται στη συνέχεια σε κάθε χρήστη του δικτύου και είναι απαραίτητο να επικυρωθούν προτού εισαχθούν σε ένα μπλοκ και καταγραφούν. Σε ένα blockchain, ο μηχανισμός των ψηφιακών χρησιμοποιείται για την πιστοποίηση της αυθεντικότητας και της ακεραιότητας των συναλλαγών [6]. Κάθε κόμβος διαθέτει ένα ζεύγος δημοσίου-ιδιωτικού κλειδιού. Προτού κάποιος κόμβος διαδώσει μία συναλλαγή την υπογράφει με το ιδιωτικό του κλειδί και οι κόμβοι που θα την λάβουν απαιτείται να ελέγξουν την αυθεντικότητά της με το δημόσιο κλειδί. Αυτός ο μηχανισμός ψηφιακής υπογραφής των συναλλαγών ενισχύει την αυθεντικότητα και την ακεραιότητα των συναλλαγών σε όλο το δίκτυο καθώς διαβεβαιώνουν την επικύρωσή τους. Πρακτικά η τυπική χρήση των ψηφιακών υπογραφών αποτελείται από 2 φάσεις [9]:

- **Την φάση της υπογραφής,**
 $\text{sig} := \text{sign}(\text{sk}, \text{transaction})$ όπου sk είναι το ιδιωτικό κλειδί. Η μέθοδος της υπογραφής λαμβάνει το ιδιωτικό κλειδί και μία συναλλαγή σαν δεδομένα εισόδου και παράγει μία «υπογραφή» σαν έξοδο.
- **Φάση της επιβεβαίωσης.**
 $\text{Valid} := \text{verify}(\text{pk}, \text{transaction}, \text{sig})$ όπου pk είναι το δημόσιο κλειδί. Η μέθοδος της επιβεβαίωσης παίρνει το δημόσιο κλειδί, την συναλλαγή και την υπογραφή σαν δεδομένα εισόδου και η έξοδος του είναι μία τιμή Boolean η οποία θα είναι αληθής αν η υπογραφή είναι έγκυρη και λάθος αν όχι.

Α'.9) Συναρτήσεις Κατακερματισμού

Οι συναρτήσεις κατακερματισμού έχουν ένα μεγάλο πεδίο εφαρμογής στον τομέα της πληροφορικής. Πρόκειται για μαθηματικές συναρτήσεις οι οποίες μετατρέπουν μία είσοδο απεριόριστο σε αριθμό δεδομένων σε μία συμβολοσειρά σταθερού μήκους. Μερικές από τις απαιτήσεις που πρέπει να πληροί μία τέτοια συνάρτηση είναι να μην μπορεί να υπάρξει ίδια τιμή κατακερματισμού από 2 διαφορετικά δεδομένα εισόδου, να είναι μονόδρομη που σημαίνει ότι κανένας που διαθέτει την έξοδο της συνάρτησης δεν μπορεί να υπολογίζει τα δεδομένα εισόδου και να είναι δύσκολο για κάποιον δεδομένου της τιμής εισόδου και εξόδου να βρει μία άλλη τιμή εισόδου που να παράγει την ίδια έξοδο. Κάθε block περιέχει ένα μοναδικό αναγνωριστικό, ένα hash που το ξεχωρίζει από τα υπόλοιπα, δηλαδή ένα ψηφιακό αποτύπωμα (fingerprint),

που παρασκευάζεται με τη χρήση συνάρτησης κατακερματισμού στην κεφαλίδα μπλοκ δύο φορές μέσω του αλγορίθμου SHA256. Το προκύπτων hash έχει μέγεθος 32-byte και ονομάζεται κατακερματισμός μπλοκ (block hash), αλλά είναι πιο ακριβής ο ορισμός « κατακερματισμός κεφαλίδας μπλοκ»(block header hash), επειδή χρησιμοποιείται μόνο η κεφαλίδα μπλοκ για τον υπολογισμό του. Για παράδειγμα, το «000000000019d6689c085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f» είναι ο κατακερματισμός του πρώτου μπλοκ bitcoin που δημιουργήθηκε ποτέ. Κάθε μπλοκ αναφέρεται επίσης σε ένα προηγούμενο μπλοκ, γνωστό ως το πατρικό μπλοκ, μέσω του πεδίου "προηγούμενο μπλοκ κατακερματισμού" (previous block hash) στην κεφαλίδα μπλοκ. Με άλλα λόγια, κάθε μπλοκ περιέχει το hash του πατρικού του μπλοκ μέσα στη δική του κεφαλίδα. Η ακολουθία των τιμών κατακερματισμού που συνδέουν κάθε μπλοκ με το πατρικό του δημιουργεί μια αλυσίδα που πηγαίνει προς τα πίσω μέχρι το πρώτο μπλοκ που δημιουργήθηκε ποτέ, γνωστό ως μπλοκ γένεσης (genesis block). Το γεγονός όμως ότι το πεδίο previous block hash βρίσκεται στην κεφαλίδα κάθε block σημαίνει πως το hash κάθε block βασίζεται στο hash του πατρικού του. Στην περίπτωση λοιπόν που ένα πατρικό block αλλάξει με οποιονδήποτε τρόπο, αλλάζει και το hash του. Απόρροια αυτού είναι η αλλαγή στο πεδίο (previous block hash) του παιδικού του block. Η αλλαγή αυτή επιφέρει μία σειρά αλλαγών μέχρι και το τελευταίο block καθώς κάθε hash από κάθε block που ακολουθεί αυτό που τροποποιήθηκε αρχικά, υπολογίζεται από την αρχή. Σε αυτό το σημείο μπορεί κανείς να παρατηρήσει το γεγονός ότι όταν ένα block διαδέχεται από πολλά άλλα, δεν μπορεί να τροποποιηθεί χωρίς να επιβληθεί ο επανυπολογισμός όλων των επόμενων block. Λόγω του ότι αυτός ο επανυπολογισμός θα απαιτούσε τεράστιους και δύσκολους υπολογισμούς, η ύπαρξη μιας μακράς αλυσίδας block ενισχύει την βαθιά ιστορία του blockchain να είναι αμετάβλητη. Αυτό είναι ένα από τα βασικά χαρακτηριστικά της ασφάλειας του blockchain.

Α'.10) Δομή Block

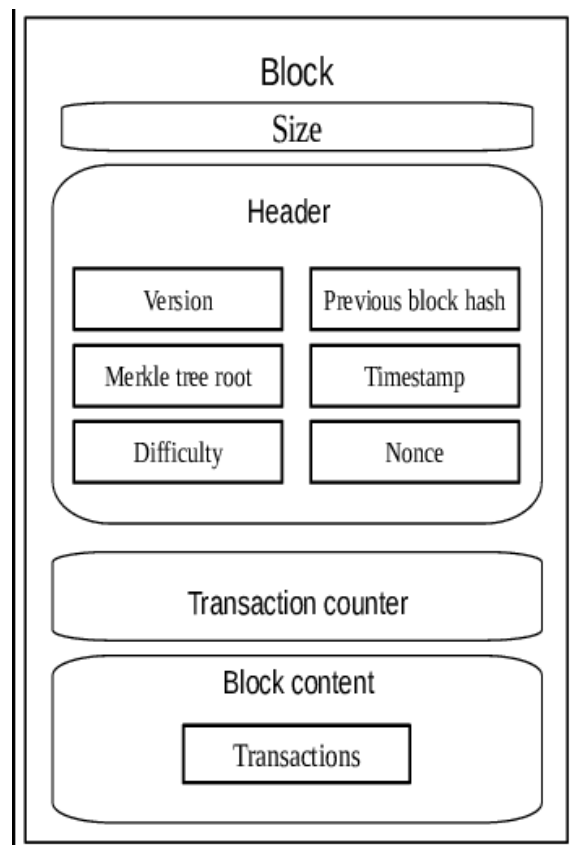
Ένα block είναι μία δομή δεδομένων η οποία συγκεντρώνει συναλλαγές με σκοπό να συμπεριληφθούν στο blockchain. Κάθε μπλοκ αποτελείται από 2 στοιχεία: Την κεφαλίδα και το σώμα. Τα πεδία κάθε block όπως φαίνεται από το παρακάτω σχήμα είναι το μέγεθός του (size) σε bytes, μία κεφαλίδα που περιέχει μεταδεδομένα ακολουθούμενη από τον συνολικό αριθμό των συναλλαγών που περιέχει και από μία μεγάλη λίστα με όλες τις συναλλαγές. Η κεφαλίδα ενός block όπως φαίνεται στην παρακάτω εικόνα αποτελείται από 6 πεδία:

- **Version:** Υποδεικνύει ποια έκδοση λογισμικού χρησιμοποιήθηκε από τον miner και ποια σειρά κανόνων επικύρωσης ακολουθήθηκαν.
- **Previous block hash:** Ένας δείκτης στο πατρικό block που δημιουργεί μία τάξη σε όλη την αλυσίδα block και διασφαλίζει ότι κανέναν προηγούμενο

block δεν μπορεί να τροποποιηθεί αν δεν αλλάξει το τωρινό και όλα τα επακόλουθα blocks.

- **Merkle tree root hash** : Ένα hash της ρίζας του merkle-tree από όλες τις συναλλαγές αυτού του block.
- **Timestamp**: Ο κατά προσέγγιση χρόνος δημιουργίας αυτού του block μετρημένος με Unix-epoch.
- **Difficulty** : Μέτρο δυσκολίας εύρεσης του νέου block.
- **Nonce** : Ένας αριθμός που χρησιμοποιείται για τον αλγόριθμο εξόρυξης του νέου block.

Υπάρχουν 2 τρόποι για να προσδιορίσει κανείς ένα μπλοκ. 1) Μέσω του hash που προκύπτει από την εφαρμογή του αλγορίθμου SHA-256 στην κεφαλίδα του (η τιμή αυτή δεν περιέχεται στην δομή του μπλοκ ούτε στην κεφαλίδα του) και 2) από την θέση του στο blockchain η οποία ονομάζεται ύψος (block-height). Το πρώτο μπλοκ δηλαδή έχει ύψος 0 (genesis-block) . Το ύψος του μπλοκ επίσης δεν περιέχεται στη δομή του μπλοκ αλλά ούτε στην κεφαλίδα του.

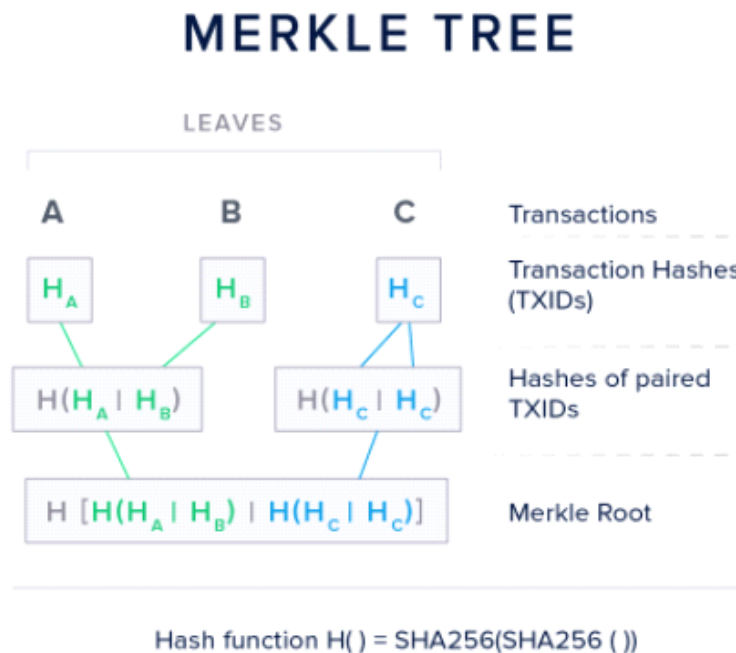


*Εικόνα 5 : Δομή ενός Block
πηγή : [11]*

Α'.11) Merkle Trees

Ένα merkle-tree είναι μία δομή δεδομένων , ένα δυαδικό δέντρο που χρησιμοποιείται σε αρκετές εφαρμογές της πληροφορικής και κυρίως στην τεχνολογία blockchain.

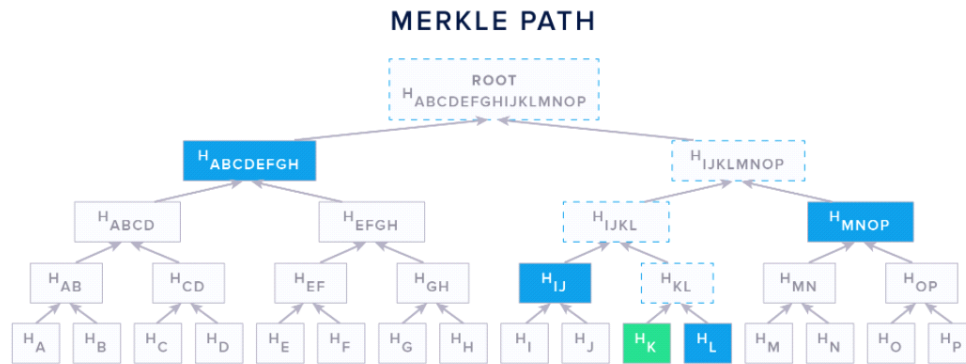
Είναι μία δομή λοιπόν που χρησιμοποιείται στο εσωτερικό των μπλοκ. Όλες οι συναλλαγές σε ένα μπλοκ συγκροτούν τα «φύλλα» του δέντρου. Η ρίζα του δέντρου, που ονομάζεται Merkle root, αντιπροσωπεύει την περίληψη από όλες τις συναλλαγές του μπλοκ και εμπεριέχεται στην κεφαλίδα του. Η κατασκευή του δουλεύει ως εξής: Η συναλλαγή που λειτουργεί σαν αμοιβή για την δημιουργία του μπλοκ (coinbase transaction) τοποθετείται πρώτη στα φύλλα του δέντρου , ακολουθούμενη από τις υπόλοιπες συναλλαγές του μπλοκ. Δημιουργείται μία τιμή κατακερματισμού εφαρμόζοντας 2 φορές τον αλγόριθμο SHA-256 για κάθε φύλλο (δηλαδή για κάθε συναλλαγή). Έπειτα οι τιμές κατακερματισμού 2 συναλλαγών ενώνονται και εισέρχονται σε μία άλλη συνάρτηση κατακερματισμού του ίδιου αλγορίθμου παράγοντας ένα ακόμα hash. Στην περίπτωση που ο αριθμός των συναλλαγών είναι περιττός , το hash της τελευταίας συναλλαγής ενώνεται με ένα αντίγραφο του εαυτού του. Αυτή η διαδικασία συνεχίζεται έως ότου να δημιουργηθεί ένα τελευταίο hash , το Merkle root.



Εικόνα 6 Η δομή ενός Merkle Tree [12]

A'.11.1) Merkle Path

Το Merkle path είναι το σύνολο των τιμών κατακερματισμού που είναι απαραίτητες προκειμένου να ανακατασκευαστεί το Merkle tree. Όπως φαίνεται στο παρακάτω σχήμα , το Merkle path της συναλλαγής K αποτελείται από το hash της συναλλαγής L $H(L)$ και έπειτα με τον συνδυασμό των $H(K,L)$, $H(M,N,O,P)$, $H(A,B,C,D,E,F,G)$. Αυτές οι 4 τιμές κατακερματισμού μαζί με την αρχική συναλλαγή επιτρέπουν την επαλήθευση της ακεραιότητας του δέντρου.



Εικόνα 7 : Παράδειγμα υπολογισμού του Merkle Path [12]

Α'.12) Ομοφωνία Blockchain (Consensus)

Βασική πτυχή της τεχνολογίας blockchain είναι ο προσδιορισμός ενός από τους συμμετέχοντες κόμβους που συμμετέχει στο δίκτυο, θα δημοσιεύσει το επόμενο μπλοκ καθώς και η διασφάλιση ότι όλοι οι κόμβοι έχουν τα ίδια δεδομένα. Αυτό το πρόβλημα επιλύεται με την εφαρμογή του ενός από τα πολλά πιθανά μοντέλα συναίνεσης(Consensus models) . Στα δημόσια δίκτυα blockchain (public – blockchain) υπάρχουν γενικά πολλοί κόμβοι που ανταγωνίζονται ταυτόχρονα για να δημοσιεύσουν το επόμενο μπλοκ. Απώτερος σκοπός τους είναι να κερδίσουν το έπαθλο του κρυπτονομίσματος μέσα από τους φόρους συναλλαγών που καταβάλλει κάθε αποστολέας συναλλαγής προκειμένου να πραγματοποιηθεί. Κάθε κόμβος που συμμετέχει στον διαγωνισμό πιθανότατα παρακινείται από μια επιθυμία για οικονομικό όφελος και όχι από την ευημερία του ίδιου του δικτύου. Στο δίκτυο blockchain που βασίζεται στους συμμετέχοντες για την επικύρωση συναλλαγών, την προσθήκη τους σε μπλοκ και την προσθήκη μπλοκ στο blockchain, όλοι οι κόμβοι προσπαθούν να επιτύχουν μια συμφωνία σχετικά με το τελευταίο μπλοκ που θα συμπεριληφθεί στην αλυσίδα. Ο μηχανισμός συναίνεσης διασφαλίζει την εγκυρότητα των συναλλαγών και την αντιγραφή τους με αποκεντρωμένο τρόπο χωρίς την παρέμβαση ενός κεντρικού ελεγκτικού μηχανισμού. Η συναίνεση είναι το πρώτο και σημαντικότερο χαρακτηριστικό του blockchain, το οποίο αναφέρεται στο σύνολο κανόνων και διαδικασιών που χρησιμοποιούνται για την ενημέρωσή του. Κατά τη διάρκεια αυτής της διαδικασίας , όλοι οι κόμβοι του δικτύου συνεργάζονται για να επιτευχθεί μία συμφωνία στην εγκυρότητα και στην σειρά των συναλλαγών που είναι υποψήφιος για να συμπεριληφθούν στο ledger. Οι αλγόριθμοι που χρησιμοποιούνται για την επίτευξη της ομοφωνίας διακρίνονται στους proof-based και στους voting-based. Στους πρώτους , οι κόμβοι που

συμμετέχουν στην διαδικασία καλούνται να αποδείξουν ότι είναι πιο κατάλληλοι από άλλους κόμβους για να επικυρώσουν και να καταγράψουν μία νέα συναλλαγή ή μπλοκ. Οι πιο γνωστοί αλγόριθμοι που ανήκουν σε αυτή την κατηγορία είναι οι Proof of Work (PoW) και Proof of Stake (PoS). Στους δεύτερους, όλο το σύνολο ή κάποιο υποσύνολο των κόμβων στο δίκτυο είναι αναγκαίο να ανταλλάξουν την απόφασή τους σχετικά με την επικύρωση ή την καταγραφή μιας νέας συναλλαγής ή μπλοκ προκειμένου να ληφθεί μία τελική απόφαση. Σε αυτή την κατηγορία χρησιμοποιούνται κυρίως οι αλγόριθμοι PBFT (Practical byzantine fault tolerance) και Ripple [13].

A'.12.1) PoW(Proof-of-Work)

Το PoW είναι μια στρατηγική συναίνεσης που χρησιμοποιείται σε πολλά κρυπτονομίσματα, ένα εκ των οποίων είναι το Bitcoin. Σε ένα αποκεντρωμένο δίκτυο, κάποιος κόμβος πρέπει να επιλεγεί για να καταγράψει τις συναλλαγές. Ο ευκολότερος τρόπος είναι η τυχαία επιλογή. Ωστόσο, η τυχαία επιλογή είναι ευάλωτη σε επιθέσεις. Έτσι, εάν ένας κόμβος θέλει να δημοσιεύσει ένα μπλοκ συναλλαγών, πρέπει να γίνει πολλή δουλειά για να αποδειχθεί ότι ο κόμβος δεν είναι πιθανό να επιτεθεί στο δίκτυο. Στο PoW, κάθε κόμβος του δικτύου που συμμετέχει στην διαδικασία δημιουργίας καινούργιο μπλοκ, υπολογίζει μια τιμή κατακερματισμού της κεφαλίδας μπλοκ. Η κεφαλίδα μπλοκ περιέχει ένα nonce δηλαδή μία μεταβλητή και οι κόμβοι που συμμετέχουν στην διαδικασία (miners) αλλάζουν συνεχώς το nonce για να λάβουν διαφορετικές τιμές κατακερματισμού. Το πρωτόκολλο απαιτεί ότι η υπολογιζόμενη τιμή κατακερματισμού πρέπει να είναι ίση ή μικρότερη από μια συγκεκριμένη δεδομένη τιμή. Όταν ένας κόμβος φτάσει την τιμή-στόχο, θα μεταδώσει το μπλοκ σε άλλους κόμβους και όλοι οι άλλοι κόμβοι πρέπει να επιβεβαιώσουν αμοιβαία την ορθότητα της τιμής κατακερματισμού. Εάν το μπλοκ επικυρωθεί, οι άλλοι miners θα προσθέσουν αυτό το νέο μπλοκ στα δικά τους blockchains. Οι κόμβοι που υπολογίζουν τις τιμές κατακερματισμού ονομάζονται miners και η διαδικασία PoW ονομάζεται mining στο Bitcoin. Η τιμή-στόχος μπορεί να τροποποιηθεί με την πάροδο του χρόνου έτσι ώστε προσαρμοστεί η δυσκολία (αύξηση ή μείωση) για να επηρεαστεί η ταχύτητα δημιουργίας μπλοκ. Για παράδειγμα, το Bitcoin, το οποίο χρησιμοποιεί το μοντέλο PoW, προσαρμόζει τη δυσκολία εύρεσης της τιμής-στόχου κάθε 2016 μπλοκ έτσι ώστε να επηρεάζει τον βαθμό δημοσίευσης μπλοκ ώστε να είναι περίπου μία φορά κάθε δέκα λεπτά. Αυτή η προσαρμογή γίνεται στην αλλαγή της τιμής-στόχου. Η μείωση της τιμής αναγκάζει τον miner σε περισσότερες επαναλήψεις της συνάρτησης κατακερματισμού άρα η δυσκολία αυξάνεται [14].

A'.12.2) PoS(Proof-of-Stake)

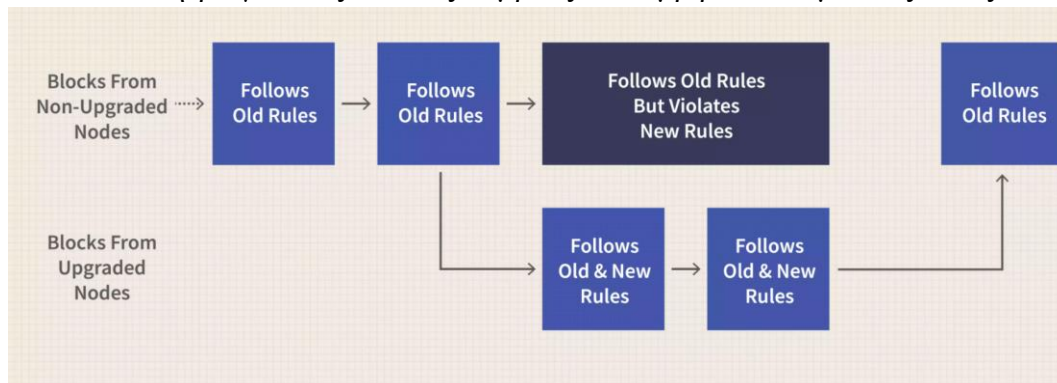
Είναι ο δεύτερος δημοφιλέστερος αλγόριθμος συναίνεσης που δημιουργήθηκε λόγω της υπερβολικής κατανάλωσης ενέργειας του Proof-of-Work και χρησιμοποιήθηκε για πρώτη φορά στο Peercoin το 2012. Η κύρια ιδέα του βασίζεται στο γεγονός ότι όποιος κόμβος επιθυμεί να συμμετάσχει στην δημιουργία ενός νέου μπλοκ επιβάλλεται να αποδείξει ότι κατέχει έναν προκαθορισμένο αριθμό νομισμάτων. Πρέπει να κλειδώσουν μια ορισμένη ποσότητα νομισμάτων που αναφέρεται ως stake σε ένα escrow λογαριασμό προκειμένου να συμμετέχουν στην διαδικασία. Το stake λειτουργεί ως διασφάλιση ότι θα συμπεριφερθεί σύμφωνα με τους κανόνες του πρωτοκόλλου. Ο κόμβος που διαφυλάσσει το stake με αυτόν τον τρόπο αναφέρεται ως stakeholder ή minter και στην περίπτωση που δεν τηρήσει τους κανόνες θα χάσει το stake. Στη συνέχεια επιλέγεται τυχαία ένας minter προκειμένου να δημιουργήσει το νέο μπλοκ. Μερικά από τα πλεονεκτήματά του είναι η εξοικονόμηση ενέργειας που υπερσπαταλείται με την χρήση του PoW καθώς στον συγκεκριμένο αλγόριθμο δεν απαιτείται η λύση σε κρυπτογραφικό γρίφο , μετατρέποντας το κρυπτονόμισμα που τον χρησιμοποιεί περισσότερο βιώσιμο μακροπρόθεσμα. Η ασφάλεια που διαθέτει αποτρέπει οποιαδήποτε παραβατική συμπεριφορά καθώς στην περίπτωση που κάποιος εμπλακεί σε μία τέτοια θα χάσει το μερίδιό του αλλά και την δυνατότητα να συμμετέχει σε οποιαδήποτε διαδικασία δημιουργίας μπλοκ στο μέλλον. Σε αντίθεση με τον PoW όπου ο δημιουργός του καινούργιου μπλοκ αμείβεται με το αντίστοιχο νόμισμα του blockchain , στον PoS οι stakeholder αμείβονται είτε από τους φόρους όλων των συναλλαγών που περιλαμβάνει το συγκεκριμένο μπλοκ είτε από το επιτόκιο

[15]

A'.13) Blockchain Forks

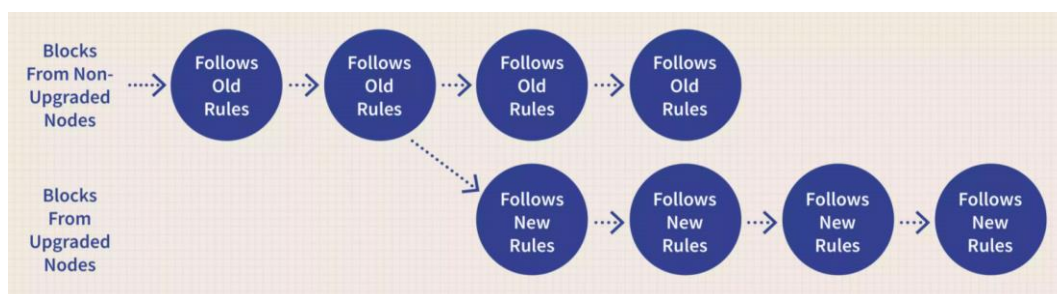
Η εκτέλεση αλλαγών και η ενημέρωση της τεχνολογίας σε ένα δίκτυο blockchain μπορεί να αποδειχθεί δύσκολη. Για ένα δημόσιο δίκτυο blockchain , το οποίο αποτελείται από πολλούς χρήστες, κατανεμημένους γύρω από το στον κόσμο, και διέπεται από τη συναίνεση των χρηστών, γίνεται εξαιρετικά δύσκολη. Γενικά ο όρος fork είναι μία ορολογία που χρησιμοποιείται για να δηλώσει την αποκοπή ή την εκτροπή από μία κυρίαρχη ή υπάρχουσα τεχνολογία, πλαίσιο , πολιτική, περιβάλλον. Μία από τις κύριες αιτίες ενός fork σε ένα blockchain δίκτυο είναι η δημοσίευση ενός νέου μπλοκ από 2 ή και περισσότερους miners οι οποίοι βρήκαν ταυτόχρονα την λύση του PoW. Επίσης οι αλλαγές σε ένα πρωτόκολλο του δικτύου ονομάζονται forks. Μία απλή ενημέρωση του λογισμικού λοιπόν είναι ικανή να οδηγήσει σε διάσπαση του δικτύου blockchain δημιουργώντας πολλαπλές εκδόσεις του ίδιου blockchain. Διακρίνεται σε δύο κατηγορίες: soft-forks και hard-forks. Στην περίπτωση του soft-fork οι κόμβοι που δεν έχουν ενημερώσει το λογισμικό τους είναι ακόμα ικανοί να συμμετέχουν στην επικύρωση και στην επιβεβαίωση συναλλαγών. Οι μη ενημερωμένοι κόμβοι είναι δηλαδή σε θέση να επικοινωνούν με

τους ενημερωμένους. Εάν κανένας ή πολύ λίγοι κόμβοι ενημερωθούν τότε οι νέοι κανόνες δεν θα τηρηθούν [16]. Ένα φανταστικό παράδειγμα ενός soft-fork θα ήταν εάν ένα blockchain αποφάσιζε να μειώσει το μέγεθος των μπλοκ (για παράδειγμα, από 1,0 MB έως 0,5 MB). Οι ενημερωμένοι κόμβοι θα προσαρμόσουν το μέγεθος του μπλοκ και θα συνεχίσουν να λειτουργούν ως συνήθως. Οι μη ενημερωμένοι κόμβοι θα βλέπουν αυτά τα μπλοκ ως έγκυρα από την αλλαγή καθώς δεν παραβιάζει τους κανόνες τους (δηλαδή, το μέγεθος του μπλοκ είναι κάτω από το μέγιστο επιτρεπόμενο επίπεδο). Ωστόσο, εάν ένας μη ενημερωμένος κόμβος δημιουργούσε ένα μπλοκ με μέγεθος μεγαλύτερο από 0,5 MB, οι ενημερωμένοι κόμβοι θα το απορρίψουν ως άκυρο. Το γεγονός αυτό αποτελεί κίνητρο για τους παλιούς κόμβους να συμβιβαστούν με τους νέους κανόνες.



