ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ

Blockchain και Έξυπνα Συμβόλαια

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Στέφανος Μαυροζούμης

Επιβλέπων: Πέτρος Στεφανέας
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2021

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ ΚΑΙ ΦΥΣΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΟΜΕΑΣ ΜΑΘΗΜΑΤΙΚΩΝ

Blockchain και Έξυπνα Συμβόλαια

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Στέφανος Μαυροζούμης

Επιβλέπων: Πέτρος Στεφανέας
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Φεβρουαρίου 2021

……	……..	……...
Πέτρος Στεφανέας Επίκουρος Καθηγητής Ε.Μ.Π.	Αντώνιος Συμβώνης Καθηγητής Ε.Μ.Π.	Αριστείδης Παγουρτζής Αναπληρωτής Καθηγητής Ε.Μ.Π.

……..
Στέφανος Μαυροζούμης

Διπλωματούχος Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών Ε.Μ.Π.

Περίληψη

Το Διαδίκτυο σήμερα, σε συνδυασμό με την ολοένα αναπτυσσόμενη ψηφιακή τεχνολογία, έχει δημιουργήσει μία τεράστια αγορά γνώσεων/πληροφοριών. Η εφεύρεση του Διαδικτύου έχει οδηγήσει σε μια επανάσταση στον τρόπο που ο καθένας αντιλαμβάνεται το κόσμο γύρω του. Ένα κοινό χαρακτηριστικό που παρατηρείται σε όλο το εύρος του (διαδικτυακές εφαρμογές, υπηρεσίες) είναι η αρχιτεκτονική πελάτη-εξυπηρετητή. Με την αρχιτεκτονική αυτή προκύπτει μία συγκέντρωση γνώσης που σχετίζεται άμεσα με το Διαδίκτυο. Το μεγάλο ερώτημα που ανακύπτει έτσι είναι το "ποιος διοικεί, ποιος ελέγχει την γνώση αυτή". Στην προσπάθεια απαλλαγής από αυτή τη συγκέντρωση της πληροφορίας κεντρικά, έχει δημιουργηθεί μια τάση αποκεντροποίησης των υπηρεσιών του Διαδικτύου. Η τεχνολογία που είναι ο κινητήριος μοχλός προς αυτήν την κατεύθυνση δεν είναι άλλη από το Blockchain.

Η επαναστατική ιδέα του Blockchain, που είναι η βασική τεχνολογία πίσω από το διάσημο κρυπτονόμισμα Bitcoin, έχει πυροδοτήσει την έναρξη μιας νέας εποχής στο Διαδίκτυο. Το Blockchain είναι ένα κατανεμημένο ledger (κατάστιχο συναλλαγών) για την καταγραφή συναλλαγών με τρόπο αδιάβλητο, που διατηρείται από ένα δίκτυο ομότιμων κόμβων (υπολογιστών) P2P, οι οποίοι δεν εμπιστεύονται απόλυτα ο ένας τον άλλο. Τα δεδομένα εισόδου από την στιγμή που θα εισέλθουν στην αλυσίδα (chain) δεν μπορούν να αλλάξουν. Αυτό επιτυγχάνεται μέσω ισχυρών κρυπτογραφικών αρχών, όπως είναι οι ψηφιακές υπογραφές, οι συναρτήσεις κατακερματισμού και τα Merkle Trees.

Ενώ οι περισσότεροι σήμερα συνδέουν το Blockchain με τα κρυπτονομίσματα, στην πραγματικότητα είναι κάτι πολύ μεγαλύτερο. Το Blockchain είναι πλέον ο θεμέλιος λίθος των αποκεντρωμένων εφαρμογών, δηλαδή εφαρμογών που βασίζονται σε ένα δίκτυο ομότιμων κόμβων και όχι στην αρχιτεκτονική πελάτη-εξυπηρετητή. Μια από τις πιο διαδεδομένες και ισχυρές πλατφόρμες ανάπτυξης αποκεντρωμένων εφαρμογών μέσω έξυπνων συμβολαίων είναι το Ethereum Blockchain.

Η ηλεκτρονική ψηφοφορία, από την άλλη πλευρά, είναι ένα άλλο δημοφιλές, αλλά κρίσιμο, θέμα που σχετίζεται με τις διαδικτυακές υπηρεσίες. Το Ethereum Blockchain με τα έξυπνα συμβόλαια, εμφανίζεται ως ένας καλός υποψήφιος για επίλυση των ζητημάτων ασφαλείας και διαφάνειας που προκύπτουν στις κοινές πλατφόρμες ηλεκτρονικής ψηφοφορίας. Για τις ανάγκες της παρούσας διπλωματικής, αναπτύχθηκε μια αποκεντρωμένη διαδικτυακή εφαρμογή ηλεκτρονικής ψηφοφορίας βασισμένη στο Ethereum.

Σε αυτήν την εργασία γίνεται εκτενής αναφορά στα θεωρητικά και πρακτικά θεμέλια τα οποία χρειάζονται για την ανάπτυξη της προαναφερθείσας εφαρμογής. Έπειτα παρουσιάζεται η λειτουργικότητα της εφαρμογής και γίνεται μια συζήτηση σχετικά με το κατά πόσο αξιόπιστο και αποτελεσματικό είναι αυτό το σύστημα ηλεκτρονικής ψηφοφορίας. Τέλος, δίνονται κάποιες μελλοντικές προτάσεις που θα βελτιώσουν σε θέματα ασφάλειας και όχι μόνο, το πρωτόκολλο ηλεκτρονικής ψηφοφορίας που προτείνουμε.

Λέξεις Κλειδιά: Blockchain, Δίκτυο ομότιμων κόμβων P2P, Κρυπτογραφία, Αποκεντρωμένες εφαρμογές, Έξυπνα Συμβόλαια, Ethereum, Ηλεκτρονική ψηφοφορία

Abstract

Nowadays the Internet combined with the ever-evolving digital technology, has created a huge market for knowledge / information. The invention of the Internet has led to a revolution in the way everyone perceives the world around them. A common feature that is observed in its entirety (web applications, services) is the client-server architecture. This architecture results in a gathering of knowledge that is directly related to the Internet. The big question that arises is: "who rules, who controls this knowledge". In an effort to get rid of this centralized collection of information, there has been a tendency to decentralize Internet services. The technology that is the main force in this direction is none other than Blockchain.

The revolutionary idea of Blockchain, which is the key technology behind the famous Bitcoin cryptocurrency, has sparked the beginning of a new era in the Internet. Blockchain is a distributed ledger (transaction log) for recording transactions in an unchangeable way, maintained by a network of peer-to-peer P2P nodes (computers), who do not completely trust each other. The input data from the moment they enter the chain can not be changed. This is achieved through strong cryptographic principles such as digital signatures, hash functions and Merkle Trees.

While most people today associate Blockchain with cryptocurrencies, it is actually something much bigger. Blockchain is now the cornerstone of decentralized applications, that is, applications based on a network of peer-to-peer nodes rather than the client-server architecture. One of the most widespread and powerful platforms to develop decentralized applications through smart contracts is Ethereum Blockchain.

Electronic voting, on the other hand, is another popular but crucial issue related to online services. Ethereum Blockchain, with its smart contracts, is emerging as a good candidate for resolving security and transparency issues that arise on common e-voting platforms. For the purposes of this diploma thesis, we developed a decentralized online e-voting application based on Ethereum.

This paper makes extensive reference to the theoretical and practical foundations needed to develop the aforementioned application. Then the functionality of the application is presented and there is a discussion about how reliable and effective this e-voting system is. Finally, some future suggestions are given that will improve in terms of security and not only, the electronic voting protocol that we propose.

Keywords: Blockchain, Peer-to-peer P2P network, Cryptography, Decentralized Applications, Smart Contracts, Ethereum, E-voting

Ευχαριστίες

Με αφορμή την ολοκλήρωση της διπλωματικής μου εργασίας και του πρώτου κύκλου σπουδών μου θα ήθελα να εκφράσω θερμές ευχαριστίες πρωτίστως στους γονείς μου, που είναι πάντα υποστηρικτές όλων των μέχρι στιγμής προσπαθειών μου, αλλά είμαι σίγουρος και των μελλοντικών. Χωρίς την αγάπη, την καθοδήγηση και την υπομονή τους σίγουρα δε θα ήμουν ο άνθρωπος που είμαι σήμερα.

Νιώθω επίσης το χρέος να ευχαριστήσω όλους του καθηγητές του Εθνικού Μετσόβιου Πολυτεχνείου για τις γνώσεις που μου μετέδωσαν και ιδιαιτέρως τον επιβλέποντα καθηγητή μου Πέτρο Στεφανέα για την εμπιστοσύνη που μου έδειξε, τον χρόνο που αφιέρωσε για μένα αλλά και την δυνατότητα που μου έδωσε να ασχοληθώ με ένα τόσο πολλά υποσχόμενο κλάδο όπως είναι αυτός του Blockchain.

Τέλος, θα ήθελα να ευχαριστήσω όλους τους κοντινούς μου ανθρώπους, φίλους και συμφοιτητές, που μου παρείχαν στήριξη σε όλη τη διάρκεια των σπουδών μου. Ιδιαίτερα όμως, οφείλω τεράστια ευγνωμοσύνη στην Θεοδώρα για την αλληλοϋποστήριξη και αλληλοβοήθεια που είχαμε σε όλο αυτό το μοναδικό ταξίδι σπουδών στο ΕΜΠ και κυρίως για τις υπέροχες στιγμές που περάσαμε μαζί.

Στέφανος Μαυροζούμης

Αθήνα, Φεβρουάριος 2021

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

1.Εισαγωγή

Είναι πλέον κατανοητό απ' την πλειοψηφία των ανθρώπων, ακόμη και για αυτούς που δεν έχουν σχέση με την Επιστήμη της Πληροφορίας και των Υπολογιστών, ότι η προάσπιση των προσωπικών τους δεδομένων θα πρέπει είναι εξασφαλίσιμη, αδιαμφισβήτητη αλλά και απαραίτητη. Είναι άραγε το διαδίκτυο με τη σημερινή του μορφή απόλυτα ασφαλές για τους χρήστες του; Η απάντηση στο ερώτημα είναι σαφέστατα όχι. Είναι, λοιπόν, καθοριστικό πλέον λόγω των καταιγιστικών εξελίξεων της κοινωνίας αλλά και της τεχνολογίας, να εξασφαλιστεί η εύρεση και υλοποίηση καινοτόμων ιδεών στον χώρο της κρυπτογραφίας.