Εικόνα 8 : Soft Fork [17]

Στην αντίθετη περίπτωση του hard fork, όλοι οι κόμβοι είναι αναγκαίο να ενημερώσουν το λογισμικό τους καθώς εκείνοι που δεν συμβιβάζονται με τις αλλαγές δεν θα είναι πλέον σε θέση να επικυρώσουν νέες συναλλαγές και να δημοσιεύσουν ένα νέο μπλοκ, αφού η έκδοση του λογισμικού τους είναι προγραμματισμένη με τέτοιο τρόπο ώστε να απορρίπτει οποιοδήποτε μπλοκ δεν είναι συμβατό με τους ίδιους «κανόνες». Το γεγονός αυτό συμβαίνει διότι η αναβάθμιση στην οποία οι κόμβοι καλούνται να επιβληθούν, δεν είναι συμβατή με παλιές εκδόσεις. Ένα παράδειγμα αυτού είναι η απόρριψη όλων των υπογήφγιων μπλοκ που δημιουργούνται από κόμβους που δεν έχουν ενημερωθεί.

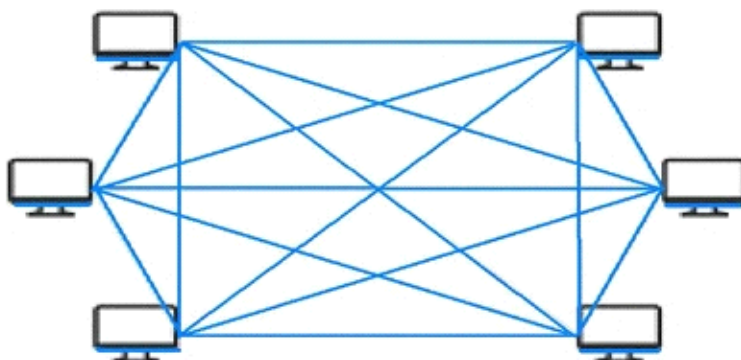


Εικόνα 9 : Hard Fork [18]

Α'.14) Τύποι Blockchain

Α'.14.1) Public

Ένα Public Blockchain έχει σχεδιαστεί για να είναι προσβάσιμο και επαληθεύσιμο από όλους τους κόμβους στο δίκτυο. Συγκεκριμένα, όλοι οι κόμβοι σε ένα δημόσιο δίκτυο blockchain μπορούν να επαληθεύσουν συναλλαγές, να διατηρήσουν ένα τοπικό αντίγραφο του blockchain και να δημοσιεύσουν ένα νέο block. Με την παροχή της εξουσίας διατήρησης ενός καθολικού αντιγράφου του blockchain σε όλους τους κόμβους, τα δημόσια blockchains κατανέμονται πλήρως. Ένα τέτοιο blockchain χρησιμοποιείται ευρέως σε ανώνυμες συναλλαγές. Ωστόσο, το σύστημα υποφέρει από τη χαμηλή ταχύτητα επικύρωσης των συναλλαγών και απαιτεί ορισμένο επίπεδο υπολογισμού για να διασφαλιστεί ότι δημιουργείται αμερόληπτο μπλοκ. Το Bitcoin είναι ένα από τα πιο δημοφιλή κρυπτονομίσματα που υποστηρίζονται από δημόσιο blockchain.

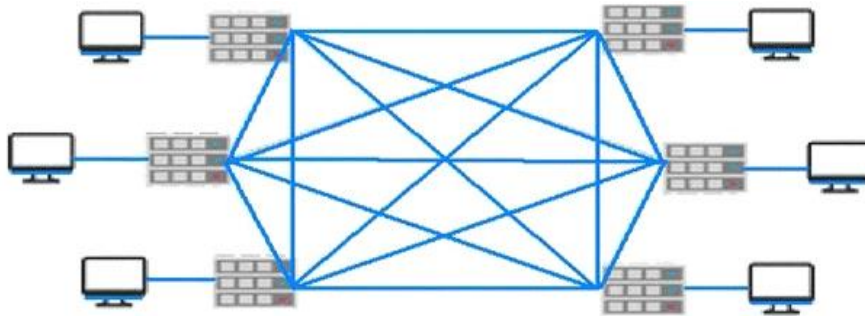


Εικόνα 10 : Public Blockchain Network [19]

Α'.14.2) Private

Ένα ιδιωτικό blockchain συνήθως διατηρείται από έναν μόνο οργανισμό. Τα δικαιώματα πρόσβασης στο blockchain και η επαλήθευση των συναλλαγών παραχωρούνται μέσω κεντρικού ελεγκτή στους επιτρεπόμενους κόμβους. Έτσι δημιουργείται ένα δίκτυο με δικαιώματα, στο οποίο μόνο οι εξουσιοδοτημένοι κόμβοι μπορούν να έχουν πρόσβαση σε ορισμένες συναλλαγές του blockchain ή να συμμετέχουν στην εργασία για τη δημοσίευση νέων μπλοκ. Με αυτόν τον τρόπο, το απόρρητο των συναλλαγών βελτιώνεται σημαντικά και η αποκέντρωση της εξουσίας επικύρωσης συναλλαγών τελεί υπό τον έλεγχο του οργανισμού. Επιπλέον, με υψηλό

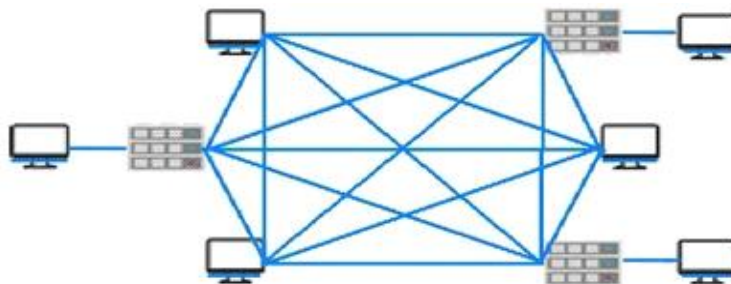
επίπεδο εμπιστοσύνης μεταξύ των κόμβων στο επιτρεπόμενο δίκτυο, δεν απαιτείται κάποιο μοντέλο συναίνεσης.



Εικόνα 11 : Private Blockchain Network [19]

Α'.14.3) Consortium

Η λειτουργία του είναι παρόμοια με ένα ιδιωτικό blockchain υπό την έννοια ότι διατηρούνται και τα δύο σε ένα δίκτυο με δικαιώματα. Η διαφορά είναι ότι σε ένα consortium blockchain, εμπλέκονται πολλοί οργανισμοί που μοιράζονται το δικαίωμα πρόσβασης και επικύρωσης των συναλλαγών. Αν και αυτοί οι οργανισμοί ενδέχεται να μην εμπιστεύονται πλήρως ο ένας τον άλλον, μπορούν να συνεργαστούν αλλάζοντας τον αλγόριθμο συναίνεσης βασιζόμενοι στο επίπεδο εμπιστοσύνης μεταξύ τους. Παρατηρείται δηλαδή ένα κράμα δημόσιων και ιδιωτικών κόμβων όπου μερικοί κόμβοι συμμετέχουν στις συναλλαγές και άλλοι στην επιτήρηση του δικτύου. Η πρόσβαση ενός κόμβου σε πληροφορίες εξαρτάται από τα δικαιώματα που έχει.



Εικόνα 12: Consortium Blockchain Network [19]

Α'.15) Πιθανές επιθέσεις σε Blockchain εφαρμογές

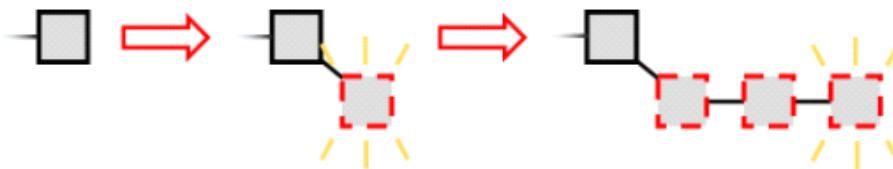
Το P2P δίκτυο πάνω στο οποίο υλοποιείται η τεχνολογία blockchain είναι ο βασικός παράγοντας διασφάλισης της ασφάλειας και της προσβασιμότητας. Ταυτόχρονα όμως είναι και ο λόγος που πολλές επιθέσεις που καταθέτονται παρακάτω γίνονται εφικτές.

Α'.15.1) Η επίθεση 51%

Η τεχνολογία blockchain βασίζεται στον μηχανισμό της καταναμημένης ομοφωνίας (consensus) για να καθιερώσει την αμοιβαία εμπιστοσύνη που απαιτείται. Παρόλα αυτά, ο μηχανισμός αυτός έχει την ευπάθεια που ονομάζεται «51%» η οποία μπορεί να εκμεταλλευτεί από επιτιθέμενους για να αποκτήσουν τον έλεγχο ολόκληρου του blockchain. Μία ομάδα από κόμβους που συνεργάζονται μέσω της υπολογιστικής τους δύναμης (hashing power) προκειμένου να επιλύσουν τον αλγόριθμο του μοντέλου ομοφωνίας, και να δημιουργήσουν ένα καινούργιο μπλοκ ονομάζεται mining pool. Ακριβέστερα, στα blockchain που εφαρμόζεται το PoW μοντέλο ομοφωνίας, αν ένα άτομο/ομάδα διαθέτει περισσότερο από το 51% όλης της υπολογιστικής δύναμης του δικτύου τότε η 51% επίθεση μπορεί να πραγματοποιηθεί. Ως εκ τούτου, η επεξεργαστική/υπολογιστική δύναμη που συσσωρεύεται σε ορισμένα mining pools δημιουργεί τον φόβο για μία ακούσια κατάσταση, όπως το γεγονός μία ομάδα να ελέγχει περισσότερο από την μισή της ολόκληρης υπολογιστικής δύναμης. Στις εφαρμογές blockchain που χρησιμοποιούν το μοντέλο PoS, η συγκεκριμένη επίθεση μπορεί να επιτευχθεί αν ο αριθμός των νομισμάτων που έχει στην κατοχή του ένα άτομο ξεπερνάει το 50% όλων των νομισμάτων. Πραγματοποιώντας την 51% επίθεση ένας επιτιθέμενος μπορεί να τροποποιήσει την πληροφορία που είναι καταγεγραμμένη στο blockchain. Πιο συγκεκριμένα ο επιτιθέμενος μπορεί να εκμεταλλευτεί αυτή την ευπάθεια για να επιτελέσει τις παρακάτω επιθέσεις: Αντιστροφή των συναλλαγών και υλοποίηση του double-spending (τα ίδια κέρματα να ξοδεύονται πολλές φορές), τροποποίηση της σειράς των συναλλαγών, παρεμπόδιση της διαδικασίας mining σε miners, εμπόδιση της λειτουργίας της επιβεβαίωσης των κανονικών συναλλαγών [20].

Α'.15.2) Selfish Mining Attack

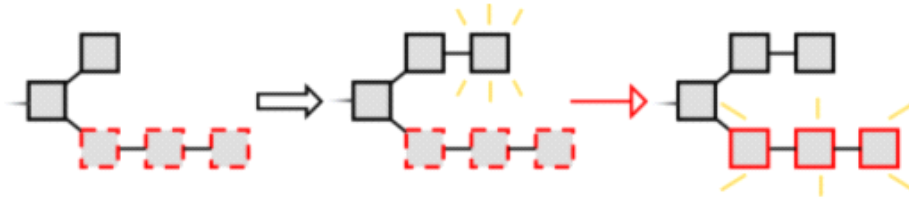
Η συγκεκριμένη επίθεση διεξάγεται από επιτιθέμενους (selfish-miners) με σκοπό την απόκτηση αδικαιολόγητων ανταμοιβών ή τη σπατάλη της υπολογιστικής δύναμης των τίμιων miners. Ο κακόβουλος χρήστης δημιουργεί ένα καινούργιο μπλοκ, το κρατάει ιδιωτικά αντί να το δημοσιεύσει και έπειτα προσπαθεί να δημιουργήσει μία ιδιωτική αλυσίδα από μπλοκ.



Εικόνα 13 : self-mining attack [21]

Ταυτόχρονα οι «τίμιοι» miners προσπαθούν να επεκτείνουν την δημόσια αλυσίδα. Ο επιτιθέμενος συνεχίζει να κρατά την ιδιωτική του αλυσίδα από μπλοκ ιδιωτική μέχρι

η δημόσια να βρίσκεται ένα μπλοκ πίσω. Τότε δημοσιεύει την ιδιωτική αλυσίδα, η οποία είναι μακρύτερη, με συνέπεια οι κόμβοι να ακολουθούν πλέον την αλυσίδα που έχτισε ο ίδιος. Τα μπλοκ που δημιουργήθηκαν από τους τίμιους miners απορρίπτονται και δεν επιφέρουν ανταμοιβή στους δημιουργούς.



Εικόνα 14 : self-mining attack [21]

A'.15.3) Eclipse Attack

Οι επιθέσεις Eclipse περιλαμβάνουν έναν κακόβουλο χρήστη του δικτύου που απομονώνει έναν συγκεκριμένο χρήστη ή κόμβο μέσα σε δίκτυο P2P. Κατά την εκτέλεση μιας τέτοιας επίθεσης, ο εισβολέας προσπαθεί να ανακατευθύνει τις εισερχόμενες και εξερχόμενες συνδέσεις του χρήστη-στόχου μακριά από τους κανονικούς γειτονικούς κόμβους του σε κόμβους που ελέγχονται από αυτόν, εγκλωβίζοντας έτσι τον στόχο σε ένα περιβάλλον που είναι εντελώς ξεχωριστό από την πραγματική δραστηριότητα του δικτύου. Αποκρύπτοντας τη νόμιμη τρέχουσα κατάσταση του βιβλίου blockchain, ο εισβολέας μπορεί να χειριστεί τον απομονωμένο κόμβο με διάφορους τρόπους που μπορεί να οδηγήσει σε παράνομες επιβεβαιώσεις συναλλαγών. Επειδή οι επιθέσεις Eclipse βασίζονται στην εκμετάλλευση των γειτονικών κόμβων ενός στόχου, η ευκολία με την οποία αυτές οι επιθέσεις μπορούν να εκτελεστούν με επιτυχία εξαρτάται σε μεγάλο βαθμό από την δομή και τα πρωτόκολλα ασφαλείας του δικτύου blockchain στόχου [22].

A'.15.4) Double Spending Attack

Η συγκεκριμένη επίθεση είναι ο κίνδυνος να δαπανηθεί ένα ψηφιακό νόμισμα δύο φορές. Είναι ένα δυνητικό πρόβλημα μοναδικό για τα ψηφιακά νομίσματα, επειδή οι ψηφιακές πληροφορίες μπορούν να αναπαραχθούν σχετικά εύκολα από κακόβουλα άτομα που κατανοούν την υποδομή του δικτύου blockchain και την υπολογιστική ισχύ που είναι απαραίτητη για να το χειριστούν. Τέτοια επίθεση είναι εφικτή στην περίπτωση της επίθεσης 51% όπου ο χρήστης ελέγχει την αλυσίδα των μπλοκ.

A'.15.5) Liveness Attack

Στόχος της επίθεσης είναι να καθυστερήσει όσο το δυνατόν περισσότερο τον χρόνο επιβεβαίωσης μιας συναλλαγής –στόχου και αποτελείται από τρεις φάσεις, τη φάση προετοιμασίας της επίθεσης, της άρνησης συναλλαγών και τη φάση επιβράδυνσης blockchain.

- **Προετοιμασίας επίθεσης.**

Ακριβώς όπως η self-mining επίθεση, ένας επιτιθέμενος δημιουργεί πλεονέκτημα έναντι των «τίμιων» miners με κάποιο τρόπο προτού μεταδοθεί η στοχευόμενη συναλλαγή TX στη δημόσια αλυσίδα. Ο επιτιθέμενος χτίζει την ιδιωτική αλυσίδα, η οποία είναι μεγαλύτερη από τη δημόσια αλυσίδα.

- **Άρνησης συναλλαγής.**

Ο εισβολέας κρατά ιδιωτικά το μπλοκ που περιέχει την συναλλαγή TX, προκειμένου να αποτραπεί η εγγραφή της στη δημόσια αλυσίδα.

- **Επιβράδυνσης blockchain.**

Όσο προστίθενται μπλοκ στην δημόσια αλυσίδα, η συναλλαγή δεν θα μπορεί πλέον να κρατείται ιδιωτικά σε συγκεκριμένο χρόνο. Σε αυτήν την περίπτωση, ο εισβολέας θα δημοσιεύσει το μπλοκ που περιέχει την συναλλαγή. Σε ορισμένα συστήματα blockchain, όταν το βάθος του μπλοκ που περιέχει την συναλλαγή είναι μεγαλύτερο από ένα σταθερό, το TX θα θεωρείται έγκυρο. Επομένως, ο επιτιθέμενος θα συνεχίσει να χτίζει ιδιωτική αλυσίδα προκειμένου να δημιουργήσει πλεονέκτημα έναντι της δημόσιας αλυσίδας. Μετά από αυτό, ο επιτιθέμενος θα δημοσιεύσει τα ιδιωτικά της μπλοκ σε δημόσια αλυσίδα στον κατάλληλο χρόνο για να επιβραδύνει τον ρυθμό ανάπτυξης της δημόσιας αλυσίδας. Η επίθεση θα τελειώσει όταν το TX επαληθευτεί ως έγκυρο στη δημόσια αλυσίδα.

A'.15.6) Denial of Service

Μία από τις πιο συνηθισμένες επιθέσεις σε διαδικτυακές υπηρεσίες είναι η επίθεση κατανεμημένης άρνησης υπηρεσίας (DDoS). Η τεχνολογία Blockchain, παρά το γεγονός ότι είναι peer-to-peer σύστημα, εξακολουθεί να είναι επιρρεπής σε επιθέσεις DDoS. Εφαρμογές που βασίζονται σε blockchain, όπως το Bitcoin και το Ethereum, έχουν υποστεί επανειλημμένα από αυτές τις επιθέσεις. Οι επιθέσεις DDoS εκδηλώνονται με διάφορους τρόπους, ανάλογα με τη φύση της εφαρμογής, την αρχιτεκτονική του δικτύου και τη συμπεριφορά των κόμβων του δικτύου. Για παράδειγμα, στο δίκτυο Bitcoin, η επίθεση 51% μπορεί να οδηγήσει σε άρνηση υπηρεσίας εάν μια ομάδα miners αποκτήσει σημαντική δύναμη κατακερματισμού, μπορεί να αποτρέψει άλλους miners από το να προσθέσουν τα μπλοκ τους στο Blockchain, να ακυρώσουν τις τρέχουσες συναλλαγές και να προκαλέσουν αποτυχία υπηρεσίας στο δίκτυο.

Κεφάλαιο Β΄

Β΄)Ethereum

Σε αυτό το κεφάλαιο περιγράφονται οι ιδιαίτερες τεχνολογίες που διαφοροποιούν το Ethereum Blockchain από τα υπόλοιπα blockchain και γίνεται μία λεπτομερής ανάλυση των προτύπων (token) που χρησιμοποιεί.

Β΄.1) Ορισμός

Το Ethereum ανήκει και αυτό στην γενική κατηγορία των blockchain , βασίζεται δηλαδή σε ένα Peer-to-peer δίκτυο που συμμετέχουν πολλοί κόμβοι. Ο κύριος ρόλος του δεν περιορίζεται στις αποκεντρωμένες πληρωμές . Η διαφορά του με τα υπόλοιπα blockchains είναι ότι δίνει την δυνατότητα στους χρήστες του να εκτελέσουν κομμάτια κώδικα (smart contracts) , ο οποίος αποθηκεύεται στο blockchain ώστε να μπορεί ο καθένας να αλληλεπιδράσει με αυτόν. Στον κόσμο του Ethereum υπάρχει ένας υπολογιστής (Ethereum Virtual Machine) για την κατάσταση του οποίου όλοι οι κόμβοι συμφωνούν μεταξύ τους. Όλοι οι κόμβοι διαθέτουν ένα αντίγραφο από την κατάσταση αυτού του υπολογιστή και κάθε ένας έχει την δυνατότητα να διαδώσει αιτήματα συναλλαγών προς αυτόν και να προκαλέσουν την αλλαγή της κατάστασής του. Το κρυπτονόμισμα που χρησιμοποιείται σε ένα δίκτυο Ethereum ονομάζεται Ether (ETH) και υποδιαιρείται σε Finney, Szabo, Gwei, Mwei, Kwei, και Wei όπου το τελευταίο είναι η μικρότερη μονάδα Eth. Ο αλγόριθμος ομοφωνίας που χρησιμοποιεί είναι ο Proof-of-work.

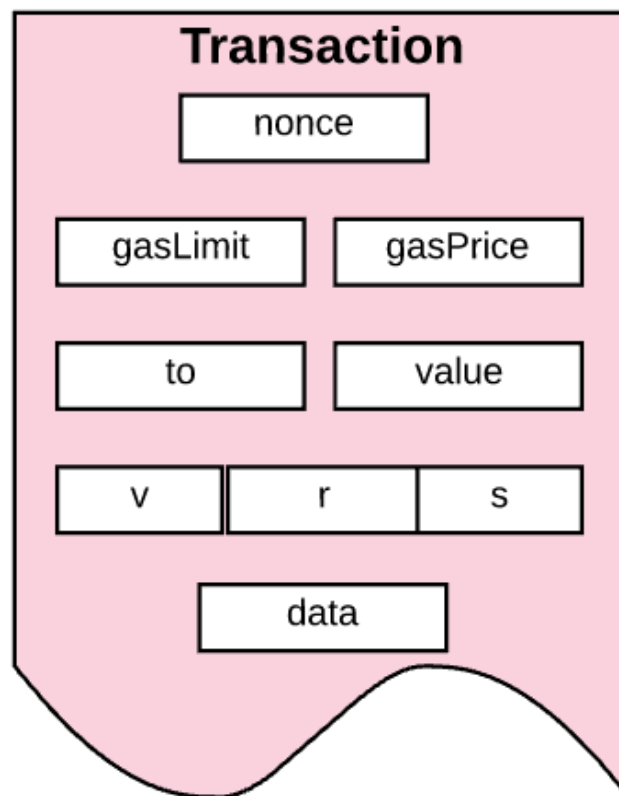
Β΄.2) Συναλλαγές

Η συναλλαγή σε ένα δίκτυο Ethereum υποδεικνύει την αρχικοποίηση μιας πράξης από έναν λογαριασμό που ανήκει σε έναν χρήστη και όχι σε ένα smart contract. Κάθε συναλλαγή που πραγματοποιείται αλλάζει την κατάσταση του EVM και είναι να αναγκαίο να διαδοθεί σε όλο το δίκτυο. Οι συναλλαγές σε ένα δίκτυο Ethereum απαιτούν έναν "φόρο" του οποίου η μονάδα μέτρησης είναι το Gas και η αξία του υπολογίζεται σε eth. Οποιαδήποτε συναλλαγή περιέχει τις εξής πληροφορίες :

- **Receipient** : η διεύθυνση του παραλήπτη είτε πρόκειται για διεύθυνση χρήστη είτε για smart contract.
- **Value** : η ποσότητα eth που πρόκειται να μεταφερθεί.
- **Data** : Προαιρετικό πεδίο αυθαίρετων δεδομένων.
- **gasLimit** : Η μέγιστη ποσότητα Gas που θα καταναλωθεί από την συναλλαγή.

- **MaxPriorityFeePerGas** : Η μέγιστη ποσότητα Gas που θα συμπεριληφθεί σαν φιλοδώρημα στον miner.
- **MaxFeePerGas** : Η μέγιστη ποσότητα Gas που ο χρήστης προτίθεται να πληρώσει για την συναλλαγή συμπεριλαμβανομένου των Base fee + MaxPriorityFeePerGas.
- **Nonce** : Αριθμός των συνολικών επιβεβαιωμένων συναλλαγών μιας διεύθυνσης.
- **v,r,s** : Τα κύρια συστατικά μιας ECDSA ψηφιακής υπογραφής του χρήστη που δημιουργεί την συναλλαγή.

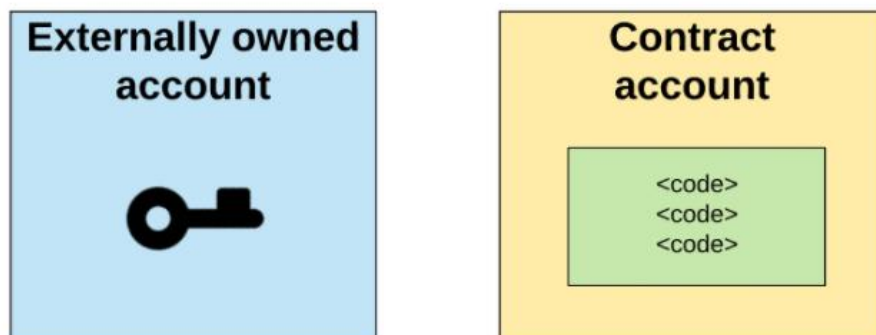
Οι συναλλαγές που δημιουργούν ένα smart contract ανήκουν σε μία ειδική κατηγορία συναλλαγών και ονομάζονται "Contract Creation". Η διαφοροποίησή τους έγκειται στο γεγονός ότι ο παραλήπτης αυτών των συναλλαγών είναι μία ειδική διεύθυνση με το όνομα "Zero Address".



Εικόνα 15 : Περιεχόμενα συναλλαγής Ethereum [23]

Β'.3) Λογαριασμοί

Στον "κόσμο" του Ethereum υπάρχουν δύο ειδών λογαριασμοί. Αυτοί που ανήκουν σε χρήστες (Externally Owned Accounts) και οι "contract accounts", οι οποίοι ελέγχονται από τον κώδικα προγραμμάτων (smart contracts). Κοινά στοιχεία είναι η αναγνώριση από μία μοναδική διεύθυνση , η ικανότητα αλληλεπίδρασής τους με άλλα smart contracts και η λήψη, μεταφορά, διατήρηση eth/tokens. Σε αντίθεση με τους λογαριασμούς χρηστών , δεν υπάρχει ζεύγος ιδιωτικού/δημοσίου κλειδιού που να συνδέεται με ένα smart contract.