Προς αυτήν την κατεύθυνση κινείται η επαναστατική ιδέα του Blockchain, η οποία αν και ξεκίνησε αποκλειστικά για το χρηματοοικονομικό σύστημα και τις οικονομικές συναλλαγές , πλέον αρχίζει να βρίσκει εφαρμογή και σε πληθώρα άλλων περιοχών.

Ως ιδέα ,το Blockchain είναι μια αναμφισβήτητα έξυπνη εφεύρεση – το πνευματικό τέκνο ενός ατόμου ή μιας ομάδας ατόμων που είναι γνωστή με το ψευδώνυμο Satoshi Nakamoto, ως μέρος μιας πρότασης για το Bitcoin που ήρθε στο φως το 2008 με την έκδοση του γνωστού White Paper : "Bitcoin: A Peer to Peer Electronic Cash System". Στόχος της πρότασης αυτής ήταν η δημιουργία P2P (peer-to-peer) χρημάτων χωρίς την ανάγκη ύπαρξης τραπεζών. Έτσι ήρθε στο φως το πρώτο ψηφιακό νόμισμα (κρυπτονόμισμα): το Bitcoin. Οι τεχνικές μέθοδοι που χρησιμοποιεί του δίνουν σαφέστατα πλεονεκτήματα , αφού γίνεται σε αυτές χρήση κρυπτογραφικών αρχών. Η παραγωγή του όπως είπαμε δεν εξασφαλίζεται από την ύπαρξη κεντρικών τραπεζών, αλλά από ένα δίκτυο ομότιμων κόμβων (υπολογιστών) που το χρησιμοποιούν. Από τότε όμως το Blockchain έχει εξελιχθεί σε κάτι μεγαλύτερο.

Είναι αδιαμφισβήτητη η χρησιμότητα και η προοπτική εξέλιξης που μπορεί να έχουν τα κρυπτονομίσματα. Την σημερινή ημέρα όμως, που στο Blockchain επενδύονται δισεκατομμύρια από τεράστιες οικονομικές βιομηχανίες και η εξέλιξη του είναι ραγδαία , τα ψηφιακά νομίσματα δεν παύουν να είναι παρά μία από τις πολλές εφαρμογές της τεχνολογίας αυτής. Ιατρική, εκπαίδευση, ψυχαγωγία, αγοροπωλησίες, ασφαλιστική ικανότητα, ακόμη και ηλεκτρονικές ψηφοφορίες μπορούν πλέον να επωφεληθούν της ασφάλειας που προσφέρει η νέα αυτή τεχνολογία.

Αν και είναι σχεδόν απίθανο να προβλεφθεί η μελλοντική εξέλιξη και ο βαθμός υιοθέτησης της νέας αυτής τεχνολογίας, είναι πολύ πιθανό η πορεία της να είναι όμοια με άλλες καινοτόμες προτάσεις του πρόσφατου διαδικτυακού παρελθόντος και τελικά, είτε να υιοθετηθεί από κεντρικές αρχές ή οργανισμούς ή να αποτελέσει τεχνολογικό οδηγό για την βελτίωση των υφιστάμενων υπηρεσιών και τεχνολογιών στο μέτρο που η τεχνολογία Blockchain προσφέρει πλεονεκτήματα έναντί τους.

1.1 Σκοπός της εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας αποκεντρωμένης εφαρμογής (decentralized application ή αλλιώς Dapp) ηλεκτρονικής ψηφοφορίας για τις προεδρικές εκλογές των ΗΠΑ που έλαβαν χώρα τον Νοέμβριο του 2020. Η διαδικτυακή αυτή εφαρμογή, η οποία δίνει τη δυνατότητα στους ψηφοφόρους να επιλέξουν ανάμεσα στους δύο υποψήφιους προέδρους, Ντόναλντ Τράμπ και Τζο Μπάιντεν, χρησιμοποιεί το Ethereum Blockchain ώστε να διασφαλίσει τη διαφάνεια για κάθε ψήφο ξεχωριστά κατά τη διάρκεια όλης της εκλογικής διαδικασίας.

Αν θέλαμε να ορίσουμε την ηλεκτρονική ψηφοφορία (e-voting) θα λέγαμε ότι είναι μια ψηφοφορία στην οποία επιβάλλεται η χρήση ηλεκτρονικών μέσων για την υποβολή της ψήφου και για μία τουλάχιστον από τις ακόλουθες δύο διαδικασίες:

- Ταυτοποίηση των ψηφοφόρων
- Καταμέτρηση των ψήφων

Στα οφέλη που προσφέρει η ηλεκτρονική ψηφοφορία θα μπορούσαμε να εντάξουμε τα παρακάτω:

i. Αύξηση της συμμετοχής στην εκλογική διαδικασία των κοινωνικών ομάδων που αντιμετωπίζουν σημαντικά φυσικά εμπόδια προσέλευσης στη κάλπη.

ii. Μεγαλύτερη αποτελεσματικότητα στην προετοιμασία για τις εκλογές και τον υπολογισμό του τελικού αποτελέσματος.

iii. Μείωση του οικονομικού κόστος των εκλογών (μακροπρόθεσμα).