Εικόνα 16 : Τύποι λογαριασμών [23]

Β'.4) Smart Contracts

Ο όρος αυτός χρησιμοποιείται για την περιγραφή αμετάβλητων υπολογιστικών προγραμμάτων που εκτελούνται από το EVM(Ethereum Virtual Machine). Τα smart contracts είναι απλά ή σύνθετα προγράμματα αποθηκευμένα σε συγκεκριμένες διευθύνσεις σε ένα blockchain , τα οποία εκτελούνται μόνο όταν πληρούνται ορισμένες προϋποθέσεις. Χρησιμοποιούνται κυρίως για την πραγματοποίηση συναλλαγών και συμφωνιών όπου οι συμμετέχοντες γνωρίζουν το αποτέλεσμα χωρίς την μεσολάβηση ενός τρίτου παράγοντα , εξοικονομώντας με αυτόν τον τρόπο αρκετό χρόνο. Η γλώσσα προγραμματισμού των smart contracts ονομάζεται Solidity . Προκειμένου να εκτελεστούν , είναι αναγκαία η μεταγλώττιση του κώδικα σε χαμηλού επιπέδου bytecode που εκτελείται στο EVM . Η δημοσίευσή τους στο blockchain πραγματοποιείται μετά την μεταγλώττισή τους και με την δημιουργία μιας ειδικού τύπου συναλλαγής που ονομάζεται " contract creation transaction" . Η μοναδική διεύθυνση ενός smart contract μπορεί να χρησιμοποιηθεί σαν παραλήπτης , όπου κάποιος μπορεί να στείλει ένα κεφάλαιο σε αυτό ή καλώντας απλά μία από τις συναρτήσεις . Ο κώδικας που περιέχουν εκτελείται μόνο στην περίπτωση που κάποιος χρήστης δημιουργήσει μία συναλλαγή με αυτό. Ένα smart contract είναι ικανό να καλέσει πολλά άλλα δημοσιοποιημένα smart contracts δημιουργώντας μία αλυσίδα συναλλαγών όπου δημιουργός της θα είναι πάντα κάποιος χρήστης. Ο κώδικας που περιέχουν παραμένει αμετάβλητος από την στιγμή της δημοσιοποίησής τους στο ethereum blockchain , παρόλα αυτά είναι εφικτή η διαγραφή του κώδικα που περιέχουν και οποιαδήποτε συναλλαγή περιέχει την διεύθυνσή τους δεν συνεπάγει την εκτέλεση κώδικα. Εκτελούνται από όλους του κόμβους του δικτύου

blockchain και αφού πιστοποιηθεί η εγκυρότητά τους και πραγματοποιηθούν , η συναλλαγή καταγράφεται για πάντα [24] .

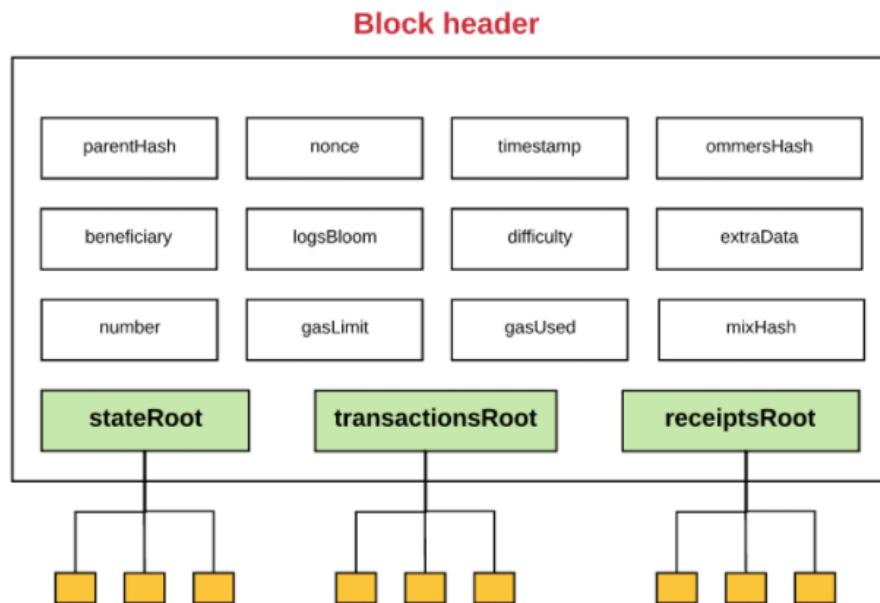
B'.5) Blocks

Τα blocks είναι δομές δεδομένων που περιέχουν όλες τις συναλλαγές που πραγματοποιούνται. Κάθε μπλοκ συνδέεται με το προηγούμενό του περιέχοντας στην κεφαλίδα του το Hash του προηγούμενου block το οποίο υπολογίζεται από τα δεδομένα που περιέχει . Αυτή η τεχνική καθιστά την παραποίηση κάποιου μπλοκ ανέφικτη καθώς στην περίπτωση που κάποιο block παραποιηθεί τότε θα παραποιηθούν και όλα τα επακόλουθα block. Οι miners είναι υπεύθυνοι για την δημιουργία νέων block μέσω του Proof-of-work. Ο νικητής είναι εκείνος που ο υπολογιστής του επιλύσει έναν μαθηματικό γρίφο. Έπειτα διαδίδει το καινούργιο block σε όλο το δίκτυο ώστε όλοι οι κόμβοι να το προσθέσουν στο τέλος του blockchain τους και λαμβάνει το βραβείο του σε Eth. Στο κύριο δίκτυο Ethereum ο μέσος όρος δημιουργίας ενός νέου block είναι 12-14 δευτερόλεπτα. Κάθε block έχει μέγεθος 15 εκατομμύρια Gas αλλά το μέγεθός του είναι μεταβλητό με το ανώτατο όριο να είναι 30 εκατομμύρια gas ($2 * \text{block size}$). Ένα block αποτελείται από το block header , πληροφορίες για όλες τις συναλλαγές που περιέχονται σε αυτό και ένα σύνολο από block headers από τα ommer block αυτού.

Ένα block header αποτελείται από [25] :

- **parent hash** : Το hash από το προηγούμενο block header
- **beneficiary** : Η διεύθυνση από τον λογαριασμό που θα λάβει τα gas fees
- **ommersHash** : Το hash από την λίστα των ommer blocks
- **stateRoot** : Το hash του κόμβου-ρίζα από το state tree
- **transactionsRoot** : Το hash από τον κόμβο-ρίζα από την δομή trie που περιέχει όλες τις συναλλαγές που βρίσκονται στο block
- **nonce** : Το hash το οποίο σε συνδυασμό με το mixHash αποδुकνύει τους υπολογισμούς που χρειάστηκαν για την κατασκευή του block
- **mixHash** : Το hash το οποίο σε συνδυασμό με το nonce αποδεικνύει τους υπολογισμούς που χρειάστηκαν για την κατασκευή του block
- **gasLimit** : Το όριο Gas του block
- **gasUsed**: Το σύνολο από το όλο το Gas που καταναλώθηκε από τις συναλλαγές του block
- **number** : Ο αριθμός ακολουθίας του block ξεκινώντας από το Genesis block
- **difficulty** : Το επίπεδο της δυσκολίας του block
- **extraData** : Δεδομένα που συνδέονται με το block

- **receiptsRoot** : Το hash από τον κόμβο ρίζα από την trie δομή που περιέχει τις αποδείξεις από όλες τις συναλλαγές του block
- **timestamp** : Ο χρόνος της έναρξης του block
- **logsBloom** : Πεδίο που υποδεικνύει ένα event για το block header.

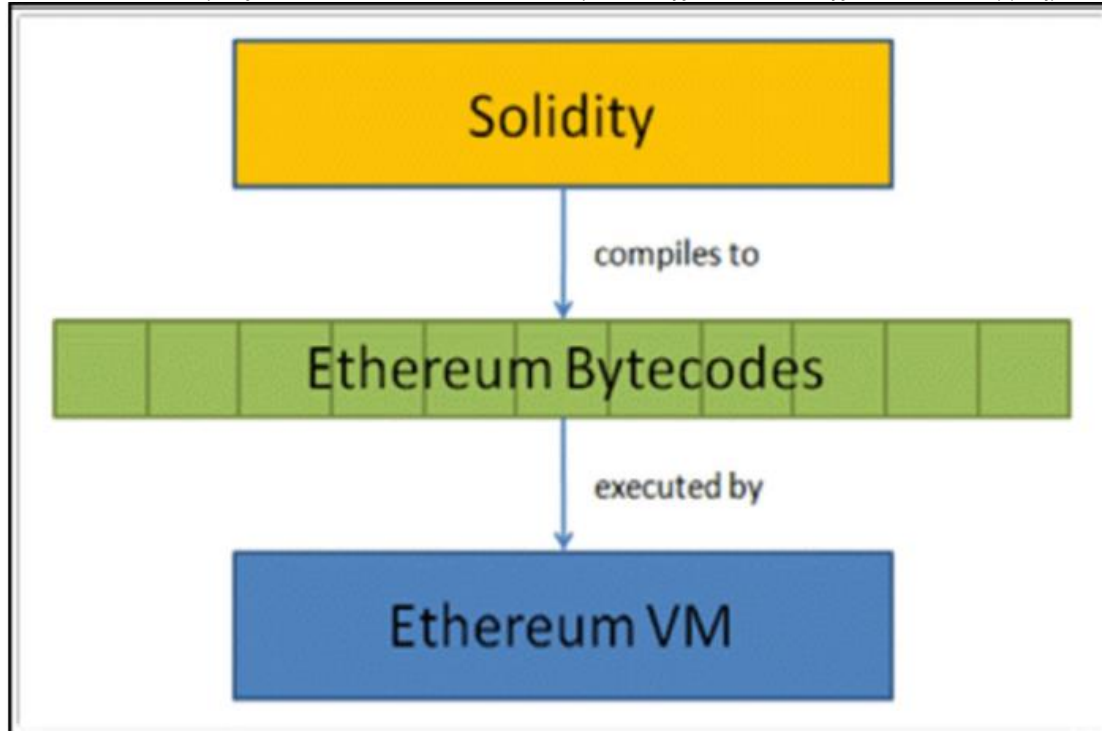


Εικόνα 17 : Block header [26]

B'.6) EVM

Το Ethereum Virtual Machine (EVM) είναι μία πλατφόρμα λογισμικού , ενσωματωμένο σε κάθε κόμβο ενός Ethereum δικτύου , δίνει την δυνατότητα σε προγραμματιστές να δημιουργήσουν Dapps(Decentralized applications) και η κύρια λειτουργία είναι η εκτέλεση του κώδικα bytecode των smart contracts. Τα smart contracts γράφονται σε υψηλού επιπέδου προγραμματιστικές γλώσσες όπως η Solidity και έπειτα μεταγλωττίζονται σε Bytecode. Υπάρχουν πετυχημένες υλοποιήσεις του σε πολλές γλώσσες προγραμματισμού όπως python, JavaScript , C++ , Ruby και άλλες. Κάθε κόμβος «τρέχει» ένα παράδειγμα EVM , γεγονός που τους επιτρέπει να εκτελούν όλοι τις ίδιες εντολές. Λόγω αυτού το Ethereum χαρακτηρίζεται μερικές φορές ως παγκόσμιος υπολογιστής. Το EVM είναι “Turing complete” το οποίο υποδηλώνει ένα σύστημα που έχει την δυνατότητα να επιλύσει ένα υπολογιστικό πρόγραμμα . Συνεπώς παίζει καθοριστικός ρόλο για την ορθή λειτουργία Ethereum πρωτόκολλου καθώς διατηρεί την ομοφωνία (consensus) στο blockchain. Δίνει την δυνατότητα σε οποιονδήποτε να εκτελέσει κώδικα σε ένα οικοσύστημα χωρίς εμπιστοσύνη

στο οποίο όμως το τελικό αποτέλεσμα της εκτέλεσης είναι εγγυημένο.



Εικόνα 18 : Μεταγλώττιση υψηλού επιπέδου γλώσσας και εκτέλεση Bytecode [27]

B'.7) Gas

Το Gas είναι μια μονάδα μέτρησης της υπολογιστικής δύναμης που απαιτείται για την εκτέλεση διεργασιών σε ένα δίκτυο Ethereum. Οποιαδήποτε συναλλαγή απαιτεί έναν μικρό "φόρο" από τον χρήστη που την πραγματοποιεί και πληρώνονται με νόμισμα του ethereum (eth) . Προκειμένου να συμπεριληφθεί μια συναλλαγή στο επόμενο μπλοκ , έχει καθιερωθεί ένα μικρό ποσό το οποίο ο αποστολέας είναι υποχρεωμένος να πληρώσει και υπολογίζεται από την ζήτηση για την συμπερίληψη σε αυτό(Base fee). Λόγω του ότι το μέγεθος των μπλοκ πλέον είναι μεταβλητό και όχι στατικό , όταν το μέγεθος ενός μπλοκ ξεπεράσει το βασικό όριο(15m gas) τότε αυξάνεται ο βασικός φόρος για το επόμενο μπλοκ. Με τον ίδιο τρόπο μειώνεται στην περίπτωση που το μέγεθος είναι μικρότερο από το όριο. Ο χρήστης αποφασίζει την αξία κάθε μονάδας Gas σε eth(Gas price). Όσο μεγαλύτερη είναι η αξία , τόσο πιο πιθανό η συναλλαγή να βρίσκεται στο επόμενο μπλοκ καθώς αυξάνεται η σημαντικότητα της συναλλαγής και το ποσό που θα ληφθεί από τους miners. Το μέγιστο ποσό σε Gas που ο χρήστης είναι διατεθειμένος να πληρώσει ονομάζεται Gas Limit(minimum 21.000) . Ο υπολογισμός λοιπόν των επιπλέον χρημάτων που πληρώνει ο κάθε χρήστης βασίζεται σε αυτόν τον πολλαπλασιασμό : $\text{Gas units (limit)} * (\text{Base fee} + \text{Tip})$. Στην περίπτωση το υπόλοιπο χρημάτων του χρήστη δεν επαρκούν για να καλύψουν την τιμή του απαιτούμενου gas , η συναλλαγή θεωρείται μη έγκυρη και οποιαδήποτε αλλαγή υπήρξε αναιρείται. Από τη στιγμή που διεξάχθηκαν υπολογισμοί , το gas που ξόδεψε ο χρήστης δεν επιστρέφεται.

Ο χρήστης έχει την δυνατότητα να καθορίσει το μέγιστο ποσό που διατίθεται να πληρώσει για αυτούς τους φόρους χρησιμοποιώντας την παράμετρο maxFeePerGas , με την προϋπόθεση ότι θα υπερβαίνει το άθροισμα του βασικού φόρου με το φιλοδώρημα. Η διαφορά του max fee με το άθροισμα base fee+ tip επιστρέφεται. Η τεχνολογία του Gas

ενισχύει την ασφάλεια του δικτύου καθώς αποτρέπει κακόβουλους χρήστες να δημιουργούν αμέτρητες συναλλαγές χωρίς κόστος χρημάτων [28] .

B'.8) Dapps

Ο όρος Dapp (Decentralized Application) χρησιμοποιείται για την κατηγοριοποίηση αποκεντρωμένων εφαρμογών που χρησιμοποιούν ένα blockchain ως backend. Μερικά από τα πλεονεκτήματά τους είναι η παροχή της απόλυτης διαφάνειας καθώς οποιαδήποτε αλληλεπίδραση μαζί του θα αποθηκευτεί στο blockchain , η ανθεκτικότητα σε επιθέσεις λόγω του ότι το back-end της εφαρμογής θα είναι καταναμημένο στο blockchain σε αντίθεση με τις υπόλοιπες εφαρμογές που χρησιμοποιούν έναν κεντρικό server, διασφαλίζεται η ιδιωτικότητα των χρηστών διότι οι χρήστες δεν είναι υποχρεωμένοι να παρέχουν στην εφαρμογή τα στοιχεία τους για αλληλεπιδράσουν με αυτή , κανείς δεν μπορεί να περιορίσει την πρόσβασή σου στην εφαρμογή , οι πληρωμές πραγματοποιούνται με το εκάστοτε κρυπτονόμισμα με αποτέλεσμα να μην απαιτείται η κοινοποίηση προσωπικών στοιχείων , οποιοσδήποτε είναι ικανός να χρησιμοποιήσει την εφαρμογή καθώς δεν ελέγχεται από καμία κεντρική αρχή. Λόγω του περιορισμού αποθήκευσης μεγάλου όγκου δεδομένων σε ένα smart contract , τέτοιου τύπου εφαρμογές χρησιμοποιούν κεντρικούς (cloud database) ή και αποκεντρωμένους server (IPFS) . Τέλος η χρήση ενός ή και πολλών smart contract έχει τον ρόλο του backend [29] .

B'.9) Δίκτυα Ethereum

Υπάρχουν πολλά δίκτυα που εφαρμόζουν το πρωτόκολλο του Ethereum και η χρησιμότητά τους ποικίλει καθώς μπορούν να χρησιμοποιηθούν για ανάπτυξη , δοκιμή και για δημοσιοποίηση εφαρμογών. Μία διεύθυνση ενός χρήστη είναι διαθέσιμη σε όλα τα δίκτυα που υπάρχουν αλλά το υπόλοιπο του λογαριασμού μαζί με το ιστορικό συναλλαγών διαφέρουν σε κάθε δίκτυο. Υπάρχουν 3 είδη δικτύων. Public , Private , Consortium . Το δίκτυο Mainnet είναι το κύριο δημόσιο δίκτυο του ethereum όπου οι συναλλαγές μεταφέρουν κανονικά χρήματα και καταγράφονται στο κύριο ethereum blockchain. Σε αντίθεση με το Mainnet υπάρχουν πολλά δημόσια testnets των οποίων η κύρια χρησιμότητά τους είναι η δοκιμή και η ανάπτυξη εφαρμογών και smart contracts πριν γίνει η δημοσιοποίησή τους στο Mainnet. Τα δημοφιλέστερα testnet είναι το Kovan , Ropsten , Rinkeby , Görli .

B'.10) Tokens

Το Ethereum έχει αναπτύξει ορισμένα πρότυπα τα οποία στοχεύουν στην διαλειτουργικότητα των εφαρμογών απέναντι σε άλλες υλοποιήσεις. Αυτά τα πρότυπα ονομάζονται tokens και στόχος τους είναι η αναπαράσταση ψηφιακών και μη στοιχείων. Στην ουσία ένα token είναι ένα smart contract. Τέτοια tokens δημιουργούνται πάνω από άλλες πλατφόρμες , μπορούν να πωληθούν , να αγοραστούν , να ανταλλαχτούν και διαφέρουν από τα κρυπτονομίσματα καθώς δεν έχουν το δικό τους blockchain και εξαρτώνται από άλλα. Υπάρχουν 4 πρότυπα token στο Ethereum δίκτυο: ERC-20 , ERC-721 , ERC-777 , ERC-1155 [30] .

B'.10.1) ERC-20

Είναι ένα από τα σημαντικότερα tokens στο στο ethereum δίκτυο. Στην πράξη πρόκειται για ένα smart contract το οποίο με τις συναρτήσεις που παρέχει δίνει την δυνατότητα στους χρήστες να ανταλλάζουν tokens , να βρουν το υπόλοιπο σε token ενός λογαριασμού , να μάθουν για την προσφορά του token όλου του δικτύου αλλά και να εγκρίνουν την συναλλαγή token ενός λογαριασμού από έναν άλλο λογαριασμό. Τα κρυπτογραφικά tokens εκπροσωπούν δικαιώματα πρόσβασης σε κάποια υποκείμενη οικονομική αξία ή σε ένα σύνολο δικαιωμάτων στον ψηφιακό και φυσικό κόσμο .

Στην περίπτωση που ένα smart contract υλοποιεί τις συγκεκριμένες συναρτήσεις του ERC-20 προτύπου τότε μπορεί να μετονομαστεί σε ERC-20 Token Contract και αφού δημοσιευτεί στο δίκτυο έχει την ικανότητα να παρακολουθεί όλα τα συγκεκριμένα tokens στο ethereum.

Οι συναρτήσεις που υλοποιούνται σε ένα ERC-20 Token smart contract είναι οι εξής [31]:

- `function name() public view returns (string)`
- `function symbol() public view returns (string)`
- `function decimals() public view returns (uint8)`
- `function totalSupply() public view returns (uint256)`
- `function balanceOf(address _owner) public view returns (uint256 balance)`
- `function transfer(address _to, uint256 _value) public returns (bool success)`
- `function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)`
- `function approve(address _spender, uint256 _value) public returns (bool success)`
- `function allowance(address _owner, address _spender) public view returns (uint256 remaining)`

B'.10.2)ERC-721

Τα smart contracts που υιοθετούν τον πρότυπο ERC-721 ονομάζονται NFT(Non-fungible-token) .Είναι ένας όρος που χρησιμοποιείται για την περιγραφή μοναδικών ψηφιακών περιουσιακών στοιχείων , των οποίων η ιδιοκτησία καταγράφεται στο Ethereum blockchain και μπορεί να πουληθεί. Κάθε nft περιέχει δεδομένα τα οποία το διαφοροποιούν από οποιοδήποτε άλλο. Τα δεδομένα αυτά είναι αδύνατο να αντιγραφούν . Το κύριο χαρακτηριστικό τους και η κύρια διαφορά τους από τα κρυπτονομίσματα και τα άλλα tokens όπως περιγράφεται από το όνομά τους, είναι ότι δεν είναι ανταλλάξιμα. Μιλώντας με οικονομικούς όρους , ένα αγαθό ορίζεται ως ανταλλάξιμο όταν μπορεί να επιτευχθεί η συναλλαγή του με ένα άλλο αγαθό της ίδιας αξίας, ενώ ένα αγαθό ορίζεται ως μη ανταλλάξιμο όταν δεν μπορεί να πραγματοποιηθεί η συναλλαγή του με κάτι που να έχει ακριβώς την ίδια αξία. Τα nft's ανήκουν στην δεύτερη κατηγορία καθώς κάθε ένα είναι ξεχωριστό και με αυτή την ιδιαιτερότητα μπορούν να προσδιορίσουν ένα αγαθό με μοναδικό τρόπο. Ο κύριος στόχος τους είναι η απόδειξη της ύπαρξης και της ιδιοκτησίας ψηφιακών αντικειμένων όπως ένα έργο τέχνης , ένα τραγούδι , ένα βίντεο κ.α. Η καταγραφή τους σε ένα blockchain ισχυροποιεί την ασφάλειά τους καθώς κανένας δεν μπορεί να τροποποιήσει , να αντιγράψει και να καταστρέψει τις καταγραφές. Κάθε nft μπορεί να ανήκει σε ένα και μόνο άτομο . Το γεγονός αυτό αποτελεί κίνητρο για ένα πιθανό αγοραστή να αγοράσει ένα nft λόγω της σπανιότητάς του η οποία καθορίζεται από τον δημιουργό του. Μπορεί να δημιουργήσει πολλά αντίγραφα του πρωτοτύπου ή και καθόλου , δίνοντάς του περισσότερη αξία. Η απόκτηση ενός nft δεν ταυτίζεται με την απόκτηση πνευματικών δικαιωμάτων του ψηφιακού στοιχείου που το αναπαριστά. Κάποιος που έχει αναπαραστήσει το έργο του μέσω ενός nft και το πουλήσει , είναι στη θέση του πωλητή αν θα του παραχωρήσει τα δικαιώματα πνευματικής ιδιοκτησίας ή να τα κρατήσει και να δημιουργήσει και άλλα αντίγραφα από το αρχικό έργο. Ο αγοραστής δεν μπορεί να κατέχει τα πνευματικά δικαιώματα παρά μόνο αν του παραχωρηθούν ρητά. Οι συναρτήσεις που χρησιμοποιεί ένα token ERC-721 είναι οι εξής [32] :

- `function balanceOf (address _owner) external view returns (uint256)`
- `function ownerOf(uint256 _tokenId) external view returns (address)`
- `function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external payable`
- `function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable`
- `function transferFrom(address _from, address _to, uint256 _tokenId) external payable`
- `function approve(address _approved, uint256 _tokenId) external payable`
- `function setApprovalForAll(address _operator, bool _approved) external`
- `function getApproved(uint256 _tokenId) external view returns (address)`

- `function isApprovedForAll(address _owner, address _operator) external view returns (bool)`

B'.10.2.1) Εφαρμογές

- **Ψηφιακή τέχνη:** Τα nft's μπορούν να χρησιμοποιηθούν για να δημιουργήσουν μοναδικά στο είδος τους ψηφιακά έργα τέχνης. Ένας καλλιτέχνης μπορεί να δημιουργήσει ένα nft το οποίο θα αντιπροσωπεύει ένα έργο τέχνης. Αυτό θα δώσει στον καταναλωτή επαρκή απόδειξη ότι είναι μοναδικό στο είδος του καθώς οποιοδήποτε ψηφιακό αρχείο μπορεί να αντιγραφεί από οποιονδήποτε. Αυτό συμβαίνει χάρη στο χαρακτηριστικό της τεχνολογίας blockchain , η οποία διασφαλίζει την αμεταβλητότητα των καταχωρήσεων. Επιπρόσθετα , δίνουν την δυνατότητα στον καλλιτέχνη , να εισπράττουν κέρδη από τις μεταπωλήσεις των έργων τους για πάντα μέσω του προγραμματισμού του nft.
- **Gaming:** Η τεχνολογία των nft μπορεί να χρησιμοποιηθεί με τέτοιο τρόπο ώστε ο παίκτης να έχει την δυνατότητα να αποκτήσει την ιδιοκτησία αντικειμένων εντός του παιχνιδιού. Αυτό μπορεί να θεωρηθεί ως επένδυση για το μέλλον καθώς η αξία του αντικειμένου μπορεί να ανέβει. Επίσης η δημιουργία μιας αγοράς θα ωφελήσει και τους δημιουργούς του παιχνιδιού αλλά και τους παίκτες. Μερικά από τα crypto-παιχνίδια που βρίσκονται στην κυκλοφορία είναι τα Cryptokitties, Cryptocats, Cryptopunks , Gods Unchanged, TradeStars κ.α.
- **Κτηματική περιουσία:** Η ιδιοκτησία ακινήτων και κτημάτων πλέον μπορεί να αναπαρασταθεί σαν νφτ και να καταγραφεί σε ένα blockchain και το ψηφιακό token θα περιέχει όλα τα απαραίτητα χαρακτηριστικά πχ τοποθεσία, τιμή. Χάρη στην ασφάλεια της τεχνολογίας blockchain κακόβουλοι παράγοντες είναι αδύνατο να παραβιάσουν ή να τροποποιήσουν την καταχώρηση. Σημαντικό πλεονέκτημα αποτελεί και η μείωση της γραφειοκρατίας αφού πλέον δεν είναι απαραίτητη η παρέμβαση μεσαζόντων όπως τράπεζες , κτηματομεσίτες , δικηγόροι.
- **Εκδηλώσεις:** Η πώληση των εισιτηρίων βασίζεται κυρίως σε κεντρικές εταιρίες οι οποίες παρέχουν εμπιστοσύνη. Οποιοσδήποτε μπορεί να μετατρέψει ένα εισιτήριο για έναν αγώνα ή μία συναυλία σε νφτ . Κάθε ένα εισιτήριο θα είναι μοναδικό, σπάνιο και θα περιέχει διαφορετικά χαρακτηριστικά. Αυτό σημαίνει ότι ο ιδιοκτήτης ενός εισιτηρίου δεν θα έχει την δυνατότητα να δημιουργήσει αντίγραφα από το πρωτότυπο από τη στιγμή που θα το πουλήσει. Το γεγονός αυτό θα συμβάλλει στην εξάλειψη της κυκλοφορίας των πλαστών εισιτηρίων.

- **Μουσική βιομηχανία:** Οι μουσικοί καλλιτέχνες και παραγωγοί έχουν πλέον την δυνατότητα να αναπαριστούν το μουσικό τους έργο σαν νφτ. Ένα τεράστιο πλεονέκτημα αυτού είναι το άμεσο χρηματικό όφελος των καλλιτεχνών χωρίς την παρουσία μιας ενδιάμεσης πλατφόρμας η οποία κερδοσκοπεί και δεν ανταμείβει τους καλλιτέχνες με βάση την πραγματική τους αξία [33] .

B'.10.2.2) Ευπάθειες

Ένα από τα σημαντικότερα προβλήματα αυτής της τεχνολογίας είναι το γεγονός ότι τα ίδια τα ψηφιακά αρχεία που αντιπροσωπεύονται από τα nft's δεν αποθηκεύονται αυτούσια σε ένα blockchain λόγω του μεγέθους τους. Λόγω αυτού οι συναλλαγές αναφέρονται σε αυτά μέσα από URL's όπου είναι αποθηκευμένα σε κεντρικούς servers. Η πρακτική μπορεί να αποδειχθεί αρκετά επώδυνη και καταστροφική στην περίπτωση που κάποιος κακόβουλος πραγματοποιήσει μία επίθεση στους servers καθώς όλα τα περιεχόμενα των nft's μπορεί να κλαπούν. Πολλά nft project αποθηκεύουν τα δεδομένα τους σε κατακευματισμένα συστήματα όπως το IPFS (InterPlanetary File System) στα οποία ένας χρήστης μπορεί να αναζητήσει πληροφορίες αρκεί κάποιος κόμβος του δικτύου να τις «φιλοξενεί» (hosting). Ακόμα και σε τέτοια συστήματα όταν ένας χρήστης αποθηκεύσει ένα περιεχόμενο του nft δεν είναι σίγουρο ότι αυτή η πληροφορία θα αντιγραφεί από όλους τους κόμβους. Το περιεχόμενο μπορεί να γίνει μη προσβάσιμο εάν έχει αποθηκευτεί σε ένα μόνο κόμβο και αυτός αποσυνδεθεί από το δίκτυο. Η γενική ιδέα των συστημάτων nft να αποθηκεύουν τα περιεχόμενά τους σε εξωτερικά συστήματα μπορεί να προκαλέσει προβλήματα. Πιθανή επίθεση που μπορεί να διεξαχθεί λόγω αυτού είναι η άρνηση υπηρεσιών DoS(Denial of Service) , όπου απειλείται η διαθεσιμότητα των ψηφιακών αρχείων από μία πιθανή επίθεση στους κεντρικούς εξυπηρετητές που περιέχονται τα αρχεία. Η παραβίαση και η τροποποίηση των αρχείων που βρίσκονται εκτός του blockchain είναι επίσης εφικτή. Με κάτι τέτοιο χάνεται η ακεραιότητα των δεδομένων. Τέλος με την επίθεση πλαστογράφησης (spoofing) , ένας κακόβουλος μπορεί να υποδυθεί μία οντότητα εκμεταλεύοντας μία ευπάθεια αυθεντικοποίησης ή να καταφέρει να κλέψει το ιδιωτικό κλειδί ενός χρήστη αποκτώντας πρόσβαση σε όλα του τα περιουσιακά στοιχεία που συνδέονται με αυτό. Στην περίπτωση που κάποιος αποκτήσει το ιδιωτικό κλειδί κάποιου άλλου και πραγματοποιήσει μία συναλλαγή , αυτή η συναλλαγή θα είναι μη αναστρέψιμη λόγω της τεχνολογίας του blockchain

[34]

Κεφάλαιο Γ΄

Γ΄) Βιβλιογραφική Ανασκόπηση

Σε αυτό το κεφάλαιο περιγράφονται συνοπτικά διαφορετικές προσεγγίσεις ανάπτυξης ενός συστήματος αποκεντρωμένης ηλεκτρονικής ψηφοφορίας με βάση την παγκόσμια βιβλιογραφία.

Γ΄.1) Απαιτήσεις Συστήματος

Η δημιουργία ενός ασφαλούς συστήματος ηλεκτρονικής ψηφοφορίας με την τεχνολογία blockchain , που θα το διαφοροποιούν από τις προ υπάρχουσες κεντριοποιημένες υλοποιήσεις ,πρέπει να πληροί ορισμένες προϋποθέσεις. Η ανωνυμία είναι βασικό και απαραίτητο χαρακτηριστικό , καθώς είναι αναγκαίο να μην υπάρχει σύνδεση μεταξύ ενός ψηφοδελτίου με την οντότητα που καταθέτει την ψήφο του. Οι χρήστες που θα χρησιμοποιήσουν το σύστημα επιβάλλεται να είναι εγγεγραμμένοι και το σύστημα να επαληθεύει την εγγραφή τους. Η ακεραιότητα κάθε ψήφου πρέπει να πιστοποιείται. Επίσης κάθε ψηφοφόρος πρέπει να μην έχει την δυνατότητα να ψηφίσει παραπάνω από μία φορά. Η καταμέτρηση των ψήφων πρέπει να επιτευχθεί με ορθό τρόπο έτσι ώστε να καταμετρηθούν όλες οι ψήφοι , να αποφευχθεί η αντιγραφή των ψήφων έτσι ώστε να εκπληρωθεί η ακρίβεια του συστήματος. Μία επιβεβαίωση ότι η ψήφος του εκάστοτε χρήστη έχει καταχωρηθεί και καταμετρηθεί προσφέρει διαφάνεια . Όλοι οι συμμετέχοντες πρέπει να έχουν πρόσβαση στα τελικά αποτελέσματα ώστε να διασφαλίζεται η επαληθευσσιμότητα αλλά να απαγορεύεται η πρόσβαση σε πιθανά αποτελέσματα κατά την διάρκεια της καταμέτρησης ώστε να μην επηρεάζονται οι ψηφοφόροι στην πρόθεση ψήφου τους.

Γ΄.2) Έμπιστη Οντότητα

Η διαδικασία εγγραφής των χρηστών και επαλήθευσης της οντότητας των χρηστών δυσχεραίνεται λόγω του ότι δεν έχουν όλοι οι πολίτες μιας χώρας πρόσβαση στο διαδίκτυο,

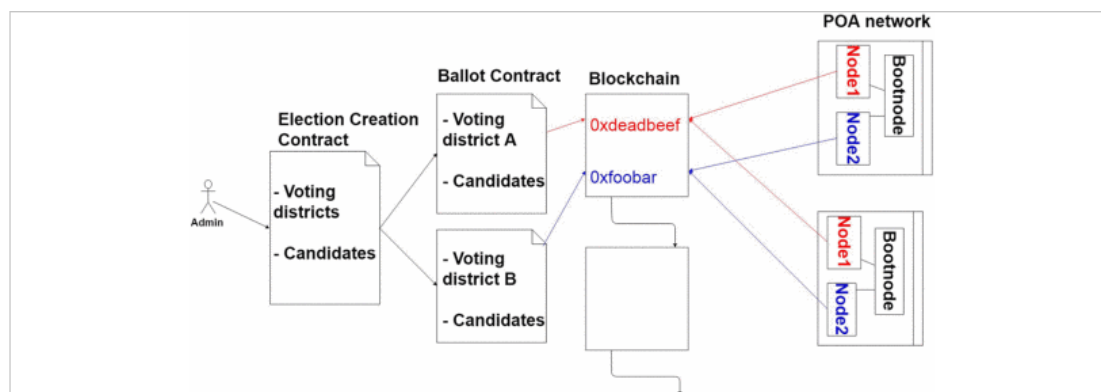
Στις περισσότερες έως τώρα υλοποιήσεις εμπλέκεται μία Τρίτη έμπιστη οντότητα η οποία παίρνει τον ρόλο του διαχειριστή της διαδικασίας και καλείται να δημιουργήσει μία λίστα από επιβεβαιωμένους ψηφοφόρους χρησιμοποιώντας μία κυβερνητική υπηρεσία επαλήθευσης οντότητας , γεγονός που μπορεί να δημιουργήσει προβλήματα ασφαλείας. Παρόλα αυτά ένας βαθμός χρήσης ενός κεντρικού συστήματος είναι απαραίτητος στην συγκεκριμένη εφαρμογή καθώς μόνο έτσι είναι εφικτή η

απόκρυψη των χρηστών και η εγγραφή επαληθευμένων ψηφοφόρων. Επιπρόσθετα όλοι οι κανόνες της διαδικασίας, όπως η διάρκεια της ψηφοφορίας, αποφασίζονται και δημοσιοποιούνται από αυτήν.

Γ'.3) Blockchain-Based E-Voting System

Στη συγκεκριμένη υλοποίηση προκειμένου να διασφαλιστεί το κατάλληλο επίπεδο ασφάλειας γίνεται η χρήση του Go-Ethereum ως ιδιωτικού ελεγχόμενου blockchain Proof-of-Authority (POA). Υπάρχουν 2 ειδών κόμβοι. Η πρώτη κατηγορία αναπαριστούν εκλογικές περιφέρειες και περιέχουν πράκτορες λογισμικού που αλληλεπιδρούν με την δεύτερη κατηγορία κόμβων και καθορίζουν τον κύκλο ζωής των smart contracts στον κόμβο αυτό. Η δεύτερη κατηγορία κόμβων ονομάζονται bootnodes των οποίων η κύρια λειτουργία τους είναι η διευκόλυνση της εξερεύνησης των περιφερειακών κόμβων μεταξύ τους και δεν διατηρούν καμία κατάσταση του blockchain.

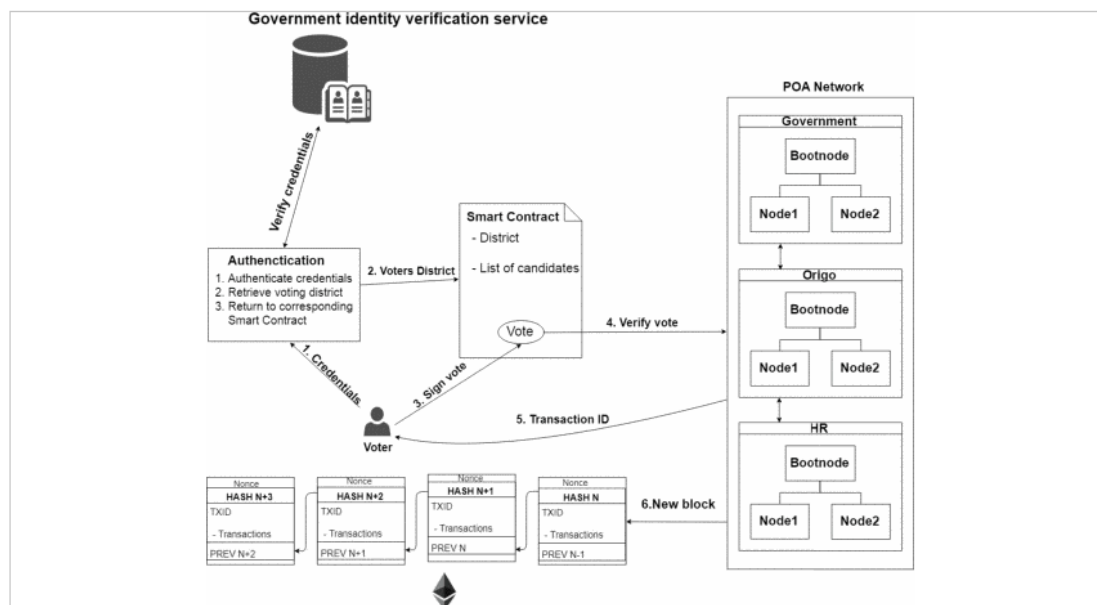
Κάθε εκλογική διαδικασία επιτυγχάνεται με τα smart contracts όπου κάθε ένα αντιστοιχεί σε μία εκλογική περιφέρεια και λειτουργεί σαν ξεχωριστό ψηφοδέλτιο. Ο διαχειριστής το συστήματος ψηφοφορίας είναι μία Τρίτη έμπιστη οντότητα όπως ένας οργανισμός ή μία εταιρία. Αυτός είναι υπεύθυνος για την δημιουργία όλων των εκλογικών ψηφοδελτίων και την εγγραφή των χρηστών με την δημιουργία μοναδικών πορτοφολιών που λειτουργεί ως ταυτότητα.



Εικόνα 19 : Δομή εκλογικού συστήματος [35]

Η καταμέτρηση των ψήφων πραγματοποιείται αυτόματα από τα smart contracts. Με την ολοκλήρωση κάθε ψήφου επιστρέφεται στους ψηφοφόρους ένα μοναδικό ID που προσδιορίζει μοναδικά την συναλλαγή που δημιουργήθηκε. Χρησιμοποιώντας αυτό το ID σε ένα blockchain explorer, ο καθένας μπορεί να εντοπίσει την συναλλαγή δήλωσης ψήφου του. Με αυτόν τον τρόπο εξασφαλίζεται η διαφάνεια του συστήματος.

Η κατάθεση ψήφου υλοποιείται με την αλληλεπίδραση του χρήστη με το smart contract της εκλογικής του περιφέρειας, το οποίο αλληλεπιδράει με το blockchain χάρη στον αντίστοιχο περιφερειακό κόμβο. Τέλος η πλειοψηφία των κόμβων επικυρώνουν την συναλλαγή ψήφου και καταγράφεται στο blockchain.



Εικόνα 20 : Διαδικασία ψηφοφορίας [36]

Τα πλεονεκτήματα αυτής της εφαρμογής είναι ότι χρησιμοποιεί ένα ιδιωτικό blockchain επομένως αποφεύγονται όλα τα κόστη των gas fees που επιβαρύνουν τους χρήστες που αλληλεπιδρούν με δημόσια blockchain. Απόρροια αυτού είναι και η αποφυγή της υψηλής κινητικότητας που επικρατεί στα δημόσια blockchain. Η δημιουργία περιφερειακών κόμβων και η παροχή ατομικών πορτοφολιών στους χρήστες πλεονεκτεί σε σχέση με τις υλοποιήσεις που οι ψηφοφόροι καλούνται να χρησιμοποιήσουν έναν browser για να ψηφίσουν καθώς είναι πολύ εύκολο για έναν επιτιθέμενο να εκμεταλλευτεί και να παραπλανήσει τους χρήστες.

Γ'.4) An Evoting Protocol with Decentralization and Privacy

Σε αυτό το σύστημα κάθε ψηφοφόρος είναι και ένας κόμβος του δικτύου [36]. Ένας χρήστης αναγνωρίζεται από το δημόσιο κλειδί του, το οποίο το αποκτά αφού κριθεί ικανός να ψηφίσει. Κάθε χρήστης καλείται να επιλέξει έναν από πολλούς προεπιλεγμένους υποψήφιους και δημοσιοποιεί μία ψηφιακή δέσμευση (digital commitment) αντί για την ψήφο του, ενισχύοντας τη δικαιοσύνη στο σύστημα. Για να κριθεί ένας ψηφοφόρος ικανός να ψηφίσει, πρέπει να ταυτοποιήσει τα στοιχεία του σε μία έμπιστη οντότητα και έπειτα να λάβει από αυτήν ένα token, το οποίο έχει την μορφή ψηφιακής υπογραφής στο δημόσιο κλειδί του χρήστη και της ψηφιακής δέσμευσης ψήφου. Κάθε ψήφος καταγράφεται στο blockchain και διαθέτει ένα

μοναδικό ID (V ID) . Σε κάθε ψηφοδέλτιο περιέχεται το δημόσιο κλειδί του χρήστη για να σηματοδοτήσει τον κάτοχο της ψήφου. Οι επόμενες 2 εικόνες αναπαριστούν το token και το ψηφοδέλτιο.

$$et_i = sig(V_{ipub}|dc_i)CA$$

Εικόνα 21 token επιβεβαίωσης

$$b_{ialt} = V_{ipub}|dc_i|et_i$$

Εικόνα 22 Ψηφοδέλτιο

Υπάρχει η δυνατότητα ακύρωσης της ψήφου με την χρήση ενός εναλλακτικού ψηφοδελτίου. Το οποίο αποτελείται από την ακύρωση της ψήφου με το μοναδικό V ID που την προσδιορίζει , το ιδιωτικό κλειδί που χρησιμοποιείται σαν κλειδί επικύρωσης υπογραφής.

$$b_i = c(VID_x)|V_{ipriv}|dc_n|sig(c(VID_x)|V_{ipriv}|dc_n)V_i$$

Εικόνα 23 Ψηφοδέλτιο αλλαγής ψήφου

Η μόνη σύνδεση της ταυτότητας του ψηφοφόρου και της ψήφου είναι το δημόσιο κλειδί του. Ο τρόπος με τον οποίο αποτρέπεται αυτή η σύνδεση είναι με την χρήση blind signatures , οι οποίες παρέχουν την δυνατότητα στην Τρίτη οντότητα να παράγουν μία υπογραφή στο digital commitment της ψήφου και στο δημόσιο κλειδί των χρηστών χωρίς να έχει την δυνατότητα να προσδιορίσει κανένα από τα δύο. Ο κάθε χρήστης στην συγκεκριμένη περίπτωση λειτουργεί σαν κόμβος του δικτύου και εφαρμόζει μία blind συνάρτηση στο δημόσιο κλειδί του και το digital commitment.

$$, blind (V_{ipub}|dc_i) :$$

Εικόνα 24 Εφαρμογή Blind συνάρτησης

Αφού ο ψηφοφόρος κάνει χρήση της blind συνάρτησης στο μήνυμα , το στέλνει στον διαχειριστή ο οποίος θα το υπογράψει και θα το στείλει πίσω. Ο χρήστης θα λάβει το μήνυμα θα βγάλει την υπογραφή και θα καταλήξει με ένα token το οποίο θα χρησιμοποιήσει για να αποδείξει το δικαίωμά του να ψηφίσει.

$$sig^{-1}(sig(V_{ipub}|dc_i)CA) = sig(V_{ipub}|dc_i)CA$$

Εικόνα 25 Αφαίρεση της υπογραφής της Έμπιστης οντότητας από τον χρήστη

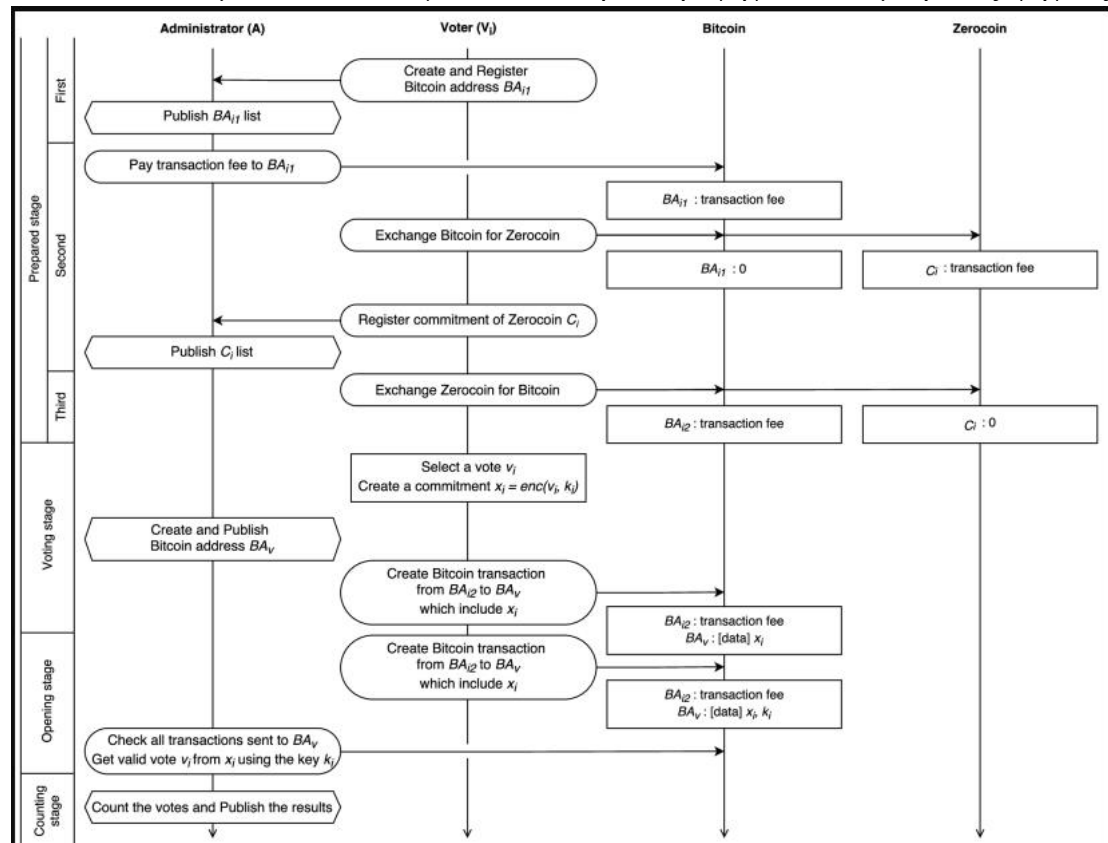
Στην φάση της ψηφοφορίας κάθε ψηφοφόρος διαδίδει στο δίκτυο την ψήφο του και στην καταμέτρηση των ψήφων καλούνται να αποκαλύψουν την επιλογή τους το μήνυμα της τρίτης εικόνας .

Με την χρήση των blind συναρτήσεων επιτυγχάνεται η ανωνυμία της διεύθυνσης των ψηφοφόρων. Επίσης διασφαλίζεται η δικαιοσύνη καθώς τα αποτελέσματα δεν είναι δυνατόν να δημοσιοποιηθούν πριν την ολοκλήρωση της καταμέτρησης μέσω του ψηφιακού commitment και διαχωρίζοντας την φάση της ψηφοφορίας με της καταμέτρησης. Από τη στιγμή που το blockchain είναι δημόσιο κάθε ψηφοφόρος μπορεί να πιστοποιήσει την εγκυρότητα της καταμέτρησης των ψήφων.

Γ'.5) Zerocoin

Σε αυτή την υλοποίηση [37] γίνεται η χρήση του Zerocoin , ενός εναλλακτικού κρυπτονομίσματος παρόμοιο με το Bitcoin για να προστεθεί η ανωνυμία στο σύστημα. Ένας χρήστης μετατρέπει Bitcoin σε Zerocoin , χρησιμοποιεί μία μέθοδο του Zerocoin που ονομάζεται Mint με την οποία μετατρέπει το Zerocoin σε Zerocoin commitment , το μετατρέπει σε καινούργιο Zerocoin με την μέθοδο Spend και ξαναγίνεται η μετατροπή του σε Bitcoin. Τα δεδομένα αποθηκεύονται εξολοκήρου στο Bitcoin blockchain και κάθε ψηφοφόρος πρέπει να διαθέτει μία διεύθυνση Bitcoin . Η έμπιστη οντότητα παραθέτει το ανάλογο ποσό σε bitcoin στις διευθύνσεις για να καλυφθούν τα έξοδα της ψηφοφορίας. Στη συνέχεια κάθε χρήστης μετατρέπει το ποσό που του αναλογεί σε Zerocoin Commitment C_i και ο διαχειριστής καταγράφει τα commitments από όλους τους χρήστες σε μία λίστα. Στην επόμενη φάση γίνεται η μετατροπή αυτών σε Bitcoin , και αποκτά μία καινούργια διεύθυνση Bitcoin αφού . Ο κάθε χρήστης επιλέγει έναν υποψήφιο και δημιουργεί ένα commitment $\xi = \text{enc}(v_i, k_i)$, όπου το k_i είναι ένα τυχαίο κλειδί. Ο ψηφοφόρος δημιουργεί μία συναλλαγή από την καινούργια του διεύθυνση προς την διεύθυνση της οντότητας και περιέχει στην παράμετρο OP Return το commitment. Δημιουργείται μία δεύτερη συναλλαγή , με την οποία ο χρήστης στέλνει το κλειδί με την παράμετρο OP return για να γίνει εφικτό το άνοιγμα του commitment. Στο τελικό στάδιο η έμπιστη οντότητα ελέγχει όλες τις συναλλαγές για την εγκυρότητα των Zerocoin commitments όταν αντάλλαξαν Zerocoin με Bitcoin και ξεκλειδώνει το

commitment x_i με το κλειδί k_i για να ανακτήσει την ψήφο v_i και μετρά τις ψήφους.



Εικόνα 26: Προτεινόμενο Σύστημα

πηγή : <https://ieeexplore.ieee.org/abstract/document/9333937>

Σε ένα τέτοιο σύστημα επιτυγχάνεται η ανωνυμία αφού η καινούργια διεύθυνση Bitcoin των χρηστών δεν μπορεί να συνδεθεί με την πρωταρχική καταχώρηση στην έμπιστη οντότητα. Επίσης αφού οι ψήφοι κρυπτογραφούνται δεν μπορούν να επηρεάσουν την διαδικασία της ψηφοφορίας κατά την διάρκειά της.

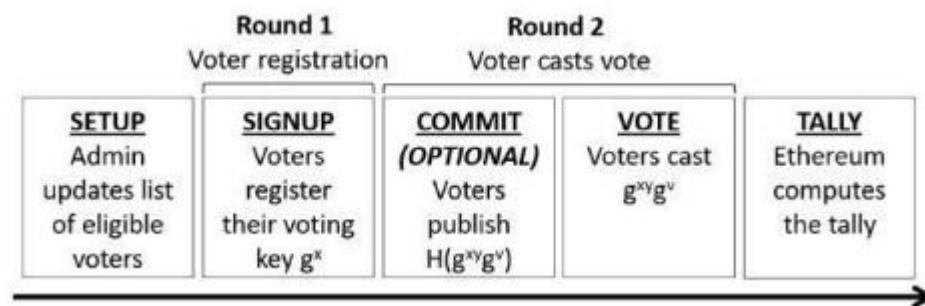
Γ'.6) ERC-20 Voting System

Στη συγκεκριμένη υλοποίηση [38] γίνεται η χρήση ενός smart contract που έχει εφαρμόσει το ERC-20 πρότυπο στον κώδικά του. Το smart contract που ελέγχει την ψηφοφορίας δημοσιοποιείται στο δίκτυο παίρνοντας ως όρισμα στον constructor την διεύθυνση ενός οποιουδήποτε ERC-20 smart contract. Ο διαχειριστής εκχωρεί tokens στο smart contract τα οποία θα χρησιμοποιηθούν ως φιλοδώρημα στους ψηφοφόρους και καθορίζει τον αριθμό των tokens που θα αμείβονται οι συμμετέχοντες. Κάθε χρήστης που διαθέτει ένα token έχει το δικαίωμα να ψηφίσει. Σε αυτό το σύστημα ψηφοφορίας οι συμμετέχοντες αποτελούν πρωταρχικό ρόλο στην λήψη αποφάσεων των οργανισμών/εταιριών που θα το υλοποιήσουν, δίνεται η δυνατότητα στην εταιρία να συλλέξει χρήσιμα δεδομένα που θα μπορούσαν να

χρησιμοποιηθούν για μελλοντικές αποφάσεις. Τα αποτελέσματα έδειξαν ότι πάνω από το 50% των ατόμων που διέθεταν tokens συμμετείχαν στην εκλογική διαδικασία.

Γ'.7) Open Vote Network (New Castle University)

Αναπτύχθηκε ένα αρκετά πρωτότυπο [49] συγκριτικά με τις όλες τις υπόλοιπες υλοποιήσεις εκλογικό σύστημα στο οποίο δεν γίνεται η χρήση έμπιστων οντοτήτων κατά την διάρκεια της εκλογικής διαδικασίας. Οι ψήφοι καταγράφονται στο κατακευματισμένο P2P δίκτυο σε πολλαπλούς γύρους και όλοι οι ψηφοφόροι πιστοποιούν την τελευταία ψήφο χωρίς να έχουν πρόσβαση στα δεδομένα άλλων ψηφοφόρων. Η κλιμάκωση του συστήματος είναι περιορισμένη λόγω των πολλών αλληλεπιδράσεων. Οι ιδιότητες του συστήματος είναι οι εξής : Πλήρης αποκέντρωση του συστήματος με υλοποίηση ψηφοφορίας σε 2 στάδια , η καταμέτρηση των ψήφων μπορεί να υλοποιηθεί από όλους τους ψηφοφόρους , χρήση του proof-of-concept , δημιουργία 2 smart contracts εκ των οποίων το ένα παίζει τον ρόλο του smart contract της ψήφου και το άλλο το κρυπτογραφικό και 5 φάσεις ψηφοφορίας setup , commit , sign-up , vote , tally.

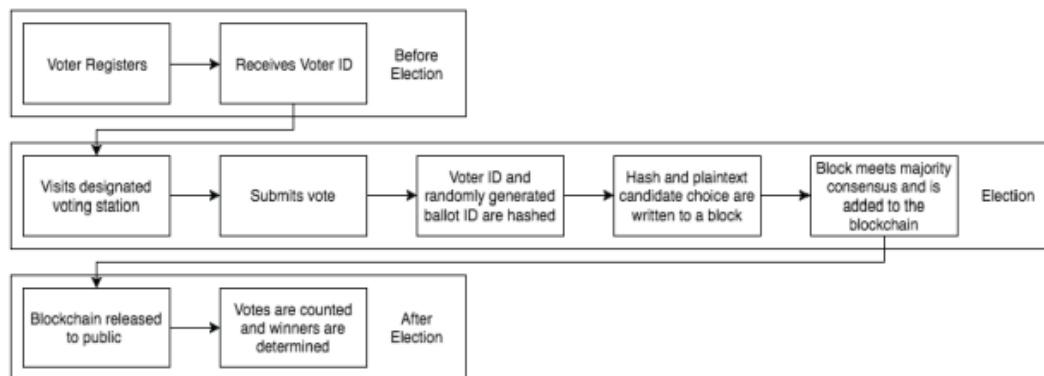


Εικόνα 27: Γύροι Ψηφοφορίας [49]

Στην πρώτη φάση ο διαχειριστής του συστήματος έχει την ευθύνη της πιστοποίησης των ψηφοφόρων κατασκευάζοντας μία λίστα δικαιούχων , υπολογίζει , ρυθμίζει και αποφασίζει τον χρόνο των υπόλοιπων φάσεων , δημιουργεί την ερώτηση ψηφοφορίας και υπολογίζει τα έξοδα εγγραφής. Οι χρήστες αφότου αποκτήσουν πρόσβαση στην ερώτηση καλούνται να καταθέσουν την προεπιλεγμένη ποσότητα eth για να εγγραφούν. Το τρίτο στάδιο είναι προαιρετικό καθώς οι χρήστες έχουν την δυνατότητα να στείλουν το Hash value των δεδομένων της προηγούμενης φάσης έτσι ώστε να διασφαλιστεί η πρόθεσή τους. Στο προτελευταίο στάδιο οι ψηφοφόροι ψηφίζουν. Όταν ο διαχειριστής λάβει την τελευταία ψήφο ειδοποιεί το blockchain για να ξεκινήσει η καταμέτρηση , της οποίας τα αποτελέσματα δημοσιεύονται στο blockchain.