Γενικότερα οι κοινές ηλεκτρονικές πλατφόρμες ψηφοφορίας που αναπτύσσονται τις μέρες μας δεν χαίρουν ιδιαίτερης αποδοχής από το ευρύ κοινό , καθώς:

i. Κανείς δεν θέλει να θέσει σε κίνδυνο τα προσωπικά του στοιχεία.

ii. Προκύπτει εύλογα η εξής απορία στον καθένα: Τι εμποδίζει τις κυβερνήσεις να χειραγωγήσουν τις ψήφους διαδικτυακά;

iii. Ένα απλό χακάρισμα θα μπορούσε να αλλοιώσει χιλιάδες ψήφους και εσύ να μη μάθεις ποτέ ότι το σύστημα δέχθηκε μια τέτοια επίθεση.

Όλα αυτά πράγματι θα μπορούσαν να καθορίσουν αρνητικά το μέλλον μιας ολόκληρης χώρας.

Η σύγχρονη κρυπτογραφία παρέχει μια λεπτομερή μεθοδολογία για το σχεδιασμό και την επίσημη καθιέρωση της ασφάλειας στα συστήματα ψηφοφορίας. Τη λύση σε αυτά τα ζητήματα που αναφέρθηκαν παραπάνω προτείνει να δώσει το Blockchain και ο αποκεντροποιημένος χαρακτήρας που έχει. Μέσω της τεχνολογίας αυτής μπορούμε να δημιουργήσουμε ένα σύστημα εκλογικής διαδικασίας σε πραγματικό χρόνο, στο οποίο θα μπορεί να εισέλθει και να ψηφίσει καθένας ξεχωριστά χωρίς να χρειαστεί να φτάσει σε κάποιο εκλογικό κέντρο. Επίσης η ασφάλεια του συστήματος μέσω της ισχυρής κρυπτογραφίας που χρησιμοποιεί το κάνει σχεδόν άτρωτο σε κάθε τύπου κυβερνοεπίθεση (Cyberattack) και παράλληλα εξασφαλίζει τη προστασία της ταυτότητας όλων των ψηφοφόρων.

Στην παρούσα διπλωματική λοιπόν, θα δούμε τον τρόπο δημιουργίας ενός Dapp, τόσο από τη πλευρά της επικοινωνίας με το Ethereum Blockchain μέσω έξυπνων συμβολαίων (smart contracts), που αποτελεί το backend της εφαρμογής, όσο και από τη σκοπιά της επικοινωνίας του χρήστη με το web application, το οποίο αποτελεί το frontend της εφαρμογής. Επιπλέον, θα μελετήσουμε τα διάφορα εργαλεία και

γλώσσες προγραμματισμού, οι οποίες καθιστούν δυνατή τη δημιουργία ενός τέτοιου Dapp.

1.2 Δομή περιεχομένου

Η παρούσα διπλωματική εργασία αποτελείται από 7 Κεφάλαια και 2 Παραρτήματα.

Αρχικά στο παρόν Κεφάλαιο, που αποτελεί και το πρώτο Κεφάλαιο, γίνεται η εισαγωγή στη διπλωματική εργασία, δηλαδή στο σκοπό και στη δομή της.

Στο δεύτερο Κεφάλαιο αναλύεται η τεχνολογία του Blockchain. Πιο συγκεκριμένα, γίνεται μια εισαγωγή σε αυτή τη τεχνολογία μέσω της αποσαφήνισης του ορισμού της, μιας ιστορικής αναδρομής, μιας παρουσίασης ενός γνωστού προβλήματος που συνδέεται με αυτή και της περιγραφής των αλγορίθμων συναίνεσης που μπορούν να χρησιμοποιηθούν στο Blockchain. Έπειτα, γίνεται αναφορά στις κρυπτογραφικές αρχές που χρησιμοποιεί ένα δίκτυο Blockchain για την ασφάλεια μέσα σε αυτό. Αναλύεται η κρυπτογραφία ελλειπτικών καμπυλών, που χρησιμοποιεί κατά κύριο λόγο το Ethereum Blockchain, οι συναρτήσεις κατακερματισμού και τα Merkle Trees. Επίσης παρουσιάζονται τα δίκτυα ομότιμων κόμβων (P2P) που αποτελούν ουσιαστικό στοιχείο του Blockchain.

Στο τρίτο Κεφάλαιο γίνεται μια αναλυτική παρουσίαση του Ethereum Blockchain, που αποτελεί την βασική τεχνολογία στην οποία υλοποιήθηκε η εφαρμογή της παρούσας διπλωματικής εργασίας. Μεταξύ άλλων, στο Κεφάλαιο αυτό γίνεται αναφορά στην νεοσύστατη μορφή του Διαδικτύου Web3, στα έξυπνα συμβόλαια (smart contracts) και στις αποκεντρωμένες εφαρμογές (Dapps).

Στο τέταρτο Κεφάλαιο παρουσιάζονται οι σημαντικότερες τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της αποκεντρωμένης εφαρμογής η οποία υλοποιήθηκε στα πλαίσια της διπλωματικής εργασίας.