Γ'.8) Votebook

Στην υλοποίησή τους υπάρχει μία Τρίτη έμπιστη οντότητα , στην οποία έχει ανατεθεί ο ρόλος της κατανομής των κλειδιών σε όλους τους κόμβους του δικτύου. Λόγω αυτού γίνεται η χρήση ενός ιδιωτικού “permissioned” blockchain. Κάθε κόμβος αποτελεί ένα σύστημα ψηφοφορίας το οποίο δημιουργεί ένα ζεύγος δημοσίου-ιδιωτικού κλειδιού από τα οποία το ιδιωτικό παραμένει κρυφό και το δημόσιο στέλνεται στην έμπιστη οντότητα. Κάθε μπλοκ αποτελείται από ένα ID που προσδιορίζει τον κόμβο μοναδικά , μία χρονοσφραγίδα η υπολογίζεται από ένα time-based πρωτόκολλο , το σύνολο των ψηφοφόρων , το Hash value του προηγούμενου μπλοκ και μία ψηφιακή υπογραφή. Στο συγκεκριμένο σύστημα δίνεται η δυνατότητα του ελέγχου της καταμέτρησης της ψήφου του κάθε ψηφοφόρου από τον ίδιο , τα αποτελέσματα της καταμέτρησης παραμένουν κρυφά πριν γίνει η τελική δημοσιοποίησή τους , δεν υπάρχει η δυνατότητα του coerce , οι ψήφοι διαμαρτυρίας δεν υπολογίζονται στην τελική καταμέτρηση.

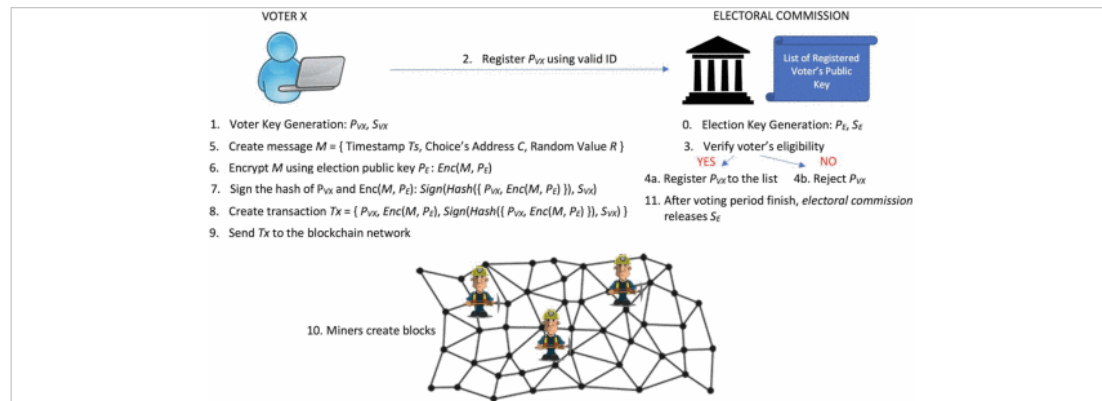


Εικόνα 28 : Ψηφοφορία [40]

Γ.9) Blockchain voting system

Η συγκεκριμένη υλοποίηση [41] χρησιμοποιεί την πρακτική του double envelop encryption σε συνδυασμό με ένα blockchain. Στην αρχή γίνεται η δημιουργία ζεύγους δημοσίου-ιδιωτικού κλειδιού από τους χρήστες και από τον έμπιστο διαχειριστή του συστήματος. Οι χρήστες καλούνται να πραγματοποιήσουν την εγγραφή του δημοσίου κλειδιού τους για να αποκτήσουν το δικαίωμα ψήφου χρησιμοποιώντας ένα προκαθορισμένο ID και να μην το γνωστοποιήσουν σε κανένα άτομο. Ο διαχειριστής ελέγχει το ID και πράττει αναλόγως. Προκειμένου ο χρήστης να ψηφίσει πρέπει να δημιουργήσει ένα μήνυμα ($M = fTs;C;Rg;$) το οποίο αποτελείται από μία χρονοσφραγίδα Ts , την διεύθυνση που αντιστοιχεί στον υποψήφιο που θέλουν να ψηφίσουν και μία τυχαία μεταβλητή R . Ο ρόλος της χρονοσφραγίδας δίνει την δυνατότητα στους ψηφοφόρους να ψηφίσουν περισσότερο από μία φορά αλλά μόνο η τελευταία χρονικά ψήφος θα καταμετρηθεί. Η τιμή R αποτρέπει έναν πιθανό επιτιθέμενο να ανακαλύψει το ζεύγος κλειδιών από τα κρυπτογραφημένα μηνύματα. Στη συνέχεια ο χρήστης πραγματοποιεί την κρυπτογράφηση του μηνύματος M με το

δημόσιο κλειδί του διαχειριστή $Enc(M;PE)$ και στη συνέχεια υπογράφει το Hash από το δημόσιο κλειδί σε συνδυασμό με το κρυπτογραφημένο μήνυμα ($Sign(Hash(PV \ X; \ Enc(M;PE)g);SV \ x)$). Η συναλλαγή που θα δημιουργήσει ο ψηφοφόρος θα είναι της μορφής :

$$Tx = PV \ Env(M;PE);signg;$$


Εικόνα 29: Ηλεκτρονικό σύστημα ψηφοφορίας [41]

Η συναλλαγή περιέχει το δημόσιο κλειδί του ψηφοφόρου $P_V \ X$, το κρυπτογραφημένο μήνυμα $Enc(M; PE)$ και την υπογραφή του Hash. Τέλος ο ψηφοφόρος στέλνει την συναλλαγή στο δίκτυο.

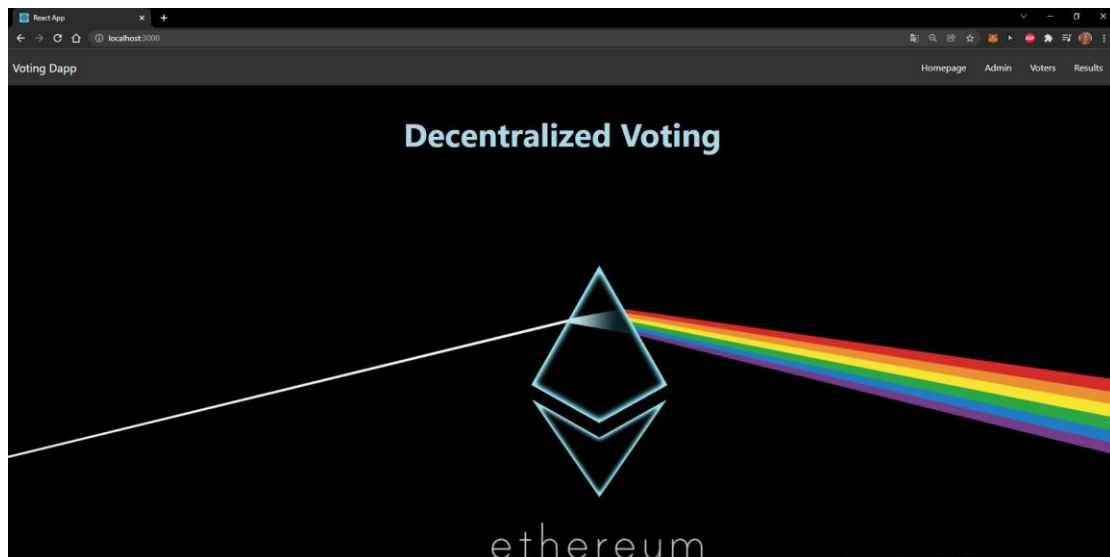
Το συγκεκριμένο σύστημα απαιτεί από τους ψηφοφόρους την δημιουργία ζεύγους κλειδιών, την κρυπτογράφηση, την δημιουργία του hash και την υπογραφή του αλλά και την αποστολή της συναλλαγής στο δίκτυο. Επομένως δεν είναι καθόλου εύρηστο. Η εκλογική έμπιστη οντότητα επίσης έχει την ικανότητα να συνδέσει τις ψήφους με τους ψηφοφόρους καθώς οι ίδιοι καταθέτουν τα δημόσια κλειδιά τους με το ID προκειμένου να εγγραφούν. Η πιστοποίηση της καταμέτρησης είναι εφικτή αφού στο τέλος της, η έμπιστη οντότητα μπορεί να γνωστοποιήσει στο ιδιωτικό της κλειδί και να δώσει πρόσβαση σε όλους στο τελικό αποτέλεσμα.

Κεφάλαιο Δ΄

Δ΄) Παρουσίαση Εφαρμογής

Σε αυτό το κεφάλαιο παρουσιάζεται η πρακτική χρήση της εφαρμογής βήμα προς βήμα με αναλυτικό τρόπο καθώς και τυχόν σφάλματα που μπορούν να προκύψουν από την λανθασμένη εκτέλεσή της.

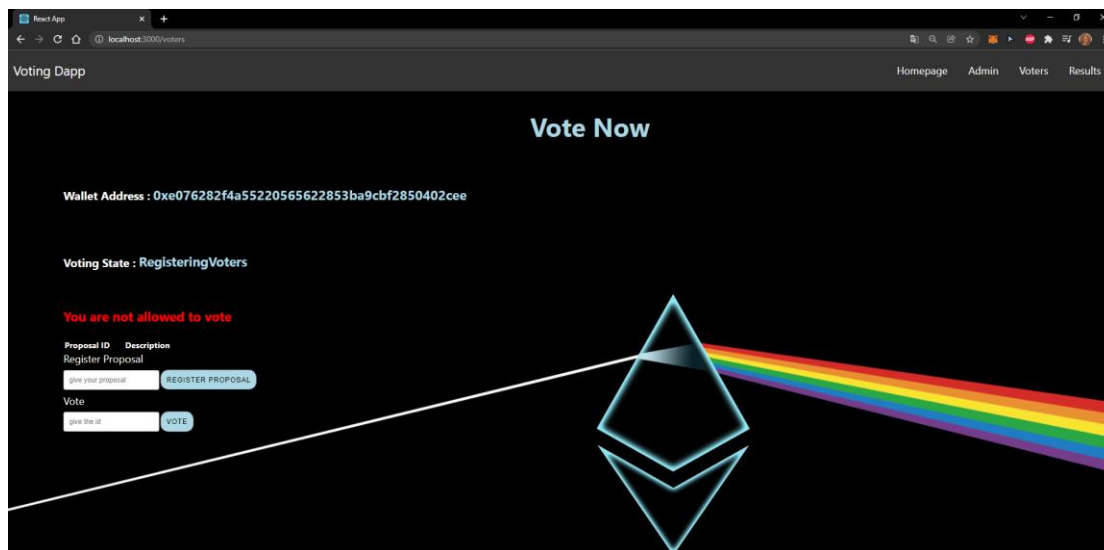
Δ΄.1) Αρχική σελίδα



Εικόνα 30 : Αρχική σελίδα

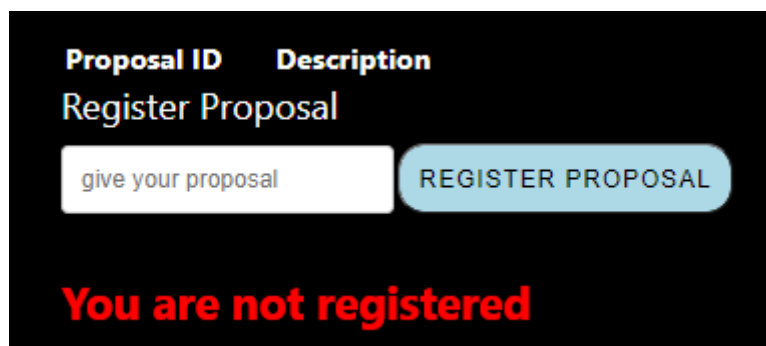
Στην πάνω εικόνα παρουσιάζεται η αρχική σελίδα της εφαρμογής. Όπως και σε όλες τις σελίδες, εμφανίζεται το πανθαρ, από το οποίο ο χρήστης μπορεί να κατευθυνθεί στα αποτελέσματα, στην σελίδα που χρησιμοποιείται για να ψηφίσει και να καταθέσει υποψήφιους, αλλά και στην σελίδα του administrator στην οποία όμως δεν μπορεί να εκτελέσει καμία ενέργεια. Κεντρικά της ιστοσελίδας αναγράφεται το όνομα της εφαρμογής και πίσω απεικονίζεται μία εικόνα.

Δ'.2) Κατάθεση ψήφων και υποψηφίων



Εικόνα 31 : Σελίδα ψηφοφόρου

Αυτή η σελίδα χρησιμοποιείται από τους χρήστες για να ψηφίσουν και να καταθέσουν υποψηφίους. Όταν ο χρήστης εισέλθει σε αυτή τη σελίδα, αυτόματα θα του ζητηθεί να συνδεθεί στο metamask wallet του, με το οποίο θα πραγματοποιεί οποιαδήποτε είδους συναλλαγή ορίζει το σύστημα. Αφού πραγματοποιηθεί αυτή η ενέργεια, εμφανίζεται αριστερά της οθόνης η διεύθυνση του πορτοφολιού του, το στάδιο της τρέχουσας ψηφοφορίας, ένα μήνυμα για το αν του επιτρέπεται σε αυτή τη φάση να ψηφίσει ή όχι, τα ονόματα των υποψηφίων που έχουν κατατεθεί μαζί με τα ID τους με την σειρά κατάθεσης, 2 input πεδία στα οποία ο χρήστης μπορεί να εισάγει το όνομα του υποψηφίου και κάτω το ID του υποψηφίου που επιθυμεί να ψηφίσει, και 2 κουμπιά για να εκτελέσει τις παραπάνω ενέργειες. Στην περίπτωση που η διεύθυνση του ψηφοφόρου δεν έχει καταγραφεί στο blockchain από τον διαχειριστή του συστήματος και αυτός πατήσει ένα από τα 2 κουμπιά, τότε εμφανίζεται το παρακάτω μήνυμα λάθους με κόκκινα γράμματα.

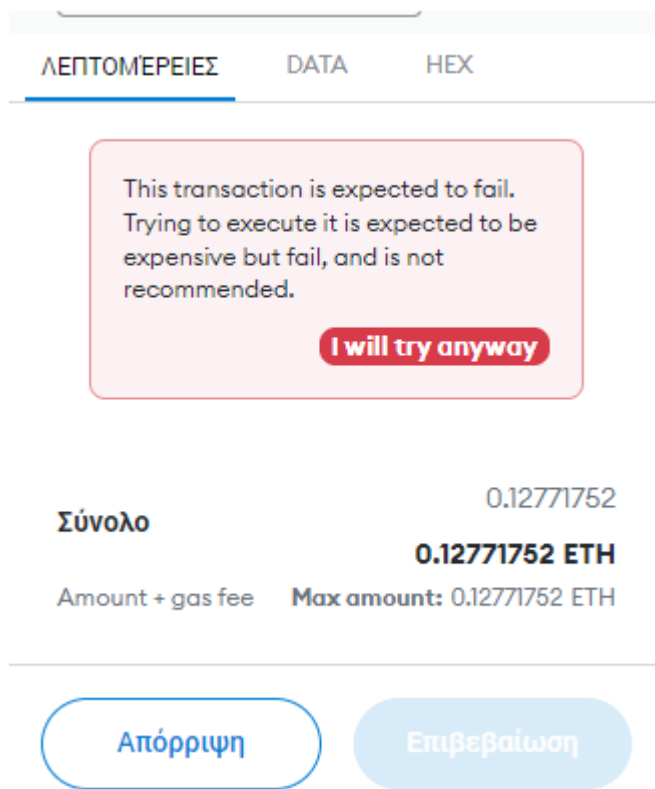


Εικόνα 32: Μήνυμα μη καταχωρημένης εγγραφής



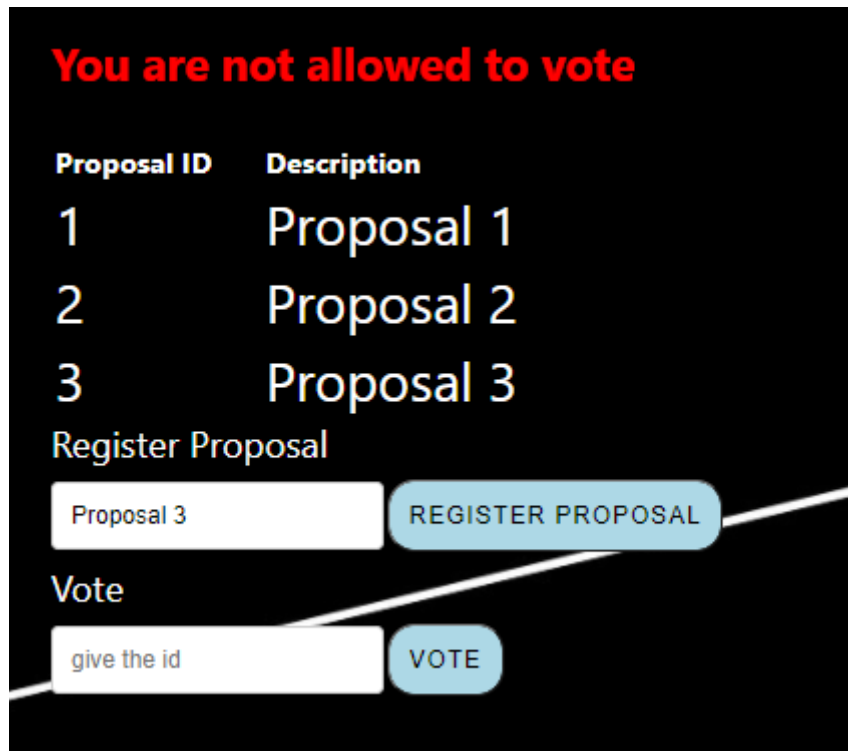
Εικόνα 33: Μήνυμα μη καταχωρημένης εγγραφής

Ένας ψηφοφόρος δεν επιτρέπεται από τον κώδικα του smart contract να προτάσσει κάποιον υποψήφιο που έχει προστεθεί προηγουμένως , να πραγματοποιήσει 2 καταχωρήσεις , να ψηφίσει πάνω από μία φορά , να προσπαθήσει να ψηφίσει ή να καταχωρήσει κάποιον υποψήφιο σε διαφορετικές φάσεις από αυτές που τις αντιστοιχούν και να εκτελέσει ενέργειες που αρμόζουν στον διαχειριστή. Στην προσπάθεια των 2 προηγουμένων , θα εμφανιστεί η αποτυχία της συναλλαγής στην παρακάτω εικόνα.



Εικόνα 34: Αποτυχημένη Συναλλαγή

Σε κάθε καταχώρηση υποψηφίων εμφανίζονται αυτόματα με την σειρά καταγραφής τους στην σελίδα. Κάθε λογαριασμός του local blockchain Ganache έχει στο υπόλοιπό του 100 eth τα οποία θα τα χρησιμοποιήσει για να πραγματοποιήσει τις συναλλαγές.



You are not allowed to vote

Proposal ID	Description
1	Proposal 1
2	Proposal 2
3	Proposal 3

Register Proposal

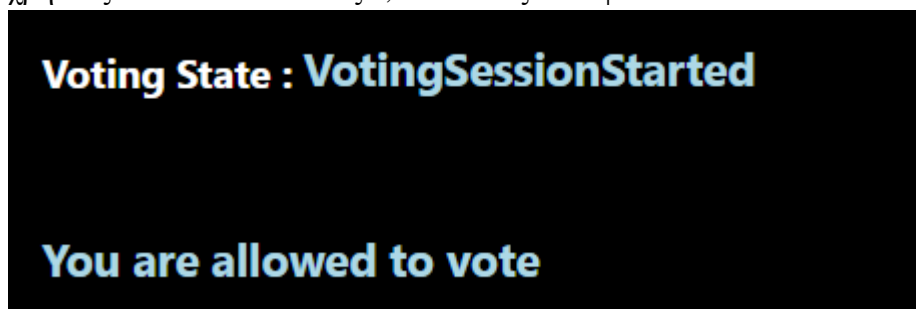
Proposal 3

Vote

give the id

Εικόνα 35: Καταχώρηση 3 υποψηφίων

Στην φάση της ψηφοφορίας το κόκκινο μήνυμα απαγόρευσης που εμφανίζεται στους χρήστες θα αλλάξει, όπως φαίνεται και στην εικόνα.



Voting State : VotingSessionStarted

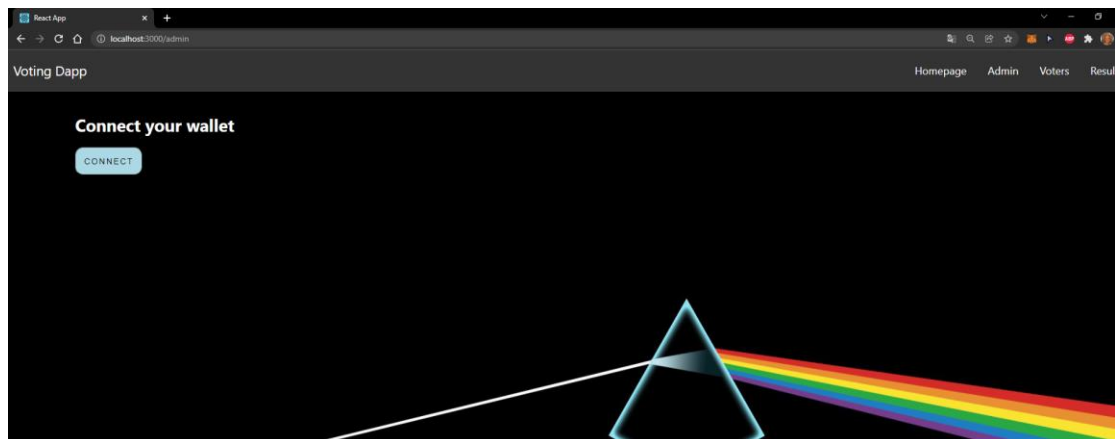
You are allowed to vote

Εικόνα 36: Φάση της ψηφοφορίας

Στη συνέχεια κάθε ψηφοφόρος καλείται να καταθέσει στο κατάλληλο input πεδίο το ID του ονόματος που επιθυμεί να ψηφίσει όπως αναγράφονται πάνω.

Δ'.3) Σελίδα διαχειριστή

Στην αριστερή πλευρά της οθόνης υπάρχει ένα κουμπί που με το πάτημά του θα ζητηθεί από τον διαχειριστή να εξουσιοδοτήσει την σύνδεση με την ιστοσελίδα.



Εικόνα 37: Σελίδα διαχειριστή πριν την σύνδεση

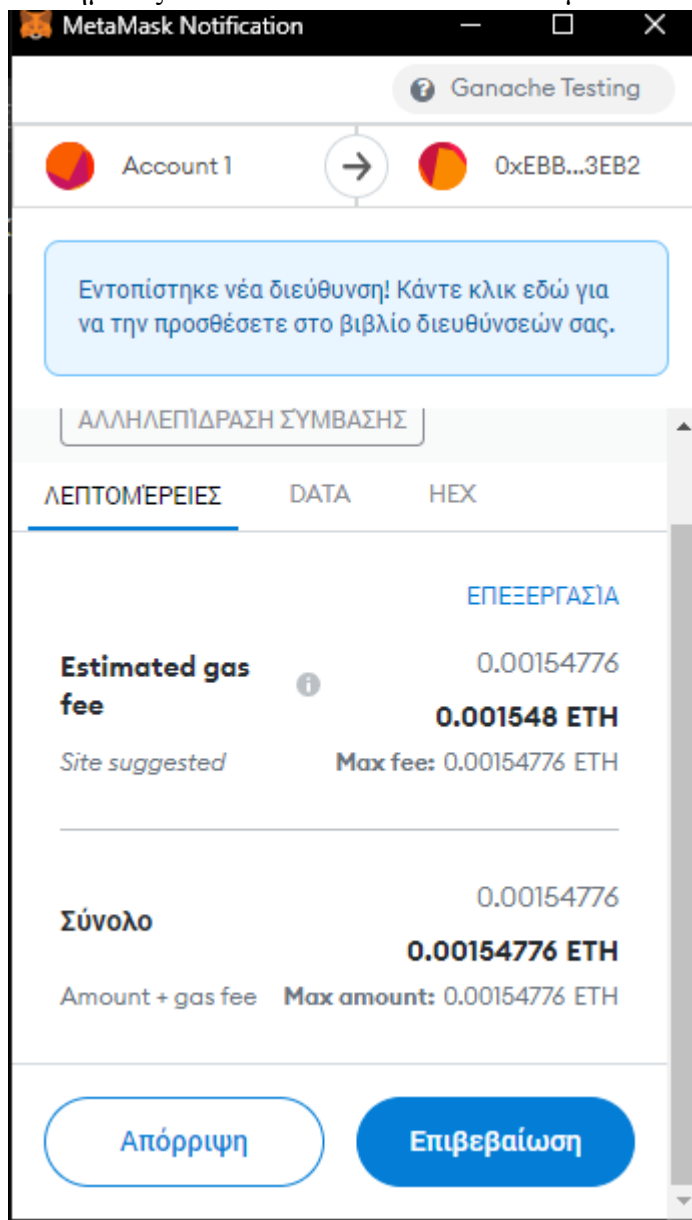


Εικόνα 38 : Σελίδα διαχειριστή μετά την σύνδεση

Στο επόμενο στάδιο αριστερά της σελίδας εμφανίζεται η διεύθυνση του πορτοφολιού, η διεύθυνση του smart contract, και η διεύθυνση του χρήστη που έχει εισέλθει στην σελίδα.

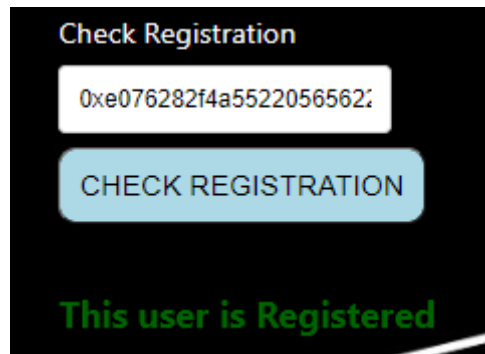
Έπειτα υπάρχουν 2 input πεδία στα οποία ο διαχειριστής πραγματοποιεί την εγγραφή των ψηφοφόρων και κάνει τον έλεγχο της εγγραφής των χρηστών. Σε αυτή την εικόνα απεικονίζεται η συναλλαγή που πρόκειται να πραγματοποιηθεί όταν ο διαχειριστής κάνει την εγγραφή κάποιου χρήστη. Όλες οι ενέργειες του

συστήματος υλοποιούνται με τον ίδιο τρόπο.

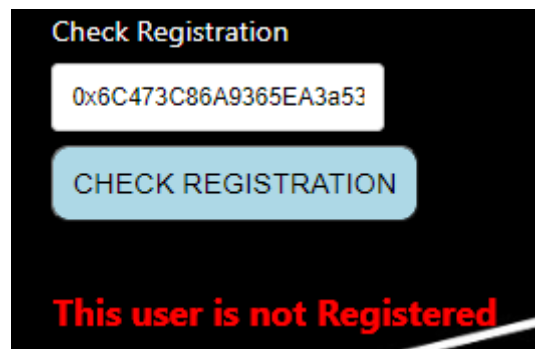


Εικόνα 39 : Επιβεβαίωση και πραγματοποίηση συναλλαγής

Οι παρακάτω 2 εικόνες δείχνουν μήνυμα ελέγχου 2 χρηστών , εκ των οποίων ο ένας έχει εγγραφεί και ο άλλος όχι.



Εικόνα 40: Μήνυμα ελέγχου εγγεγραμμένου ψηφοφόρου

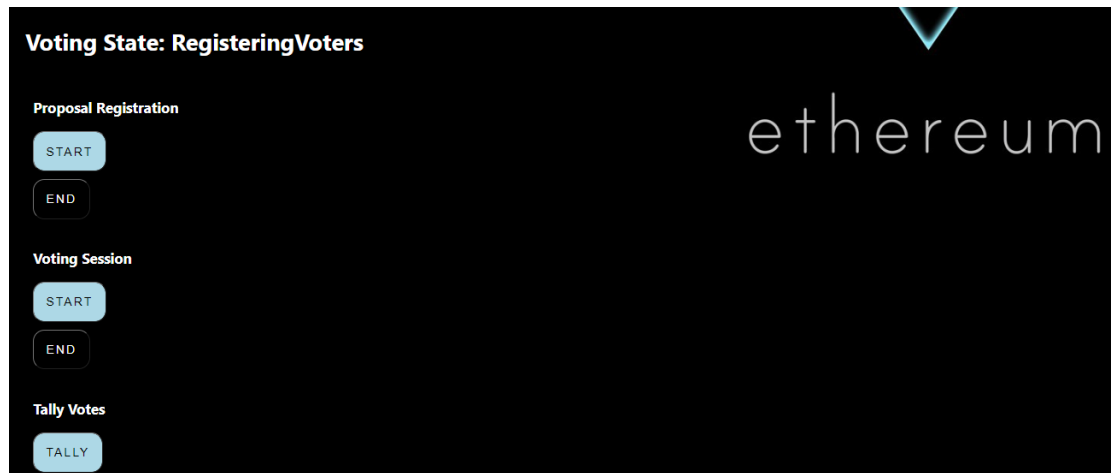


Εικόνα 41: Μήνυμα ελέγχου με εγγεγραμμένου ψηφοφόρου

Ακριβώς κάτω από την φάση της ψηφοφορίας υπάρχουν 5 κουμπιά. Κάθε κουμπί αντιστοιχεί στην αλλαγή της φάσης της ψηφοφορίας. Όταν μία φάση αλλάξει δεν είναι δυνατόν να γίνει αλλαγή σε μία προγενέστερη. Υπάρχουν 6 φάσεις.

- **RegisteringVoters**
- **ProposalRegistrationStarted**
- **ProposalRegistrationEnded**
- **VotingSessionStarted**
- **VotingSessionEnded**
- **TallyVotes**

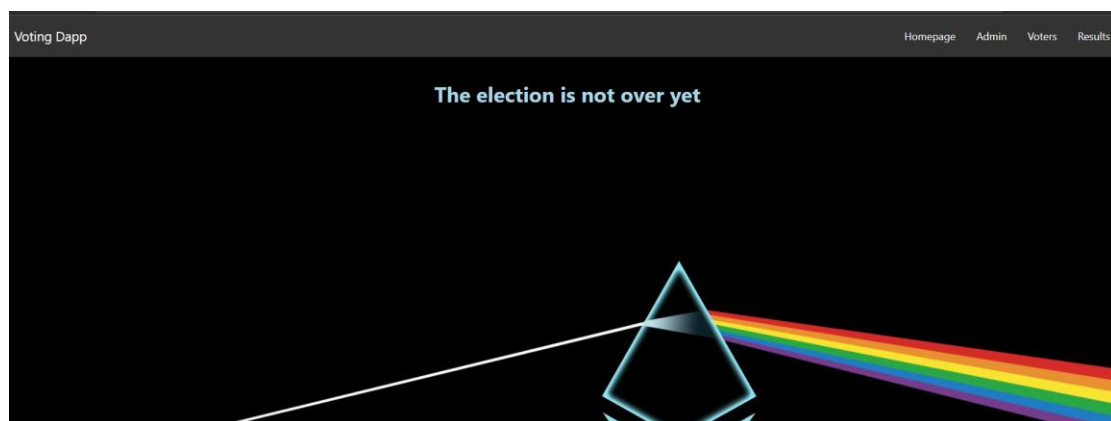
Στην πρώτη φάση γίνεται η εγγραφή των ψηφοφόρων. Στην δεύτερη φάση γίνεται η κατάθεση των υποψηφίων από τους εγγεγραμμένους χρήστες. Η τρίτη και η Πέμπτη φάση σηματοδοτούν το τέλος των ακριβώς προηγούμενων. Στην τέταρτη φάση οι χρήστες έχουν το δικαίωμα και την δυνατότητα να ψηφίσουν τους προτασόμενους υποψήφιους. Τέλος στην έκτη φάση υπολογίζονται τα αποτελέσματα της ψηφοφορίας. Η διαδικασία εναλλαγής φάσης, όπως και όλες οι ενέργειες, επιτυγχάνεται με την μορφή συναλλαγής και καταγράφεται στο blockchain.



Εικόνα 42: Κουμπιά διαχειριστή

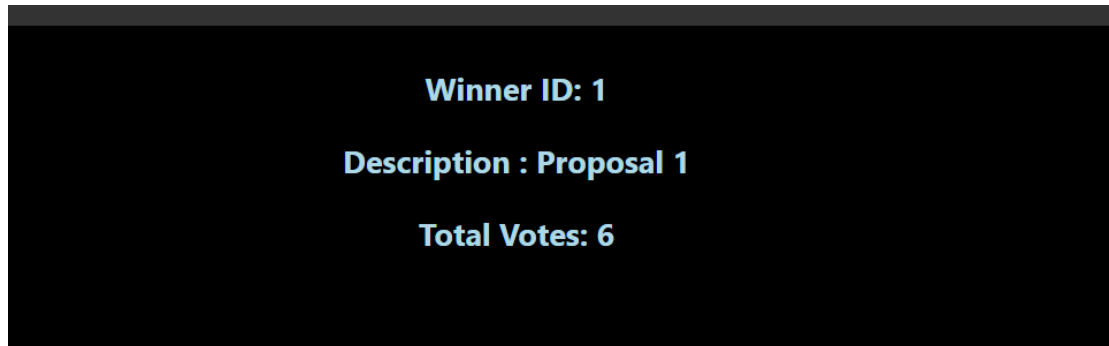
Δ'.4) Αποτελέσματα

Στις 5 πρώτες φάσεις αυτή η σελίδα θα είναι διαθέσιμη και θα εμφανίζει στους χρήστες το παρακάτω μήνυμα.



Εικόνα 43: Σελίδα αποτελεσμάτων πριν την καταμέτρηση ψήφων

Όταν ολοκληρωθεί η διαδικασία της ψηφοφορίας και ο διαχειριστής πατήσει το κουμπί της τελευταίας φάσης (Tally) , στο κέντρο της σελίδας θα εμφανιστεί το όνομα του νικητή των εκλογών , το ID του και οι συνολικές ψήφων που δόθηκαν σε αυτόν.



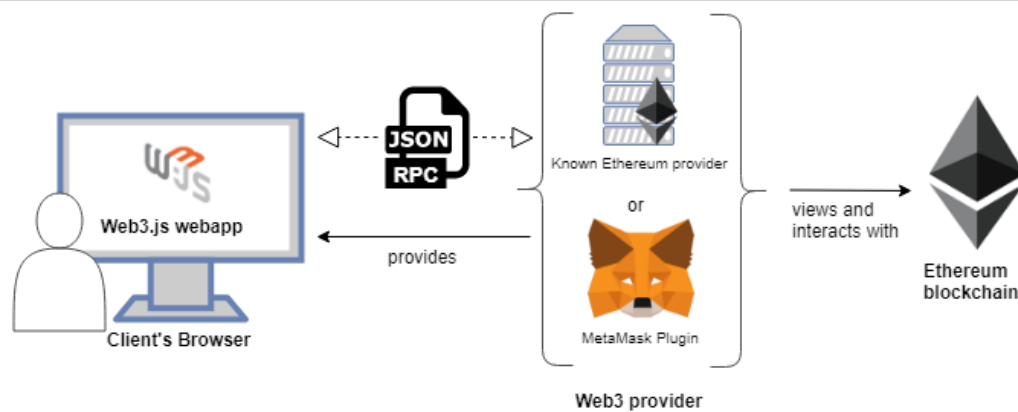
Εικόνα 44 : Σελίδα αποτελεσμάτων μετά την καταμέτρηση ψήφων

Κεφάλαιο Ε΄

Ε΄) Υλοποίηση Εφαρμογής

Σε αυτό το κεφάλαιο αναφέρονται όλες οι τεχνολογίες που χρησιμοποιήθηκαν προκειμένου να αναπτυχθεί η εφαρμογή.

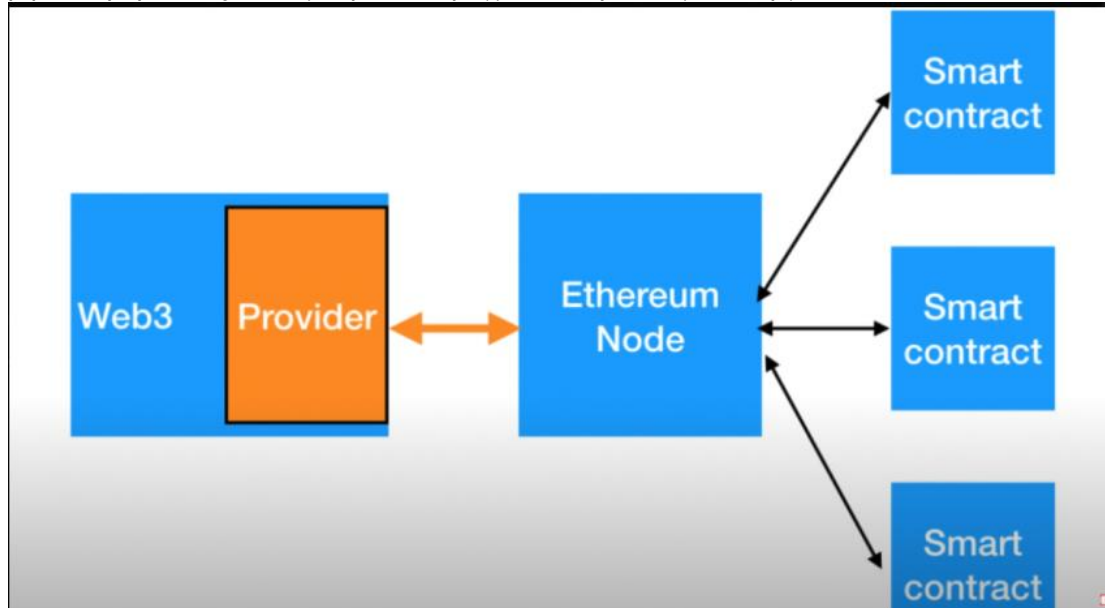
Ε΄.1) Σχεδιασμός



Εικόνα 45 Μηχανισμός Εφαρμογής

Το Metamask extension εκχωρεί στον browser ένα API με το οποίο ο χρήστης μπορεί να αλληλεπιδράσει με το blockchain πραγματοποιώντας ενέργειες γραφής ή ανάγνωσης δεδομένων. Αυτό επιτυγχάνεται με την χρήση της JavaScript βιβλιοθήκης web3 και του provider που εκχωρεί το Metamask. Η ιστοσελίδα επικοινωνεί με το ganache , ο οποίος πέρα από blockchain λειτουργεί και σαν node του δικτύου, με την χρήση του web3. Ένας provider λαμβάνει JSON-rpc requests και επιστρέφει ένα response.Χωρίς την χρήση ενός provider η

βιβλιοθήκη web3.js δεν μπορεί να πραγματοποιήσει καμία ενέργεια σε κανένα blockchain.



Ε'.2) Smart Contract

Για την συγγραφή του κώδικα του smart contract χρησιμοποιήθηκε η έκδοση 0.8.10 της γλώσσας προγραμματισμού solidity. Η συγκεκριμένη υλοποίηση μπορεί να εκτελεστεί σε οποιοδήποτε blockchain δίκτυο που χρησιμοποιεί το Ethereum Virtual Machine.

Ακολουθεί η ανάλυση των βασικότερων κομματιών του κώδικα.

Ε'.2.1) Απαραίτητες Μεταβλητές


```

contract VotingSystem{

    enum State {RegisteringVoters,
    ProposalsRegistrationStarted,
    ProposalsRegistrationEnded,
    VotingSessionStarted,
    VotingSessionEnded,VotesTallied}

    struct Candidate{
        string description;
        uint votes;
    }
    struct Voter{
        bool is_registered;
        bool did_voted;
        bool did_propose;
        uint candidateId;
    }

    address public administrator;
    State public voting_state;
    mapping(address=>Voter) public all_voters;
    Candidate[] public all_candidates;
    uint private candidateWinnerId;
}

```

Εικόνα 46: Δήλωση μεταβλητών smart contract

Στην αρχή γίνεται η δήλωση του ονόματος του smart contract ως “VotingSystem”. Έπειτα ακολουθεί η δήλωση και αρχικοποίηση μιας enum μεταβλητής η οποία περιέχει 6 διαφορετικές προδιαγεγραμμένες τιμές που αναπαριστούν τις φάσεις της ψηφοφορίας. Στη συνέχεια ορίζονται 2 structs , ένα για τους υποψήφιους και ένα για τους ψηφοφόρους. Το struct του υποψηφίου αποτελείται από μία μεταβλητή τύπου string η οποία αποθηκεύει την περιγραφή του και από μία uint μεταβλητή , η οποία καταγράφει τον συνολικό αριθμό των ψήφων προς αυτόν. Το struct του ψηφοφόρου περιέχει 3 μεταβλητές bool, οι οποίες παρέχουν πληροφορίες σχετικά με το αν έχει επιτευχθεί η εγγραφή του , αν έχει ψηφίσει και αν έχει κάνει πρόταση για υποψήφιο. Η τελευταία μεταβλητή είναι τύπου uint και υποδεικνύει το ID του υποψηφίου που ψήφισε. Ακολουθούν μία address τύπου μεταβλητή η οποία αποθηκεύει τον υποκινητή της ψηφοφορίας , μία State η οποία προέρχεται από το προαναφερθέν enum όπου θα συγκρατεί την φάση της ψηφοφορίας , ένα mapping το κλειδί του οποίου είναι η διεύθυνση του ψηφοφόρου και η τιμή του ένας ψηφοφόρος από το struct Voter. Τέλος δηλώνεται ένας πίνακας τύπου struct candidate στον οποίο θα αποθηκεύονται όλοι οι υποψήφιοι και μία uint μεταβλητή που συμβολίζει τον νικητή της ψηφοφορίας.

Ε'.2.2) Modifier

Σειρά έχουν τα modifiers , τα οποία χρησιμοποιούνται για την επιβολή «κανόνων» στις συναρτήσεις.

```
modifier onlyAdmin(){
    require(msg.sender==administrator,"Only the system admin can call this function");
    _;
}

modifier onlyRegistered(){
    require(all_voters[msg.sender].is_registered==true,"You are not registered");
    _;
}
```

Εικόνα 47 : Δήλωση των modifier του smart contract

Εδώ παρουσιάζονται 2 από τα συνολικά 11 modifiers , στα οποία δηλώνονται 2 κανόνες. Ο πρώτος είναι ότι αυτός που καλεί τις συναρτήσεις στις οποίες έχει επιβληθεί ο κανόνας πρέπει να είναι και ο διαχειριστής του smart contract. Ο δεύτερος απαγορεύει την εκτέλεση της συνάρτησης από κάποιον που δεν έχει εγγραφεί στην πρώτη φάση της ψηφοφορίας.

Ε'.2.3) Constructor

```
constructor(){
    administrator=msg.sender;
    voting_state=State.RegisteringVoters;
}
```

Εικόνα 48: Constructor

Ο constructor στην γλώσσα solidity καλείται όταν γίνεται deploy το smart contract. Στην συγκεκριμένη περίπτωση χρησιμοποιείται για να αρχικοποιήσει τον διαχειριστή του smart contract με την διεύθυνση του λογαριασμού που το κάνει deploy στο δίκτυο και την μεταβλητή voting_state με την πρώτη φάση της ψηφοφορίας.

Ε'.2.4) Εγγραφή ψηφοφόρων

```
function registerVoters(address voter_address)
external onlyAdmin duringVoterRegistration()
notRegistered(voter_address)
{
    all_voters[voter_address]=Voter(true,false,false,0);
}
```

Εικόνα 49: Συνάρτηση εγγραφής ψηφοφόρων

Με την χρήση αυτής της συνάρτησης επιτυγχάνεται η εγγραφή των ψηφοφόρων στο smart contract. Παίρνει ως όρισμα την διεύθυνση ενός ψηφοφόρου , δημιουργεί μια μεταβλητή τύπου struct Voter και αποθηκεύεται στο mapping all_voters όπου το key είναι η διεύθυνση και το value η local μεταβλητή Voter.

Ε΄.2.5) Πρόταση υποψηφίων

```
function setProposal(string memory _description)
external onlyRegistered() duringProposalRegistration()
newProposalOnly(_description) setProposalOnce()
{
    all_candidates.push(Candidate(_description,0));
    all_voters[msg.sender].did_propose=true;
}
```

Εικόνα 50: Συνάρτηση κατάθεσης υποψηφίων

Τα modifiers επιτρέπουν το κάλεσμα της συνάρτησης από κάποιον λογαριασμό που είναι εγγεγραμμένος , δεν έχει ξανακάνει πρόταση υποψηφίου , δεν υπάρχει καταχώρηση του ίδιου υποψηφίου από άλλον χρήστη και η φάση της ψηφοφορίας είναι κατά την διάρκεια της εγγραφής υποψηφίων. Η συνάρτηση παίρνει ως όρισμα ένα string το οποίο θα αποτελεί την περιγραφή ή το όνομα του προτασόμενου υποψηφίου. Δημιουργεί ένα local Candidate, το εισάγει στον πίνακα all_candidates όπου αποθηκεύονται όλοι οι υποψήφιοι και αλλάζει η τιμή της μεταβλητής που καθορίζει αν ο ψηφοφόρος που εκτέλεσε την συνάρτηση έκανε πρόταση ή όχι σε true.

Ε΄.2.6) Δήλωση Ψήφου

```

function vote(uint _candidateId)
external onlyRegistered duringVotingPhase
{
    require(all_voters[msg.sender].did_voted==false,"this address has already voted");
    all_voters[msg.sender].did_voted=true;
    all_voters[msg.sender].candidateId=_candidateId;
    all_candidates[_candidateId].votes+=1;
}

```

Εικόνα 51: Συνάρτηση κατάθεσης ψήφου

Εδώ τα modifiers περιορίζουν το κάλεσμα της συνάρτησης σε όσους έχουν εγγραφεί. Επίσης η συνάρτηση μπορεί να καλεστεί μόνο κατά την διάρκεια της φάσης της ψηφοφορίας (voting phase). Στην αρχή γίνεται ένας έλεγχος για το αν ο λογαριασμός που εκτελεί την συνάρτηση έχει ψηφίσει. Αν ναι τότε η συναλλαγή δεν μπορεί να ολοκληρωθεί. Στην περίπτωση που δεν έχει ψηφίσει, αλλάζουν οι μεταβλητές του συγκεκριμένου ψηφοφόρου. Η bool μεταβλητή did_voted παίρνει την τιμή true και η candidateId παίρνει το όρισμα που δέχεται η συνάρτηση. Τέλος γίνεται η καταμέτρηση της ψήφου, προσθέτοντας +1 στην μεταβλητή votes που περιέχει ο υποψήφιος με το id του ορίσματος της συνάρτησης.

Ε'.2.7) Καταμέτρηση ψήφων

```

function tallyVotes()
external onlyAdmin AfterVotingPhase()
{
    uint winningVoteCount = 0;
    uint winningProposalIndex = 0;
    for(uint i=0;i<all_candidates.length;i++){
        if(all_candidates[i].votes>winningVoteCount){
            winningVoteCount=all_candidates[i].votes;
            winningProposalIndex=i;
        }
    }
    candidateWinnerId=winningProposalIndex;
    voting_state=State.VotesTallied;
}

```

Εικόνα 52: Συνάρτηση καταμέτρησης ψήφων

Η συνάρτηση επιτρέπει το κάλεσμά της αποκλειστικά και μόνο από τον διαχειριστή του smart contract και μετά το τέλος της φάσης της ψηφοφορίας. Δηλώνονται 2 τοπικές αριθμητικές μεταβλητές uint. Η πρώτη συμβολίζει έναν μετρητή ψήφων και η δεύτερη το ID ενός υποψήφιου. Στη συνέχεια με ένα for loop υπολογίζεται ποιος υποψήφιος έχει συλλέξει τις περισσότερες ψήφους, η μεταβλητή που καθορίζει τον

νικητή candidateWinnerId παίρνει την τιμή που υπολογίζεται από το for loop και τέλος η φάση της ψηφοφορίας τελειώνει.

5.2.8) Αλλαγή φάσης ψηφοφορίας

```
function startProposalRegistration()
external onlyAdmin() duringVoterRegistration()
{
    voting_state=State.ProposalsRegistrationStarted;
}
```

Εικόνα 53: Συνάρτηση αρχής κατάθεσης υποψηφίων

```
function endProposalRegistration()external onlyAdmin() duringProposalRegistration(){
    voting_state=State.ProposalsRegistrationEnded;
}

function startVotingSession()external onlyAdmin() duringProposalRegistrationEnd(){
    voting_state=State.VotingSessionStarted;
}
```

Εικόνα 54: Συναρτήσεις τέλους κατάθεσης υποψηφίων και αρχής ψηφοφορίας

```
function endVotingSession()
external onlyAdmin() duringVotingPhase()
{
    voting_state=State.VotingSessionEnded;
}
```

Εικόνα 55: Συνάρτηση τέλους φάσης ψηφοφορίας

Οι συναρτήσεις αλλάζουν την φάση της ψηφοφορίας τροποποιώντας την global μεταβλητή voting_state.

Ε.2.9) Τελευταίες συναρτήσεις

Η χρήση αυτών συνεισφέρει στην παροχή πληροφοριών στους χρήστες της εφαρμογής, επιστρέφοντας πληροφορίες για τις global μεταβλητές του smart contract.

```

function getCandidatesNumbers()
external view returns(uint)
{
    return all_candidates.length;
}

function getCandidateDescription(uint candidateId)
external view returns(string memory)
{
    return all_candidates[candidateId].description;
}

function getCandidateWinnerId()
external view duringVotesTallied() returns(uint)
{
    return candidateWinnerId;
}

function getCandidateWinnerDescription()
external view duringVotesTallied() returns(string memory)
{
    return all_candidates[candidateWinnerId].description;
}

function getTotalVotes()
external view duringVotesTallied() returns(uint)
{
    return all_candidates[candidateWinnerId].votes;
}

function isRegistered(address _voter_address)
external view returns(bool)
{
    return all_voters[_voter_address].is_registered;
}

function getVotingStatus()
external view returns(State)
{
    return voting_state;
}

```

Εικόνα 56: Βοηθητικές συναρτήσεις

Ε.3) Smart Contract Deployment

Το deploy του smart contract στο τοπικό ganache blockchain πραγματοποιήθηκε με την χρήση του εργαλείου truffle.

```
module.exports = {
  contracts_build_directory: './voting-dapp/src/contracts_abi',
  networks: {
    development: {
      host: '127.0.0.1',      // Localhost (default: none)
      port: 7545,            // Standard Ethereum port (default: none)
      network_id: '*',      // Any network (default: none)
    },
  },
}
```

Εικόνα 57: Παραμετροποίηση αρχείου truffle-config.js

Στο αρχείο truffle-config.js το οποίο δημιουργήθηκε αυτόματα με την εντολή truffle-init , παραμετροποιούμε το network object σύμφωνα με τα άλλα εργαλεία που χρησιμοποιούμε. Στην συγκεκριμένη περίπτωση ορίζουμε ως host την IP του ganache και ως port το port στο οποίο λειτουργεί. Στην περίπτωση που θέλαμε το deploy να γίνει σε κάποιο δίκτυο testnet , στο αρχείο αυτό υπάρχει ο κώδικας που θα τροποποιήσουμε.

```
// ropsten: {
//   provider: () => new HDWalletProvider(mnemonic, `https://ropsten.infura.io/v3/YOUR-PROJECT-ID`),
//   network_id: 3,      // Ropsten's id
//   gas: 5500000,       // Ropsten has a lower block limit than mainnet
//   confirmations: 2,   // # of confs to wait between deployments. (default: 0)
//   timeoutBlocks: 200, // # of blocks before a deployment times out (minimum/default: 50)
//   skipDryRun: true    // Skip dry run before migrations? (default: false for public nets )
// },
```

Εικόνα 58 : Σύνδεση σε άλλα δίκτυα από το truffle-config.js

Πριν προχωρήσουμε στο deployment είναι απαραίτητη η μεταγλώττιση του smart contract χρησιμοποιώντας την εντολή truffle compile.

```
Compiling your contracts...
=====
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\VotingSystem.sol
> Compiling .\contracts\VotingSystem.sol
> Artifacts written to C:\Users\kakos\Desktop\votingSystem\voting-dapp\src\contracts_abi
> Compiled successfully using:
   - solc: 0.8.10+commit.fc410830.Emscripten.clang
```

Εικόνα 59: Μεταγλώττιση του smart contract

Τέλος με την εντολή truffle migrate , υλοποιείται το deployment στο blockchain που

έχουμε

συνδεθεί.

```

Starting migrations...
=====
> Network name:      'development'
> Network id:        5777
> Block gas limit: 6721975 (0x6691b7)

1_initial_migration.js
=====

  Replacing 'Migrations'
  -----
  > transaction hash: 0x5f893ea391702ad923327bf7c2efc32d1a46d989d60306a9f832ac5e02d24468
  > Blocks: 0        Seconds: 0
  > contract address: 0xf62b0CDD9c453D2deB0416FfC5f2A1086A4df5C9
  > block number:     1
  > block timestamp:  1643926295
  > account:          0xe3585c76fD54820a228fc7e556C36451c7085f8d
  > balance:          99.99502292
  > gas used:         248854 (0x3cc16)
  > gas price:        20 gwei
  > value sent:       0 ETH
  > total cost:       0.00497708 ETH

  > Saving migration to chain.
  > Saving artifacts
  -----
  > Total cost:       0.00497708 ETH

2_VotingSystem_migration.js
=====

  Replacing 'VotingSystem'
  -----
  > transaction hash: 0xbc8e2e9b66d9da6c75f9a990365c2321b7127bb164408d490d5db866d1af7604
  > Blocks: 0        Seconds: 0
  > contract address: 0xC2A427E0Ca6442bb9d91a90f6F1F79D2C3Bb7291
  > block number:     3
  > block timestamp:  1643926297
  > account:          0xe3585c76fD54820a228fc7e556C36451c7085f8d
  > balance:          99.95224198
  > gas used:         2096534 (0x1ffd96)
  > gas price:        20 gwei
  > value sent:       0 ETH
  > total cost:       0.04193068 ETH

  > Saving migration to chain.
  > Saving artifacts
  -----
  > Total cost:       0.04193068 ETH

Summary
=====
> Total deployments: 2
> Final cost:       0.04690776 ETH

```

Εικόνα 60: Deployment των smart contracts στο δίκτυο

Ε΄.4) Εργαλεία και Τεχνολογίες

Προκειμένου να ολοκληρωθεί η υλοποίηση της εφαρμογής χρησιμοποιήθηκαν τα ακόλουθα εργαλεία :

- Ganache
- Metamask
- Truffle Suite
- Visual Studio Code
- Node.js
- React.js
- Javascript
- Web3.js
- Solidity

E'.4.1) Ganache

Το Ganache [42] αποτελεί ένα εργαλείο χειρισμού ενός προσωπικού ιδιωτικού blockchain. Παρέχει στον χρήστη 10 διευθύνσεις δηλαδή 10 διαφορετικούς λογαριασμούς με τα ιδιωτικά τους κλειδιά. Κάθε λογαριασμός έχει σαν υπόλοιπο 100 εικονικά Ether. Χρησιμοποιείται για την ανάπτυξη εφαρμογών πριν γίνει η δημοσίευση του Dapp ή του smart contract σε κάποιο testnet ή στο κύριο δίκτυο του Ethereum το mainnet. Κάθε συναλλαγή που πραγματοποιείται καταγράφεται σε 1 block το οποίο δημιουργείται αυτόματα. Ο χρήστης έχει την δυνατότητα να δει το περιεχόμενο του κάθε block και όλων των συναλλαγών που πραγματοποιούνται. Οποιαδήποτε στιγμή είναι εφικτό να γίνει επανεκκίνηση του ganache και να δημιουργήσει 10 εντελώς καινούργιες διευθύνσεις με τα αρχικά ποσά τους. Πιο συγκεκριμένα πρόκειται για ένα Ethereum node το οποίο δέχεται json-RPC requests από την εφαρμογή. Πέρα από την κλασσική εφαρμογή ganache με την βοήθεια γραφικού περιβάλλοντος, υπάρχει και το ganache-cli το οποίο διαθέτει περισσότερες λειτουργίες και επιλογές για τον χρήστη. Στην παρούσα εφαρμογή έγινε η χρήση του ganache με γραφικό περιβάλλον.