Στο πέμπτο Κεφάλαιο παρουσιάζεται αναλυτικά ο σχεδιασμός και η υλοποίηση της εφαρμογής. Πιο συγκεκριμένα, αναλύουμε το έξυπνο συμβόλαιο που αναπτύξαμε ώστε να επιτευχθεί το τελικό αποτέλεσμα. Επίσης, μέσω της JavaScript περιγράφονται οι διαδικασίες που υλοποιούνται στην web app μας και γίνονται διάφορες παρατηρήσεις επί της εφαρμογής.

Στο έκτο Κεφάλαιο επιδεικνύεται η λειτουργικότητα της αποκεντρωμένης εφαρμογής μας.

Στο έβδομο Κεφάλαιο συνοψίζεται η εργασία και δίνονται κατευθύνσεις για μελλοντικές βελτιώσεις πάνω στο πρωτόκολλο που προτείνεται.

Στο Παράρτημα Ι περιέχονται αναλυτικές οδηγίες για την χρήση της εφαρμογής. Εδώ υπάρχει επίσης και ένας υπερσύνδεσμος για ολόκληρο τον κώδικα της εφαρμογής, καθώς δεν ήταν δυνατόν να τον παραθέσουμε ολόκληρο στο κείμενο της διπλωματικής.

Τέλος, στο Παράρτημα ΙΙ παραθέτουμε τον κώδικα της ιστοσελίδας της εφαρμογής.

2.Blockchain

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο 2.1 θα κάνουμε μια εισαγωγή στην τεχνολογία του Blockchain. Θα αποσαφηνίσουμε τον ορισμό του, θα κάνουμε μια ιστορική αναδρομή, θα παρουσιάσουμε ένα πρόβλημα από την επιστήμη των υπολογιστών,γνωστό ως "Το πρόβλημα των Βυζαντινών Στρατηγών", το οποίο είναι άρρηκτα συνδεδεμένο με την τεχνολογία του Blockchain, και θα αναλύσουμε δύο βασικούς αλγόριθμους συναίνεσης που μπορούν να χρησιμοποιηθούν σε ένα δίκτυο Blockchain.

2.1.1 Τι είναι το Blockchain

Ένα Blockchain είναι, με απλά λόγια, μια χρονομετρημένη σειρά αμετάβλητων αρχείων δεδομένων που διαχειρίζεται ένα σύμπλεγμα υπολογιστών που δεν ανήκουν σε καμία οντότητα. Κάθε ένα από αυτά τα μπλοκ δεδομένων ασφαλίζεται και συνδέεται το ένα με το άλλο χρησιμοποιώντας κρυπτογραφικές αρχές και έτσι σχηματίζουν μια αλυσίδα (chain).

Όταν μιλάμε για ένα σύμπλεγμα υπολογιστών εννοούμε ένα αποκεντρωμένο δίκτυο ομότιμων κόμβων (peer-to-peer network) όπου όλοι ανά πάσα στιγμή μπορούν να έρθουν σε ομοφωνία μεταξύ τους χωρίς την παρέμβαση κάποιας κεντρικής αρχής και χωρίς να χρειαστεί καν να έρθουν σε επικοινωνία μεταξύ τους. Θα δούμε σε επόμενο κεφάλαιο τι ακριβώς συμβαίνει με αυτά τα δίκτυα.

Το δίκτυο Blockchain, όπως αναφέραμε, δεν έχει κεντρική εξουσία. Αυτό από μόνο του το συνδέει άμεσα με ένα εκδημοκρατισμένο σύστημα. Δεδομένου ότι είναι ένα κοινόχρηστο και αμετάβλητο καθολικό συναλλαγών, οι πληροφορίες σε αυτό είναι ανοιχτές για να τις δουν όλοι. Ως εκ τούτου, οτιδήποτε είναι χτισμένο στο Blockchain είναι από τη φύση του διαφανές και όλοι οι εμπλεκόμενοι είναι υπεύθυνοι για τις πράξεις τους.

Το Blockchain, λοιπόν, είναι ένα κατανεμημένο ledger (κατάστιχο συναλλαγών) για την καταγραφή συναλλαγών, που διατηρείται από πολλούς κόμβους (υπολογιστές), οι οποίοι δεν εμπιστεύονται απόλυτα ο ένας τον άλλο. Αυτή η έλλειψη εμπιστοσύνης "κερδίζεται" μέσω ενός κατανεμημένου κρυπτογραφικού πρωτοκόλλου. Στο Blockchain δεν υπάρχει κεντρική αρχή. Οι κόμβοι επικυρώνουν τις πληροφορίες που είναι υποψήφιες προς προσθήκη στο Blockchain και ένα consensus protocol (πρωτόκολλο συναίνεσης) διασφαλίζει πως όλοι οι κόμβοι συμφωνούν σε μια μοναδική σειρά στην οποία θα προστεθούν οι πληροφορίες.