E'.4.2) Metamask

Πρόκειται για ένα browser extension [43], για τους φυλλομετρητές Google Chrome, Opera, Brave και Mozilla Firefox και προσφέρει μία εύχρηστη κατανοητή και εύκολη διεπαφή μέσω της οποίας ένας χρήστης έχει την δυνατότητα να διαχειρίζεται λογαριασμούς χρηστών σε όλα τα Ethereum δίκτυα, είτε δημόσια είτε ιδιωτικά, διαχειρίζεται ιδιωτικά κλειδιά και hardware wallets, συνδέει τον χρήστη με το blockchain, να στείλει και να υπογράψει συναλλαγές/δεδομένα. Ουσιαστικά είναι ένας τρόπος σύνδεσης του browser με το blockchain. Οι προγραμματιστές το μόνο που έχουν να κάνουν είναι να αλληλοεπιδράσουν με το Ethereum API το οποίο είναι διαθέσιμο σε browser που έχουν εγκατεστημένο το extension. Αυτό το API επιτρέπει σε ιστοσελίδες να διαβάζουν και να τροποποιούν δεδομένα από το blockchain στο οποίο είναι συνδεδεμένες.

E'.4.3) Truffle Suite

Είναι ένα οικοσύστημα [44] που αποτελείται από 3 κομμάτια. Ένα από αυτά είναι το truffle.

Το truffle είναι ένα εργαλείο , ένα περιβάλλον ανάπτυξης το οποίο στοχεύει στην διευκόλυνση δημιουργίας των Dapps. Δίνει την δυνατότητα στον χρήστη μέσω του command line να κάνει deploy smart contracts στο Ethereum δίκτυο της επιλογής του , να αλληλοεπιδράσει με αυτά , αυτοματοποιεί την διαδικασία της δοκιμής των smart contracts και διευκολύνει τον χρήστη στην ανάπτυξή τους.

Ε'.4.4) Node.js

Το Node.js [45] είναι μία Server-side πλατφόρμα η οποία είναι κατασκευασμένη πάνω από το Google Chrome's Javascript Engine (V8 Engine) και υλοποιήθηκε από τον Ryan Dahl το 2009. Απώτερος σκοπός του είναι η ανάπτυξη γρήγορων και με επεκτατικές προοπτικές διαδικτυακών εφαρμογών με εύκολο τρόπο. Χρησιμοποιεί ένα event-driven non-blocking I/O που το καθιστά αποτελεσματικό χρησιμοποιώντας λίγους πόρους. Ο κώδικας προγραμματισμού που περιέχουν οι εφαρμογές που υλοποιούνται με την χρήση του node.js είναι η Javascript και είναι συμβατές με λειτουργικά συστήματα όπως Microsoft Windows, Linux, OS X. Προσφέρει μία ποικιλία βιβλιοθηκών από Javascript Modules τα οποία βελτιστοποιούν και απλοποιούν τις διαδικτυακές εφαρμογές εκμεταλλευοντάς τες. Συνεπώς το node.js αποτελεί περιβάλλον εκτέλεσης αλλά και μία Javascript βιβλιοθήκη.

Στην συγκεκριμένη εφαρμογή έγινε η χρήση η έκδοση 16.13.1 του node.js. Έχοντας εγκαταστήσει το node.js , έγινε η χρήση του npm (node package manager) μέσω του command line για την εγκατάσταση όλων των απαραίτητων εργαλείων προκειμένου να αναπτυχθεί η εφαρμογή. Μερικά από τα βασικά χαρακτηριστικά που το καθιστούν μοναδικό είναι τα εξής :

- Ασύγχρονο και οδηγούμενο από γεγονότα : Όλες οι προγραμματιστικές διεπαφές (API's) από την βιβλιοθήκη του node.js είναι ασύγχρονες. Εκ των πραγμάτων αυτό σημαίνει ότι ένας εξυπηρετητής βασισμένος στο node.js δεν περιμένει ποτέ από ένα API να επιστρέψει δεδομένα καθώς συνεχίζει στο επόμενο API και λαμβάνει μία ειδοποίηση από ένα μηχανισμό γεγονότων από το node.js που βοηθά τον εξυπηρετητή να λάβει την απάντηση από το προηγούμενο API.
- Ταχύτητα : Από τη στιγμή που έχει υλοποιηθεί πάνω από το Google Chrome V8 JavaScript Engine , η βιβλιοθήκη node.js είναι πολύ γρήγορη στην εκτέλεση του κώδικά της.
- Δεν υπάρχει προσωρινός χώρος αποθήκευσης δεδομένων στις εφαρμογές που έχουν αναπτυχθεί με το node.js καθώς στέλνουν τα δεδομένα σε κομμάτια (chunks) .
- Κλιμακώσιμο αλλά μονού νήματος : Το Node.js κάνει χρήση του μοντέλου ενός νήματος με event-looping. Αυτός ο μηχανισμός βοηθάει τον εξυπηρετητή στο να ανταποκρίνεται με έναν non-blocking τρόπο και δίνει την ιδιότητα της

κλιμακώτητας διαφοροποιώντας τον από τους κλασσικούς εξυπηρετητές οι οποίοι δημιουργούν περιορισμένου αριθμού νημάτων για να διαχειριστούν τα αιτήματα πρόσβασης στην ιστοσελίδα. Το πρόγραμμα του ενός νήματος δίνει την δυνατότητα στον server να εξυπηρετήσει πολύ μεγαλύτερο αριθμό αιτημάτων από τους παραδοσιακούς servers όπως είναι ένας κλασσικός HTTP ή Apache.

Μερικές περιπτώσεις εφαρμογών που το Node.js μπορεί να χρησιμοποιηθεί με πολύ αποτελεσματικό τρόπο είναι οι εξής:

- Data streaming
- Μονής σελίδας
- Εφαρμογές βασισμένες σε JSON API
- I/O

Ε'.4.5) NPM(Node Package Manager)

Το NPM [46] αποτελεί το μεγαλύτερο μητρώο προγραμμάτων στον κόσμο και είναι ένα πρόγραμμα διαχείρισης και προσφοράς JavaScript πακέτων. Ένα πακέτο είναι ένας φάκελος ή ένα Directory το οποίο περιγράφεται από ένα package.json φάκελο. Ένα πακέτο επιβάλλεται να περιέχει ένα package.json φάκελο προκειμένου να αποθηκευτεί στη βάση δεδομένων registry. Ένα module είναι ένας φάκελος ή Directory που μπορεί να φορτωθεί στο κώδικα από το Node.js μέσω της συνάρτησης require(). Αποτελείται από 3 διαφορετικά στοιχεία :

- Την ιστοσελίδα
- Το Command Line Interface (CLI) , το οποίο βρίσκεται στον υπολογιστή κάθε χρήστη και είναι ο τρόπος με τον οποίο οι προγραμματιστές κάνουν install ,update και uninstall τα πακέτα.
- Ένα αρχείο registry το οποίο είναι μία δημόσια μεγάλη βάση δεδομένων από λογισμικό Javascript (Node packages) και τα Node modules.

E.4.5) Visual Studio Code

Όλος ο κώδικας (HTML,CSS,Solidity,JS,React.js) αναπτύχθηκε με το vsCode. Το Visual Studio Code [47] είναι ένα editor πηγαίου κώδικα. Αποτελεί προϊόν της Microsoft και η εφαρμογή είναι συμβατή για τα λειτουργικά συστήματα Windows , Linux και macOS. Οι χρήστες μπορούν να προσαρμόσουν το συγκεκριμένο πρόγραμμα επιλέγοντας τα χρώματα του κώδικα , του background , να προσθέσουν επεκτάσεις κατεβάζοντάς τις οι οποίες προσθέτουν πρόσθετες λειτουργίες και διευκολύνουν την ανάπτυξη κώδικα. Υποστηρίζει πολλές γλώσσες προγραμματισμού και διαθέτει για κάθε μία από αυτές διαφορετικές λειτουργίες. Ο χρήστης έχει την δυνατότητα να δημιουργεί και να αποθηκεύει κάθε project στη μορφή Directory τα

οποία αποθηκεύονται σε workspace για μελλοντική επαναχρησιμοποίηση. Η επέκταση του προγράμματος επιτυγχάνεται με την εγκατάσταση των extensions τα οποία είναι διαθέσιμα σε ένα κεντρικό repository. Τα περισσότερα extensions παρέχουν αλλαγή στον editor και υποστήριξη για όλες τις γλώσσες που είναι συμβατές το vscode. Αυτή η υποστήριξη περιέχει syntax highlighting , bracket matching , code folding και τροποποιήσιμα snippets κώδικα.

Ε'.4.6) Javascript

Η Javascript [48] είναι μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού στη σύγχρονη εποχή καθώς είναι ο κύριος τρόπος κατασκευής διαδικτυακών εφαρμογών. Πρόκειται Μπορεί να χρησιμοποιηθεί για την ανάπτυξη του Front-end αλλά ταυτόχρονα και του Back-end. Υποστηρίζεται από όλους τους σύγχρονους browsers και ο κώδικάς της μπορεί να εκτελεστεί από οτιδήποτε περιέχει ένα javascript engine. Μερικές περιπτώσεις χρήσεις της Javascript είναι οι εξής:

- Αλλαγή του κώδικα HTML της σελίδας , τροποποίηση του περιεχομένου.
- Αντίδραση σε events του χρήστη ενός browser όπως πάτημα ποντικιού,πάτημα κουμπιού , κούνημα κέρσορα κ.α.
- Αποστολή request σε απομακρυσμένους servers.
- Δημιουργία cookies
- Αποθήκευση δεδομένων στην πλευρά του χρήστη (local storage).

Διαθέτει έτοιμες προγραμματιστικές διεπαφές (API's) για την επεξεργασία κειμένου , για την τροποποίηση του DOM(Document Object Model) , την εύρεση της τοποθεσίας του client αλλά και για την δημιουργία γραφικών . Μερικά από τα πλεονεκτήματά της είναι τα εξής :

- Λιγότερη αλληλεπίδραση με τον Server καθώς τα δεδομένα που εισάγει ο χρήστης στην ιστοσελίδα μπορούν να επικυρωθούν και να αξιολογηθούν πριν σταλθούν στον Server αποφεύγοντας με αυτόν τον τρόπο αυξημένο φόρτο εργασίας.
- Κομψότερες διεπαφές όπου ο χρήστης μπορεί να κάνει drag and drop αντικείμενα από την ιστοσελίδα.
- Αυξημένη διαδραστικότητα . Δημιουργία διεπαφών που αντιδρούν στις κινήσεις του χρήστη μέσω του ποντικιού ή του πληκτρολογίου.
- Άμεση ανατροφοδότηση στους χρήστες της ιστοσελίδας. Εμφανίζονται μηνύματα που προσφέρουν ενημέρωση για την κατάσταση διεκπεραίωσης χρονοβόρων διεργασιών.

Ε'.4.7) Web3.js

Η έκδοση που χρησιμοποιήθηκε σε αυτή την εργασία είναι η 1.6.1 και αποτέλεσε ίσως το σημαντικότερο npm πακέτο για την υλοποίηση της εφαρμογής. Η web3.js [49] είναι μία συλλογή βιβλιοθηκών που κάνουν εφικτή και εύκολη την αλληλεπίδραση με έναν τοπικό ή απομακρυσμένο κόμβο σε ένα δίκτυο Ethereum χρησιμοποιώντας HTTP ή IPC σύνδεση. Εκ των πραγμάτων η web3.js δίνει την δυνατότητα στους προγραμματιστές να αναπτύξουν ιστοσελίδες που να αλληλεπιδρούν με ένα smart contract γράφοντας ή διαβάζοντας δεδομένα. Η επικοινωνία με ένα Ethereum Blockchain επιτυγχάνεται με την χρήση του JSON RPC (Remote Procedure Call) πρωτόκολλου. Η web3.js επιτρέπει την δημιουργία και την αποστολή request σε έναν κόμβο ενός Ethereum δικτύου με το JSON RPC προκειμένου να ανακτήσουν ή να μεταβάλλουν δεδομένα.

E'.4.8) React.js

Η react.js [50] είναι μία βιβλιοθήκη ανοιχτού λογισμικού της γλώσσας προγραμματισμού javascript , η οποία χρησιμοποιείται για την δημιουργία διεπαφών χρήστη. Ο κύριος σκοπός της είναι η δημιουργία και επαναχρησιμοποίηση των components αλλά και η αλλαγή των δεδομένων της ιστοσελίδας χωρίς την επαναφόρτωσή της. Κύρια χαρακτηριστικά της είναι η απλότητα , η ευκολία της στην εκμάθηση , η ταχύτητα και η επεκτασιμότητα.

E'.4.9) Solidity

Η Solidity [51] είναι η πιο γνωστή και χρησιμοποιούμενη γλώσσα προγραμματισμού για την ανάπτυξη κώδικα Smart Contracts. Μερικά από τα κύρια χαρακτηριστικά της είναι ότι είναι αντικειμενοστρεφής υψηλού επιπέδου , έχει επηρεαστεί κυρίως από την c++ και την Javascript με αποτέλεσμα να χαρακτηρίζεται curly-bracket γλώσσα , υποστηρίζει τις αρχές της κληρονομικότητας δημιουργώντας νέα smart contracts βασισμένα σε προηγούμενα , ο προγραμματιστής μπορεί να εισάγει βιβλιοθήκες χρησιμοποιώντας επαναχρησιμοποιήσιμο κώδικα από διαφορετικά smart contracts , πολύπλοκοι τύποι δεδομένων που καθορίζει ο χρήστης πχ structs,enums,contracts και είναι γλώσσα στατικού τύπου. Εκτελείται στο Ethereum Virtual Machine (EVM) και έχει δημιουργηθεί για να μεγιστοποιήσει τις ικανότητες της ιδεατής αυτής μηχανής. Ο πηγαίος κώδικας της γλώσσας Ethereum είναι γραμμένος με Solidity.

E'.5) Front-End

Η δημιουργία του Front-end υλοποιήθηκε με την χρήση της Javascript βιβλιοθήκης React.js .

E'.5.1) Components



Εικόνα 61: Όλα τα αρχεία του front-end

Στο directory voting-dapp περιέχονται όλα τα απαραίτητα αρχεία και modules που χρησιμοποιούνται για την ομαλή λειτουργία της εφαρμογής. Στον φάκελο components βρίσκονται όλες οι συναρτήσεις που επιστρέφουν ξεχωριστά κομμάτια του User Interface στα οποία εισάγονται και όλα τα .css αρχεία παρακάτω. Το αρχείο index.js είναι εκείνο στο οποίο ενσωματώνονται όλα τα functional components που δημιουργήσαμε και στο app.js περιέχονται όλα τα components μαζί.

Ε'.5.2) App.js

```
import './App.css';
import './Navbar.css'
import Admin from "../components/Admin.js"
import Voter from "../components/Voter.js"
import Homepage from "../components/homepage.js"
import Error from "../components/error.js"
import Navbar from "../components/navbar.js"
import Results from "../components/Results.js"
import {BrowserRouter as Router,Route,Routes,Link} from "react-router-dom"

function App() {

  return (

    <Router>
      <Navbar />
      <Routes>
        <Route path="/" element={<Homepage />}/>
        <Route path="/admin" element={<Admin />}/>
        <Route path="/voters" element={<Voter />}/>
        <Route path="/results" element={<Results />}/>
        <Route path="*" element={<Error />}/>
      </Routes>
    </Router>

  );
}

export default App;
```

Εικόνα 62: App.js Component

Εδώ περιέχονται όλα τα components και γίνεται η χρήση της React-router βιβλιοθήκης με το οποίο ο χρήστης καθοδηγείται σε διαφορετικές σελίδες ανάλογα με το αίτημα της αναζήτησης.

Ε'.5.3) Functional Components

Στη συνέχεια παραθέτω screenshots από όλο τον κώδικα του Registration.js component.

Τα περισσότερα components διέπονται από πανομοιότυπη λογική και κανόνες καθώς ο κώδικάς τους μοιάζει αρκετά. Υπάρχουν σχόλια σε κάθε συνάρτηση που βοηθούν στην κατανόηση της λειτουργίας της.

Αρχικά γίνονται τα απαραίτητα imports του css αρχείο και των βιβλιοθηκών που χρησιμοποιούνται. Έπειτα δηλώνεται η συνάρτηση του component η οποία παίρνει ως παράμετρο τα props των οποίων η χρησιμότητά τους είναι η μεταφορά πληροφοριών από component σε component. Γίνεται η δήλωση 2 μεταβλητών και 2 συναρτήσεων με τη useState βιβλιοθήκη, η οποία ανήκει στην κατηγορία των React Hooks και η χρησιμότητά της είναι πως όταν αλλάζει η τιμή της μεταβλητής με την συνάρτηση που ορίστηκαν, το component θα γίνει re-rendered χωρίς την απαίτηση να γίνει ανανέωση της σελίδας. Με αυτόν τον τρόπο αλλάζει η εικόνα του front-end γρήγορα και αποτελεσματικά. Τέλος οι τιμές που περιέχει το object props, καταχωρούνται μία προς μία στις local μεταβλητές που ορίζονται.

```
import React from 'react';
import {useState} from "react";
import "../admin.css";

export default function Registration(props) {
  //Error state Variables
  let [isRegistered,setIsRegistered]=useState(null);
  let [isNotRegistered,setIsNotRegistered]=useState(null);
  let [inputError,setInputError]=useState(null);

  //Take the props
  const {isConnected,registerError,checkRegisterError,onlyAdmin,contrInst,browser_account}=props;
```

Εικόνα 63: Μεταβλητές του Registration.js

Παρακάτω γίνεται η δήλωση της checkRegistered συνάρτησης, η οποία εκτελείται όταν πατιέται το κουμπί από τον διαχειριστή. Ρόλος της είναι να τσεκάρει αν η διεύθυνση που παραθέτεται ως input, είναι εγγεγραμμένη και έχει την δυνατότητα να ψηφίσει.

Καταρχάς παίρνει σαν όρισμα το event του πατήματος του κουμπιού και με την χρήση της συνάρτησης preventDefault ακυρώνει την προδιαγεγραμμένη συμπεριφορά του. Έπειτα καλείται η συνάρτηση onlyAdmin, η προέλευση της οποίας είναι το Admin component και έργο της η επιβεβαίωση ότι ο λογαριασμός που εκτελεί την συνάρτηση είναι ο administrator της ιστοσελίδας, αποθηκεύεται το value από το input field σε μία τοπική μεταβλητή, γίνεται ο έλεγχος του και αν «περάσει» μετατρέπεται σε lower case και καλείται η συνάρτηση του smart contract που ελέγχει αν είναι αποθηκευμένη αυτή η διεύθυνση. Αν δεν περάσει τον έλεγχο Τέλος με ένα if statement, γίνεται ο έλεγχος της τιμής που επιστράφηκε από την συνάρτηση. Στην περίπτωση που η διεύθυνση δεν έχει καταχωρηθεί θα εμφανιστεί το κατάλληλο μήνυμα στο UI και το ίδιο θα συμβεί στην

αντίθετη

περίπτωση.

```

//Button function that checks
// if the address in the input field is registered
// in the smart contract
async function checkRegistered(e){
    //Cancel the default button behavior
    setInputError(null);
    e.preventDefault();

    //Check if the user trying to execute the function is the Administrator.
    onlyAdmin(1);

    //Get the input text , convert it to lowerCase ,
    // execute the function from the solidity file to check if the input is registered
    let voter_addr=document.querySelector("#voter-address2").value;
    if(voter_addr=="" || voter_addr.length<42 || voter_addr.length>42){
        setInputError("Your input has to be 42 characters");
    }else{
        let new_voter_addr=voter_addr.toLowerCase();
        let response=await contrInst.methods.isRegistered(new_voter_addr).call();

        //Check if the result is true or false to Set/Unset the error variables
        if(response){
            setIsRegistered(response);
            setIsNotRegistered(false);
        }else{
            setIsNotRegistered(!response);
            setIsRegistered(false);
        }
    }
}

```

Εικόνα 64 : Συνάρτηση ελέγχου εγγραφής του αρχείου Registration.js

Εδώ γίνεται η δήλωση της συνάρτησης `registerVoters`. Με παρόμοιο τρόπο γίνεται η ακύρωση της συμπεριφοράς του κουμπιού, ελέγχεται αν η συνάρτηση καλείται από τον `admin`, γίνεται έλεγχος του `input value`, μετατρέπεται σε lower case και εκτελείται η συνάρτηση `registerVoters` του smart contract η οποία δημιουργεί μία συναλλαγή και αποθηκεύει την διεύθυνση στο blockchain. Αν ο έλεγχος αποτύχει τυπώνεται το κατάλληλο μήνυμα στην οθόνη.

```
//Button function for registration
async function registerVoters(e){
  e.preventDefault();
  setInputError(null);
  onlyAdmin(0);

  //Take the input text , convert it to lowerCase
  // then call the solidity function to register
  // it in the blockchain with a form of a transaction
  let voter_addr=document.querySelector("#voter-address1").value;
  if(voter_addr==" || voter_addr.length<42 || voter_addr.length>42){
    setInputError("Your input has to be 42 characters");
  }else{
    let new_voter_addr=voter_addr.toLowerCase();
    let response=await contrInst.methods.registerVoters(new_voter_addr).send({from:browser_account});
  }
}
```

Εικόνα 65: Συνάρτηση εγγραφής ψηφοφόρων του αρχείου *Registration.js*

Στις 2 τελευταίες εικόνες απεικονίζεται όλη η html που επιστρέφεται στον χρήστη. Εφαρμόζονται τα conditional statements τα οποία ανάλογα με τις τιμές που έχουν οι state μεταβλητές εμφανίζεται και το αντίστοιχο περιεχόμενο. Για παράδειγμα αν η διεύθυνση δεν έχει εγγραφεί τότε εμφανίζεται το μήνυμα “This user is not registered”.

```
return (
  <div className="item3 centered">
    {isConnected &&
      <form>
        <label htmlFor="voter-address1">Register Voter</label>
        <br></br>
        <input type="text" id="voter-address1" placeholder="voter address" required></input>
        <button type="submit" className="submitButton startButton" onClick={registerVoters}>register</button>
      </form>
      {inputError && <h2 className="errors">{inputError}</h2>
    }
    {registerError &&
      <div>
        <h4 className="errors">{registerError}</h4>
      </div>
    }
    {isConnected &&
      <form>
        <label htmlFor="voter-address2">Check Registration</label>
        <br></br>
        <input type="text" id="voter-address2" placeholder="check if registered"></input>
        <button type="submit" className="submitButton startButton" onClick={checkRegistered}>Check Registration</button>
      </form>
    }
  </div>
)
```

Εικόνα 66: *JSX .1*

```
{checkRegisterError &&  
  <div>  
    <h4 className="errors">{checkRegisterError}</h4>  
  </div>  
{isNotRegistered &&  
  <div className="centered errors">  
    {isNotRegistered && <h4>This user is not Registered</h4>}  
  </div>  
}  
{isRegistered &&  
  <div className="centered correct">  
    {isRegistered && <h4>This user is Registered</h4>}  
  </div>  
</div>
```

Εικόνα 67: JSX.2

Κεφάλαιο ΣΤ΄

ΣΤ΄) Μελλοντικές βελτιώσεις / Συμπεράσματα

Στο τελευταίο αυτό κεφάλαιο περιγράφονται τα συμπεράσματά μου από την υλοποίηση της εφαρμογής καθώς και πιθανές και απαραίτητες προτάσεις βελτιστοποίησης της παρούσας εφαρμογής.

Περνώντας στην εποχή του Web.3 , πολλές από τις κορυφαίες εφαρμογές της τωρινής εποχής θα αλλάξουν ριζικά καθώς θα υιοθετήσουν την λογική της αποκέντρωσης και της απεξάρτησής τους από έναν κεντρικό Server. Σταδιακά η τεχνολογία του blockchain , πέρα από απλά web applications θα εφαρμοστεί και σε ύψιστης σημασίας applications όπως οι ηλεκτρονικές ψηφοφορίες σε τοπικό , αλλά γιατί όχι και σε εθνικό επίπεδο. Τα πλεονεκτήματα που προσφέρουν τέτοιου τύπου εφαρμογές είναι πολύ μεγαλύτερα σε αξία από ότι το κόστος της ολικής αλλαγής τους μακροπρόθεσμα καθώς θα υπάρχει απόλυτη διαφάνεια των καταγραφών/συναλλαγών . Οι ίδιες οι κοινωνικές συνθήκες είναι αυτές που απαιτούν ένα σύστημα ψηφοφορίας το οποίο θα είναι απολύτως αξιόπιστο και τα δεδομένα του να είναι δημοσίως προσβάσιμα για ανάγνωση, καθώς οι περιπτώσεις εκλογικής απάτης και διαφθοράς δεν αποτελούν πλέον μειονότητα.

Σημαντικές βελτιώσεις :

- Βελτιστοποίηση του UI (user interface) της εφαρμογής ώστε να γίνει πιο φιλικό και προσιτό προς τον χρήστη.
- Κατάργηση της εξάρτησης της εφαρμογής από το metamask προσθέτοντας περισσότερες επιλογές ψηφιακών πορτοφολιών καθώς δεν το χρησιμοποιούν όλοι.
- Δημιουργία ενός δεύτερου smart contract το οποίο θα εφαρμόζει το ERC-20 πρότυπο , όπου ο διαχειριστής θα «τυπώνει» έναν συγκεκριμένο αριθμό από tokens , θα τα διανείμει σε όλους τους ψηφοφόρους και αυτοί με την σειρά τους θα ψηφίζουν με αυτά.
- Υλοποίηση της εφαρμογής σε διαφορετικό blockchain και με διαφορετικά εργαλεία έτσι ώστε να αποφευχθούν τα gas fees.
- Παροχή ανωνυμίας ως προς τις διευθύνσεις των ψηφοφόρων από τον διαχειριστή του συστήματος έτσι ώστε να μην είναι εφικτή η σύνδεση μιας ψήφου με την διεύθυνση των χρηστών από τον διαχειριστή.