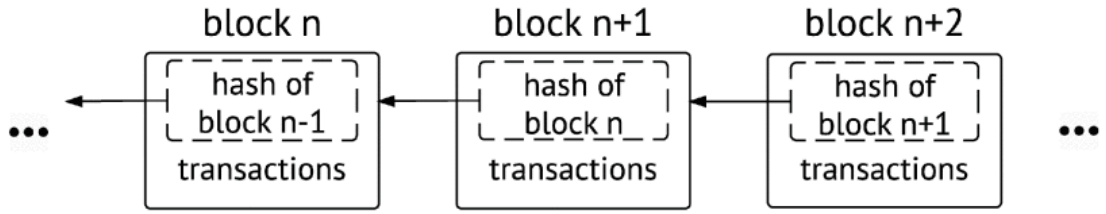
Στο δίκτυο του Blockchain όλοι οι συμμετέχοντες είναι ισότιμοι μεταξύ τους. Αυτό σημαίνει πώς κανένα πρόσωπο μέσα στο δίκτυο δεν υπερέχει έναντι κάποιου άλλου και εκλείπει κάθε είδους προτεραιότητα. Όλα τα πρόσωπα που συμμετέχουν στο δίκτυο, έχουν πρόσβαση σε αυτό κρατώντας ο καθένας ένα αντίγραφο του αρχείου καταχωρήσεων (copy of ledger), κάτι που εξασφαλίζει την ασφάλεια και τη διαφάνεια των συναλλαγών. Κάθε φορά που πρόκειται να εγκριθεί κάποια συναλλαγή στο δίκτυο , ενεργοποιείται το πρωτόκολλο συναίνεσης που αναφέραμε πριν, και σύμφωνα με αυτό ενημερώνεται το αντίγραφο του αρχείου καταχωρήσεων όλων των εμπλεκόμενων στο δίκτυο. Το αρχείο αυτό που δημιουργείται και μοιράζεται σε ένα

μεγάλο αριθμό χρηστών, είναι κρυπτογραφημένο και δεν επιτρέπει την αλλαγή στις εγγραφές που έχουν ήδη περαστεί σε αυτό. Ένα απλό καλό παράδειγμα που μοιάζει με τον τρόπο που δουλεύει το Blockchain είναι το Google Docs. Δύο χρήστες, έχουν πρόσβαση στο ίδιο αρχείο και βλέπουν ακριβώς την ίδια έκδοση, τη ίδια στιγμή, και οποιεσδήποτε αλλαγές κάνει ο κάθε ένας. Φανταστείτε τώρα ένα μεγαλύτερο αριθμό αρχείων τα οποία θα φυλάσσονται κάπως έτσι. Αντί να τα στέλνει ο ένας στον άλλο και να μην μπορεί να διατηρηθεί η ενημερωμένη έκδοση του αρχείου, ούτε να συγχρονίζει με τις υπόλοιπες συσκευές, μέσω αυτής της τεχνολογίας να μπορεί ο κάθε χρήστης να δει την τελευταία ενημέρωση. Όλοι οι συμβαλλόμενοι θα μπορούν να βλέπουν την κάθε αλλαγή και να εμπλακούν σε αυτή.

Ας επιστρέψουμε στον ορισμό του Blockchain και ας εμβαθύνουμε λίγο περισσότερο.

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

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



Σχήμα 2.1 - Παράδειγμα Blockchain

Με αυτόν τον τρόπο δημιουργείται μια αλυσίδα από τα υφιστάμενα block, τα οποία είναι συνδεδεμένα και διατεταγμένα. Σε αυτή την αλυσίδα προφανώς θα υπάρχει το πρώτο block (genesis block ή αλλιώς Block 0) το οποίο δεν θα έχει το hashing του προηγούμενου block όπως όλα τα υπόλοιπα. Κάθε κόμβος του δικτύου περιέχει ένα αντίγραφο του Blockchain, ενώ όταν υπάρχει consensus ανάμεσα στους χρήστες σχετικά με το επόμενο block, αυτό προστίθεται στην αλυσίδα και όλα τα αντίγραφα ενημερώνονται και επικυρώνονται.

Υπάρχουν δύο ειδών Blockchain ως προς την δημοσιότητά τους:

- **Δημόσιο Blockchain**: Οποιοσδήποτε στον κόσμο μπορεί να διαβάσει, να κατεβάσει, να μεταδώσει τις συναλλαγές του Blockchain.
- **Ιδιωτικό Blockchain**: Το Blockchain ανήκει μόνο σε ένα άτομο, σε μια κυβέρνηση ή σε έναν οργανισμό που δεν είναι δημόσιος.

Το Bitcoin και το Ethereum είναι φυσικά δημόσια Blockchain.

2.1.2 Ιστορική Αναδρομή

Η ιδέα πίσω από την τεχνολογία του Blockchain περιγράφηκε για πρώτη φορά το 1991, όταν οι ερευνητές Stuart Haber και W. Scott Stornetta εισήγαγαν μια υπολογιστικά πρακτική λύση για χρονική "σφράγιση" ψηφιακών εγγράφων, έτσι ώστε να μην μπορούν να παραποιηθούν ή να επικαιροποιηθούν ξανά. Το σύστημα χρησιμοποίησε μια κρυπτογραφημένη ασφαλή αλυσίδα από blocks για να αποθηκεύσει τα χρονικά σφραγισμένα έγγραφα. Έπειτα το 1992 τα Merkle trees ενσωματώθηκαν στο σχεδιασμό, καθιστώντας τον πιο αποτελεσματικό επιτρέποντας τη συλλογή πολλών εγγράφων σε ένα μπλοκ. Ωστόσο, αυτή η τεχνολογία δεν χρησιμοποιήθηκε και όλοι αυτοί οι σχεδιασμοί σταμάτησαν το 2004, τέσσερα χρόνια πριν από την εμφάνιση του Bitcoin.

Και κάπως έτσι οδηγούμαστε στο 2008 και στην δημιοσίευση μιας εργασίας (white paper) με το όνομα: "Bitcoin: A Peer to Peer Electronic Cash System" από μία ομάδα ανθρώπων ή μπορεί ακόμα και από ένα άτομο, πίσω απ' το ψευδώνυμο Satoshi Nakamoto. Σε αυτή την εργασία παρουσιάστηκε μια «καθαρά peer-to-peer έκδοση ηλεκτρονικών μετρητών» γνωστή ως Bitcoin και έτσι η τεχνολογία Blockchain έκανε το δημόσιο ντεμπούτο της.

Το Bitcoin με την έλευση του ήταν αυτό που έδωσε λύση στο πρόβλημα των Βυζαντινών Στρατηγών, καθώς καταφέρνει να επιτευχθεί συναίνεση ανάμεσα σε πολλά μέλη, τα οποία όμως δεν εμπιστεύονται το ένα το άλλο, μέσω ανταλλαγής μηνυμάτων, χωρίς να είναι γνωστές οι προθέσεις του κάθε αποστολέα. Στην περίπτωση των ψηφιακών νομισμάτων, το υπολογιστικό πρόβλημα σχετίζεται με το πρόβλημα της διπλής σπατάλης (double spending problem), το οποίο αντιμετωπίζει το πως μπορούμε να επιβεβαιώσουμε ότι μία μονάδα του ψηφιακού νομίσματος έχει χρησιμοποιηθεί για μία μόνο συναλλαγή και όχι για παραπάνω, χωρίς την επιβεβαίωση μιας αξιόπιστης κεντρικής αρχής, όπως για παράδειγμα μίας τράπεζας, η οποία κρατάει αρχείο με όλες τις συναλλαγές και τα υπόλοιπα των λογαριασμών.

Το Blockchain, η τεχνολογία που χρησιμοποιεί το Bitcoin, έχει εξελιχθεί την τελευταία δεκαετία σε μια από τις μεγαλύτερες πρωτοποριακές τεχνολογίες του σήμερα με δυνατότητα να επηρεάσει κάθε κλάδο, από τον οικονομικό έως τον ιατρικό ακόμη και την εκπαίδευση.

Συνεχίζουμε χρονικά τη πορεία του Blockchain και φτάνουμε στο 2013 όπου εκεί ξεκινάει να γίνεται αντιληπτό ότι πλέον το Bitcoin και το Blockchain παύουν να είναι μια ολότητα και αρχίζει η διάσπαση τους. Τότε ένας φιλόδοξος νέος, ο Vitalik Buterin, προγραμματιστής και συνιδρυτής του περιοδικού Bitcoin, δήλωσε ότι το Bitcoin χρειάζεται μια δική του γλώσσα προγραμματισμού για τη δημιουργία αποκεντρωμένων εφαρμογών. Αποτυγχάνοντας να βρει απήχηση η άποψη του στην κοινότητα, ο Vitalik ξεκίνησε την ανάπτυξη μιας νέας κατανεμημένης υπολογιστικής πλατφόρμας που βασίζεται στο Blockchain, το Ethereum. Η μεγαλύτερη διαφορά μεταξύ των Bitcoin και Ethereum, είναι ότι το δεύτερο μπορεί να προγραμματίσει και να σχεδιάσει αποκεντρωμένες εφαρμογές (Dapps) μέσω "έξυπνων συμβολαίων", και όχι απλά να εκδίδει κρυπτονομίσματα. Το Ethereum πέραν του αυτόνομου νομίσματός του, το ETH, μπορεί να χρησιμοποιηθεί για τη δημιουργία "έξυπνων συμβολαίων". Περιληπτικά θα πούμε εδώ ότι είναι κομμάτια κώδικα τα οποία ενεργοποιούνται όταν ικανοποιηθεί ένα σύνολο κριτηρίων στο Blockchain του Ethereum. Θα τα μελετήσουμε λεπτομερώς σε επόμενο κεφάλαιο.

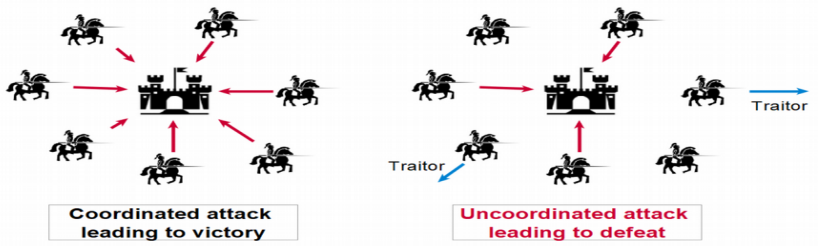
2.1.3 Το πρόβλημα των Βυζαντινών Στρατηγών

Όπως αναφέραμε πριν, το Bitcoin με την έλευση του έδωσε λύση στο πρόβλημα των Βυζαντινών Στρατηγών και πιο συγκεκριμένα στο πρόβλημα της διπλής σπατάλης ενός ψηφιακού νομίσματος. Ας δούμε τι ακριβώς συμβαίνει με το γνωστό "*The Byzantine Generals Problem*".