Παράρτημα Ι : Κώδικας

Στο παράρτημα αυτό καταθέτεται όλος ο κώδικας του Smart Contract που αναπτύχθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας. Ο πηγαίος κώδικας του VotingSystem Smart Contract γραμμένος στην γλώσσα προγραμματισμού Solidity.

```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract VotingSystem{

    enum State {RegisteringVoters,
    ProposalsRegistrationStarted,
    ProposalsRegistrationEnded,
    VotingSessionStarted,
    VotingSessionEnded,VotesTallied}

    struct Candidate{
        string description;
        uint votes;
    }
    struct Voter{
        bool is_registered;
        bool did_voted;
        bool did_propose;
        uint candidateId;
    }

    address public administrator;
    State public voting_state;
    mapping(address=>Voter) public all_voters;
    Candidate[] public all_candidates;
    uint private candidateWinnerId;
```

```
//MODIFIERS
modifier onlyAdmin(){
    require(msg.sender==administrator,"Only the system admin can call this function");
    _;
}

modifier onlyRegistered(){
    require(all_voters[msg.sender].is_registered==true,"You are not registered");
    _;
}

modifier notRegistered(address _address){
    require(all_voters[_address].is_registered==false,"You are registered");
    _;
}

modifier newProposalOnly(string memory _description){
    for(uint i=0;i<all_candidates.length;i++){
        Candidate memory cand_obj=all_candidates[i];
        require(keccak256(abi.encodePacked((cand_obj.description)))!=keccak256(abi.encodePacked(_description)),"the candidate is already proposed");
    }
    _;
}
}
```

```
modifier setProposalOnce(){
    require(all_voters[msg.sender].did_propose==false,"You can't propose more than one time");
    _;
}

modifier duringVoterRegistration(){
    require(voting_state==State.RegisteringVoters,"Not in the voter registration phase");
    _;
}

modifier duringProposalRegistration(){
    require(voting_state==State.ProposalsRegistrationStarted,"Not in the proposal registration phase");
    _;
}

modifier duringVotingPhase(){
    require(voting_state==State.VotingSessionStarted,"Not in the voting phase");
    _;
}
}
```

```
modifier AfterVotingPhase(){
    require(voting_state==State.VotingSessionEnded,"Not in the post-voting phase");
    _;
}

modifier duringVotesTallied(){
    require(voting_state==State.VotesTallied,"Not in the votes tallied phase");
    _;
}

modifier duringProposalRegistrationEnd(){
    require(voting_state==State.ProposalsRegistrationEnded,"Not in the proposalRegistrationEnd");
    _;
}
}
```



```
//FUNCTIONS
constructor(){
    administrator=msg.sender;
    voting_state=State.RegisteringVoters;
}

function registerVoters(address voter_address)
external onlyAdmin duringVoterRegistration()
notRegistered(voter_address)
{
    all_voters[voter_address]=Voter(true,false,false,0);
}

function startProposalRegistration()
external onlyAdmin() duringVoterRegistration()
{
    voting_state=State.ProposalsRegistrationStarted;
}

function setProposal(string memory _description)
external onlyRegistered() duringProposalRegistration()
newProposalOnly(_description) setProposalOnce()
{
    all_candidates.push(Candidate(_description,0));
    all_voters[msg.sender].did_propose=true;
}
}
```

```
function endProposalRegistration()external onlyAdmin() duringProposalRegistration(){
    voting_state=State.ProposalsRegistrationEnded;
}

function startVotingSession()external onlyAdmin() duringProposalRegistrationEnd(){
    voting_state=State.VotingSessionStarted;
}

function vote(uint _candidateId)
external onlyRegistered duringVotingPhase
{
    require(all_voters[msg.sender].did_voted==false,"this address has already voted");
    all_voters[msg.sender].did_voted=true;
    all_voters[msg.sender].candidateId=_candidateId;
    all_candidates[_candidateId].votes+=1;
}

function endVotingSession()
external onlyAdmin() duringVotingPhase()
{
    voting_state=State.VotingSessionEnded;
}
}
```

```

function tallyVotes()
external onlyAdmin AfterVotingPhase()
{
    uint winningVoteCount = 0;
    uint winningProposalIndex = 0;
    for(uint i=0;i<all_candidates.length;i++){
        if(all_candidates[i].votes>winningVoteCount){
            winningVoteCount=all_candidates[i].votes;
            winningProposalIndex=i;
        }
    }
    candidateWinnerId=winningProposalIndex;
    voting_state=State.VotesTallied;
}

function getCandidatesNumbers()
external view returns(uint)
{
    return all_candidates.length;
}

function getCandidateDescription(uint candidateId)
external view returns(string memory)
{
    return all_candidates[candidateId].description;
}

```

```
function getCandidateWinnerId()
external view duringVotesTallied() returns(uint)
{
    return candidateWinnerId;
}

function getCandidateWinnerDescription()
external view duringVotesTallied() returns(string memory)
{
    return all_candidates[candidateWinnerId].description;
}

function getTotalVotes()
external view duringVotesTallied() returns(uint)
{
    return all_candidates[candidateWinnerId].votes;
}

function isRegistered(address _voter_address)
external view returns(bool)
{
    return all_voters[_voter_address].is_registered;
}

function getVotingStatus()
external view returns(State)
{
    return voting_state;
}
```

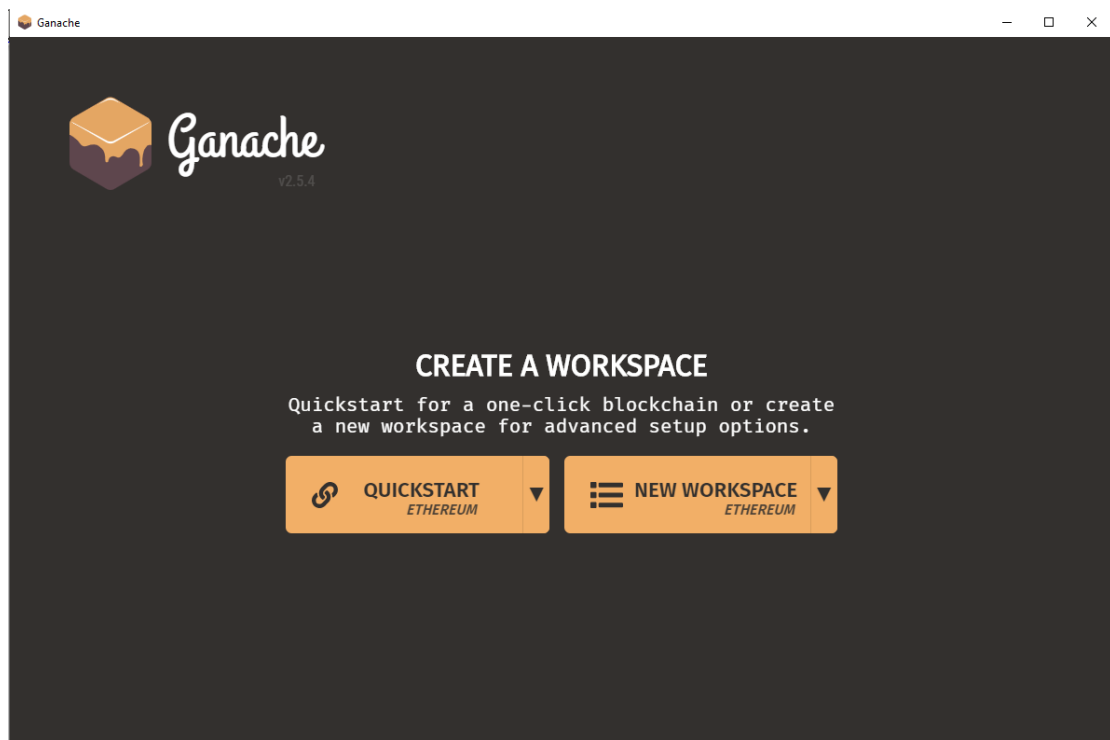

Παράρτημα II Οδηγίες Εγκατάστασης

Προκειμένου κάποιος να χρησιμοποιήσει την εφαρμογή πρέπει να εγκαταστήσει τα παρακάτω εργαλεία στον υπολογιστή του :

- Ganache
- Truffle
- Metamask browser extension
- Τον κώδικα του project

Βήμα 1

Αφού εγκαταστήσετε την εφαρμογή Ganache από την ιστοσελίδα [42] και εκτελέσετε το πρόγραμμα θα σας εμφανιστεί η παρακάτω εικόνα:



Πατήστε την επιλογή quickstart και στην συνέχεια θα έχετε πρόσβαση σε 10 ethereum λογαριασμούς τα οποία περιέχουν 100eth μαζί με τα ιδιωτικά τους κλειδιά.

ACCOUNTS

BLOCKS

TRANSACTIONS

CONTRACTS

EVENTS

LOGS

SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK

4

GAS PRICE

20000000000

GAS LIMIT

6721975

HARDFORK

MUIRGLACIER

NETWORK ID

5777

RPC SERVER

HTTP://127.0.0.1:7545

MINING STATUS

AUTOMINING

WORKSPACE

QUICKSTART

SAVE

SWITCH

MNEMONIC

exotic music soda chimney exhibit secret mirror junk minute clinic hood mutual

HD PATH

m/44'/60'/0'/0/account_index

ADDRESS

0x1D09F75ba5842805D1A14DBE578398f77148aA71

BALANCE

99.95 ETH

TX COUNT

4

INDEX

0

ADDRESS

0x74ef9008581d3a98ECd272Fd63b571B35cC3327b

BALANCE

100.00 ETH

TX COUNT

0

INDEX

1

ADDRESS

0x5b2b925f62AE9A455BF88E2e1e77447b136bDf12

BALANCE

100.00 ETH

TX COUNT

0

INDEX

2

ADDRESS

0xcb42632F7cE979e7be48B7BA43F0CB33b06B5D16

BALANCE

100.00 ETH

TX COUNT

0

INDEX

3

ADDRESS

0x46f3Dc007a58a5D03C50588DC80ECF60e95DfF6F

BALANCE

100.00 ETH

TX COUNT

0

INDEX

4

ADDRESS

0x32B0441A12fd4307219919F00E6590769e4e0Cf3

BALANCE

100.00 ETH

TX COUNT

0

INDEX

5

ADDRESS

0xB7bBa173634Aa7CdF261bd7F28138aC71f933342

BALANCE

100.00 ETH

TX COUNT

0

INDEX

6

Βήμα 2

Ανοίξτε το directory project votingSystem και μεταβείτε στον φάκελο **truffle-config.js**. Εκεί βρίσκονται όλες οι επιλογές τροποποίησης για το δίκτυο στο οποίο θα γίνει deploy το smart contract. Στο object module.exports πρέπει να βάλετε ως host την διεύθυνση που τρέχει το Ganache , ως port το 7545 και ως network_id την *. Όσο αναφορά το contracts_build_directory , καθορίζει το σημείο στο οποίο θα αποθηκευτεί το abi του smart contract μετά την μεταγλώττισή του από το truffle. Είναι σημαντικό διότι το χρησιμοποιούμε για να δημιουργήσουμε το contract_instance του smart contract με το οποίο αλληλεπιδρά ο χρήστης της εφαρμογής.

```
module.exports = {
  contracts_build_directory: "./voting-dapp/src/contracts_abi",
  networks: {
    development: {
      host: "127.0.0.1",      // Localhost (default: none)
      port: 7545,           // Standard Ethereum port (default: none)
      network_id: "*",      // Any network (default: none)
    },
  },
}
```

Βήμα 3

Ανοίξτε ένα τερματικό , μεταβείτε με την εντολή `cd votingSystem` και έπειτα εκτελέστε την εντολή **truffle migrate** για να γίνει deploy το smart contract στο δίκτυο στο οποίο είναι συνδεδεμένο το Ganache Blockchain με τις παραμετροποιήσεις του προηγούμενου βήματος. Κάθε φορά που θέλετε να ξανακάνετε deploy το smart contract στο δίκτυο για να αλληλεπιδράτε με τις αρχικές του τιμές θα πρέπει να εκτελέσετε την εντολή **truffle migrate - - reset**. Ο λογαριασμός που καταθέτει το smart contract στο δίκτυο και συνεπώς ο administrator του συστήματος ψηφοφορίας θα είναι πάντα ο πρώτος λογαριασμός από το Ganache Blockchain. Υπάρχει και η επιλογή χρήσης του Ganache cli με περισσότερες λειτουργίες αλλά δεν έγινε η χρήση του στην παρούσα πτυχιακή εργασία. Το αποτέλεσμα των προηγούμενων εντολών φαίνεται στην παρακάτω εικόνα :

```
2_VotingSystem_migration.js
=====

Replacing 'VotingSystem'
-----
> transaction hash: 0xddc07967cea6b172f121d27fb2175ff1a8ba811e7a45188e2d02f5cca385d733
> Blocks: 0 Seconds: 0
> contract address: 0x137CF7d0B5303cEB755587Bb4d86d35d4E1a082C
> block number: 3
> block timestamp: 1645196241
> account: 0x1D09F75ba5842805D1A14DBE578398f77148aA71
> balance: 99.95224198
> gas used: 2096534 (0x1ffd96)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.04193068 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.04193068 ETH

Summary
=====
> Total deployments: 2
> Final cost: 0.04690776 ETH

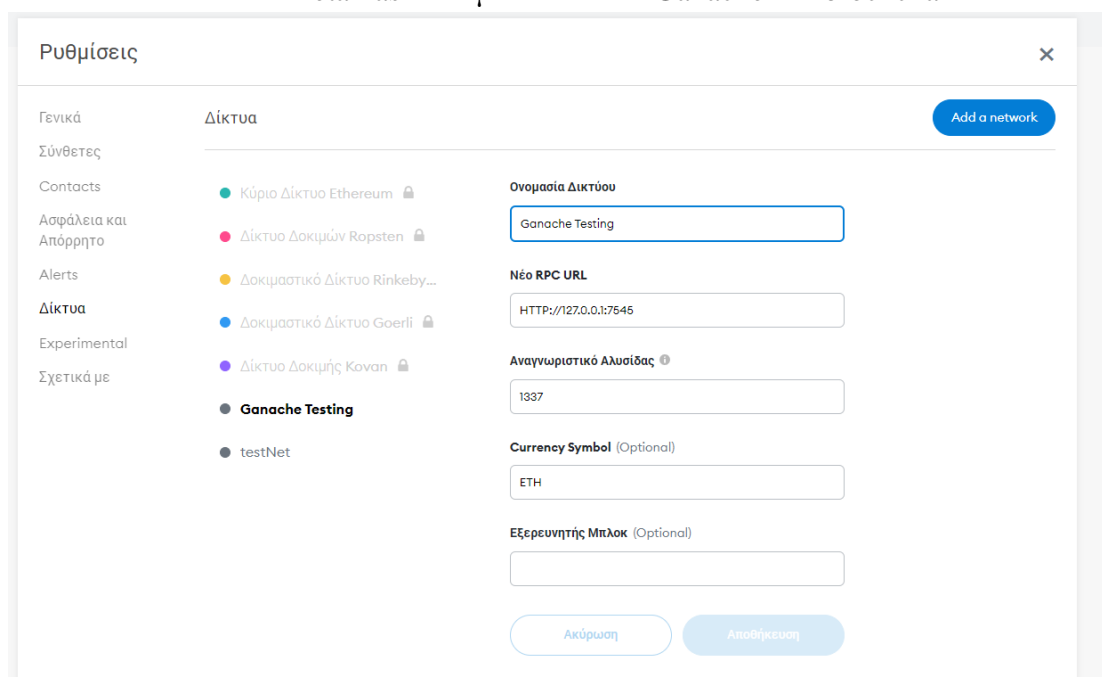
C:\Users\kakos\Desktop\votingSystem>
```

Προφανώς το Eth που χρησιμοποιείται για όλες τις συναλλαγές και για το smart contract deployment στο ganache είναι «ψεύτικο». Στην περίπτωση όμως που αποφασίσετε να μεταφέρετε το smart contract σε κάποιο testnet όπως πχ Rinkeby, Ropsten, Gorli κ.α, θα πρέπει να ζητήσετε να σας καταχωρίσουν μία μικρή ποσότητα eth σε έναν έγκυρο λογαριασμό. Αν θελήσετε να χρησιμοποιήσετε το mainnet θα πρέπει να πληρώσετε με «πραγματικό» Eth.

Βήμα 4

Ανοίξτε ένα browser κατά προτίμηση Google Chrome και εγκαταστήστε το browser extension Metamask από την ιστοσελίδα [43]. Αυτό θα αποτελέσει το ψηφιακό σας πορτοφόλι με το οποίο θα υπογράφονται και θα πραγματοποιούνται όλες οι

συναλλαγές σας και θα καταγράφονται στο local blockchain . Αφού γίνει η εγκατάσταση μεταβείτε στις ρυθμίσεις του και φτιάξτε ένα νέο local RPC δίκτυο με τις ρυθμίσεις που φαίνονται στην παρακάτω εικόνα. Με αυτόν τον τρόπο θα συνδέσετε το metamask με το Ganache blockchain δίκτυο.

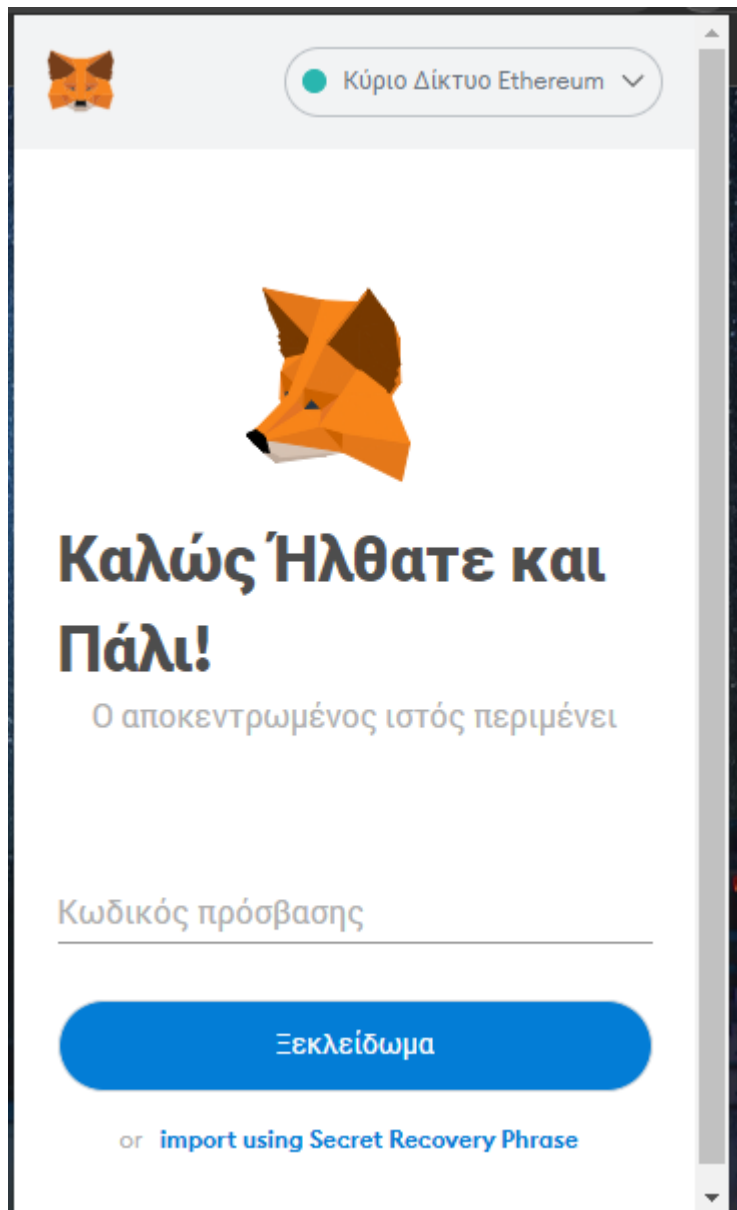


Βήμα 5

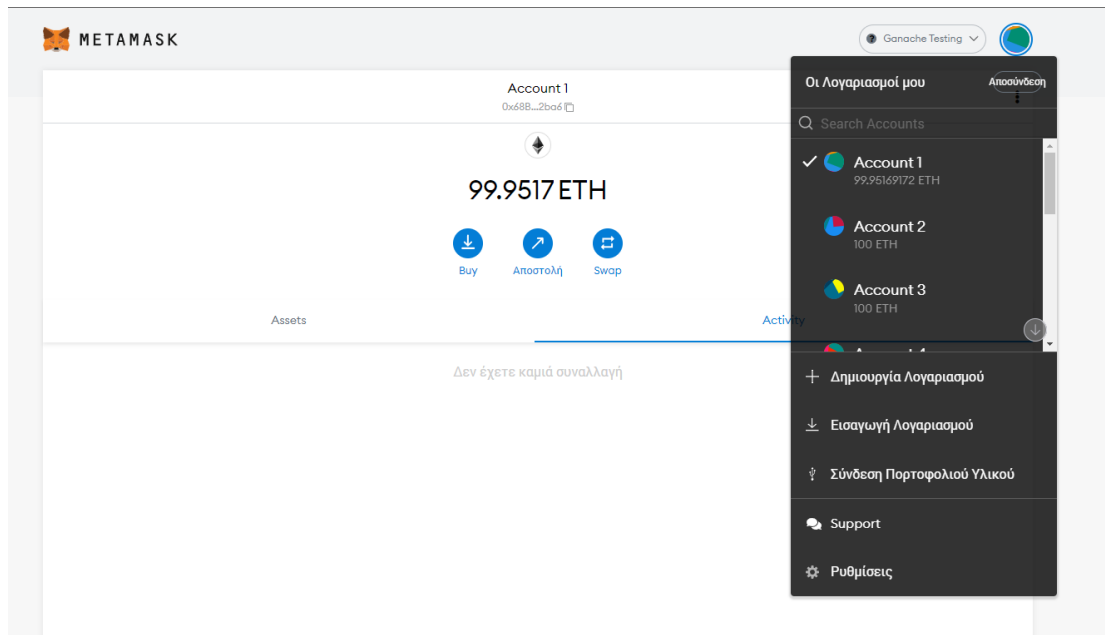
Ανοίξτε ένα τερματικό από τον υπολογιστή σας , μεταβείτε στο αρχείο του project votingSystem και έπειτα εκτελέστε την εντολή **cd voting-dapp**. Εκεί βρίσκονται όλα τα αρχεία που δομούν το front-end της εφαρμογής. Εκτελέστε την εντολή **npm run start** προκειμένου να δημιουργηθεί ο server στο προεπιλεγμένο port. Αυτόματα θα ανοίξει ο browser σας με την αρχική σελίδα της εφαρμογής.

Βήμα 6

Αφού βρίσκεται στην αρχική σελίδα της εφαρμογής μεταβείτε από το Navbar που βρίσκεται στο πάνω μέρος της οθόνης στην σελίδα του Admin. Προτού πατήσετε το connect button που βρίσκεται στην οθόνη , ανοίξτε το metamask extension που βρίσκεται πάνω δεξιά. Η εικόνα που πρέπει να σας εμφανιστεί είναι αυτή :



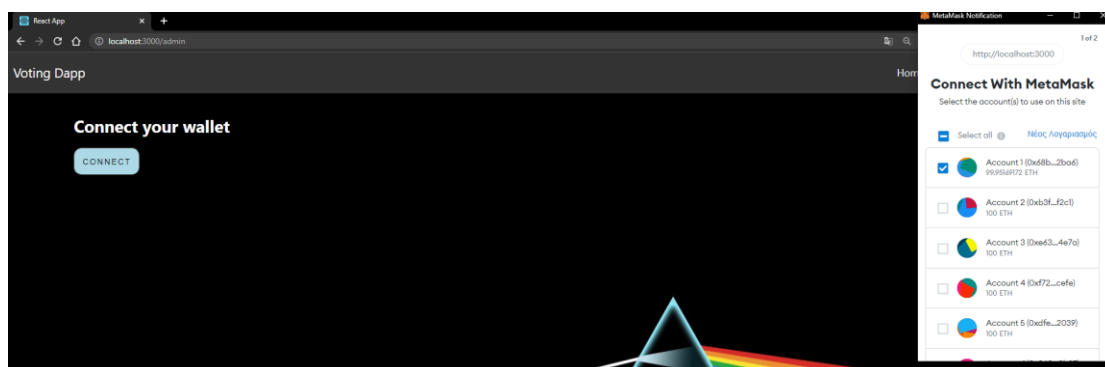
Σε αυτήν την περίπτωση θα επιλέξετε την επιλογή import using Secret Recovery Phase. Έπειτα θα μεταβείτε σε μία άλλη σελίδα η οποία θα σας παροτρύνει να πληκτρολογήσετε το wallet secret recovery phrase. Εκεί θα κάνετε αντιγραφή/επικόλληση το Mnemonic phrase που βρίσκεται στην αρχική σελίδα του Gnapache μαζί με έναν καινούργιο κωδικό πρόσβασης ή κάποιον από έναν λογαριασμό που ήδη διαθέτετε. Το αποτέλεσμα των παραπάνω πράξεων εμφανίζεται στην παρακάτω εικόνα :



Όπως παρατηρείται υπάρχουν στο metamask και οι 10 λογαριασμοί του ganache με υπόλοιπο 100eth εκτός από την πρώτη διεύθυνση που είναι ο διαχειριστής και έκανε το deploy του smart contract. Απαραίτητη προϋπόθεση είναι να βρίσκεστε στο τοπικό δίκτυο που φτιάξατε και όχι στο mainnet διότι εκεί οι λογαριασμοί διαθέτουν 0 Eth.

Βήμα 7

Αφού έχετε υλοποιήσει όλα τα παραπάνω βήματα με επιτυχία μεταβείτε στην σελίδα του administrator αφού πρώτα κάνετε ανανέωση της σελίδας και πατήστε το connect button. Δεξιά της οθόνης θα σας εμφανιστεί το παρακάτω μήνυμα το οποίο σας προτρέπει να συνδέσετε τους λογαριασμούς με την εφαρμογή.



Πατήστε σύνδεση και η εφαρμογή είναι έτοιμη για να χρησιμοποιηθεί.

Βιβλιογραφία

- [1] <https://bitcoin.org/bitcoin.pdf>
- [2] <https://link.springer.com/content/pdf/10.1007/s12599-017-0467-3.pdf>
- [3] <https://academy.binance.com/en/articles/history-of-blockchain>
- [4] <https://www.sciencedirect.com/science/article/pii/S0065245818300664>
- [5] <https://www.euromoney.com/learning/blockchain-explained/how-transactions-get-into-the-blockchain>
- [6] https://www.researchgate.net/publication/337306138_Blockchain_for_Dynamic_Spectrum_Management
- [7] <https://www.indeed.com/career-advice/career-development/what-is-a-peer-to-peer-network>
- [8] <https://101blockchains.com/blockchain-nodes/>
- [9] <https://ieeexplore.ieee.org/abstract/document/8823094>
- [10] hash functions
- [11] https://www.researchgate.net/publication/325136332_A_Decentralised_Approach_to_Task_Allocation_Using_Blockchain
- [12] <https://academy.horizen.io/technology/expert/blockchain-as-a-data-structure/>
- [13] https://link.springer.com/chapter/10.1007%2F978-3-030-33495-6_20
- [14] <https://www.sciencedirect.com/science/article/pii/S0957417420302098>
- [15] <https://arxiv.org/pdf/2001.07091.pdf>
- [16] <https://arxiv.org/pdf/2102.10006.pdf>
- [17] <https://www.investopedia.com/terms/s/soft-fork.asp>
- [18] <https://www.investopedia.com/terms/h/hard-fork.asp>
- [19] <https://link.springer.com/article/10.1007/s10586-018-2387-5#Sec8>
- [20] <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9323061>
- [21] <https://decentralizedthoughts.github.io/2020-02-26-selfish-mining/>
- [22] <https://www.gemini.com/cryptopedia/eclipse-attacks-defense-bitcoin>

- [23] <https://preethikasireddy.medium.com/how-does-ethereum-work-anyway-22d1df506369>
- [24] <https://github.com/ethereumbook/ethereumbook/blob/develop/07smart-contracts-solidity.asciidoc>
- [25] <https://ethereum.github.io/yellowpaper/paper.pdf>
- [26] <https://preethikasireddy.medium.com/how-does-ethereum-work-anyway-22d1df506369>
- [27] https://www.researchgate.net/figure/Execution-process-of-Ethereum-virtual-machine_fig5_349918758
- [28] <https://www.investopedia.com/terms/g/gas-ethereum.asp>
- [29] <https://github.com/ethereumbook/ethereumbook/blob/develop/12dapps.asciidoc>
- [30] <https://ethereum.org/en/developers/docs/standards/tokens/>
- [31] <https://ethereum.org/en/developers/docs/standards/tokens/erc-20/>
- [32] <https://ethereum.org/en/developers/docs/standards/tokens/erc-721/>
- [33] https://en.wikipedia.org/wiki/Non-fungible_token#Uses
- [34] <https://arxiv.org/pdf/2105.07447.pdf>
- [35] <https://ieeexplore.ieee.org/document/8457919>
- [36] <https://ieeexplore.ieee.org/abstract/document/8726645/references#references>
- [37] <https://ieeexplore.ieee.org/abstract/document/9333937>
- [38] <https://ieeexplore.ieee.org/abstract/document/9143877>
- [39] https://link.springer.com/chapter/10.1007/978-3-319-70972-7_20
- [40] <https://arxiv.org/ftp/arxiv/papers/1802/1802.10134.pdf>
- [41] <https://ieeexplore.ieee.org/document/8611593>
- [42] <https://trufflesuite.com/ganache/>
- [43] <https://metamask.io/>
- [44] <https://trufflesuite.com/>
- [45] <https://www.tutorialspoint.com/nodejs/index.htm>
- [46] <https://docs.npmjs.com/about-npm>
- [47] https://en.wikipedia.org/wiki/Visual_Studio_Code
- [48] https://www.tutorialspoint.com/javascript/javascript_overview.htm
- [49] <https://web3js.readthedocs.io/en/v1.7.0/>

[50] <https://reactjs.org/docs/getting-started.html>

[51] <https://docs.soliditylang.org/en/v0.8.11/>