Το σενάριο που περιγράφει το πρόβλημα αφορά δύο αυτοκρατορίες που ο στρατός της μιας αυτοκρατορίας βρίσκεται εντός των τειχών μιας πόλης ενώ τα στρατεύματα της δεύτερης αυτοκρατορίας έχουν περικυκλώσει την πόλη και ελέγχονται από στρατηγούς που περιμένουν να επιτεθούν. Για να μπορέσουν να καταλάβουν την πόλη οι στρατηγοί θα πρέπει να επιτεθούν την ίδια χρονική στιγμή και όλοι μαζί, αλλιώς θα χάσουν τη μάχη και τον πόλεμο. Ο μόνος τρόπος με τον οποίο μπορούν να επικοινωνήσουν και να συντονίσουν την επίθεσή τους οι στρατηγοί της ίδιας αυτοκρατορίας είναι μέσω κρυφών αγγελιοφόρων που θα περάσουν μέσα από την πόλη και θα μεταδώσουν μηνύματα στους άλλους στρατηγούς. Το πρόβλημα είναι ότι υπάρχει η πιθανότητα κάποιοι από τους αγγελιοφόρους να συλληφθούν, να αποκρυπτογραφηθούν τα μηνύματα που μεταφέρουν και να σταλθούν παραποιημένα στους αποδέκτες. Αλλά ακόμη και να φτάσουν τα μηνύματα επιτυχώς στον προορισμό τους, υπάρχει η πιθανότητα ο στρατηγός-παραλήπτης, για οποιονδήποτε λόγο, να ενεργήσει κακόβουλα και να στείλει πίσω ένα δόλιο μήνυμα για να μπερδέψει τους άλλους στρατηγούς, οδηγώντας σε πλήρη αποτυχία την επίθεση.

Εάν εφαρμόσουμε το πρόβλημα αυτό στο πλαίσιο των Blockchain, κάθε στρατηγός αντιπροσωπεύει έναν κόμβο στο δίκτυο (υπολογιστή) και οι κόμβοι πρέπει να επιτύχουν συναίνεση σχετικά με την τρέχουσα κατάσταση του συστήματος. Με άλλα λόγια, η πλειονότητα των συμμετεχόντων σε ένα κατανεμημένο δίκτυο πρέπει να συμφωνήσουν και να εκτελέσουν την ίδια ενέργεια προκειμένου να αποφευχθεί η πλήρης αποτυχία. Επομένως, ο μόνος τρόπος για να επιτευχθεί συναίνεση σε αυτούς τους τύπους κατανεμημένου συστήματος είναι να έχουμε τουλάχιστον 2/3 ή περισσότερους αξιόπιστους και ειλικρινείς κόμβους δικτύου. Αυτό σημαίνει ότι εάν η πλειονότητα του δικτύου αποφασίσει να δράσει κακόβουλα, το σύστημα είναι ευαίσθητο σε αποτυχίες και επιθέσεις.

Στα πλαίσια του Blockchain υπάρχει ο όρος «Βυζαντινή ανοχή σφάλματος» (Byzantine Fault Tolerance΄ή BFT) που σημαίνει ότι το Blockchain θεωρείται ως ασφαλές όταν οι κακόβουλοι χρήστες έχουν λιγότερη υπολογιστική ισχύ στη διάθεσή τους από ότι οι υπόλοιποι χρήστες και το γεγονός αυτό καθιστά το Blockchain ασφαλές για τη μετάδοση έγκυρων πληροφοριών. Αυτό σημαίνει ότι ένα σύστημα BFT είναι σε θέση να συνεχίσει να λειτουργεί ακόμη και αν ορισμένοι από τους κόμβους αποτύχουν ή ενεργήσουν κακόβουλα.



Σχήμα 2.2 – Το πρόβλημα των Βυζαντινών Στρατηγών

Υπάρχουν πολλές πιθανές λύσεις στο πρόβλημα των Βυζαντινών στρατηγών και, ως εκ τούτου, πολλοί τρόποι οικοδόμησης ενός συστήματος με «Βυζαντινή ανοχή σφάλματος» . Κάπως έτσι, όλες αυτές οι διαφορετικές προσεγγίσεις για ένα

Blockchain με σκοπό την επίτευξη βυζαντινής ανοχής σφαλμάτων μας οδηγεί στους λεγόμενους αλγόριθμους συναίνεσης.

2.1.4 Αλγόριθμοι συναίνεσης

Μπορούμε να ορίσουμε έναν αλγόριθμο συναίνεσης ως τον μηχανισμό μέσω του οποίου ένα δίκτυο Blockchain επιτυγχάνει συναίνεση (consensus). Καθώς σε ένα δίκτυο Blockchain, δεν υπάρχει κεντρική οντότητα με αποτέλεσμα να μην υπάρχει κάποιος να ελέγχει ποιες συναλλαγές θα καταγραφούν στην αλυσίδα, είναι αναγκαία η ύπαρξη ενός αλγορίθμου συναίνεσης με βάση τον οποίο οι συμμετέχοντες θα επαληθεύουν και θα εγκρίνουν συγκεκριμένες πληροφορίες προτού κατοχυρωθούν στο σύστημα. Έτσι επιτυγχάνεται η εμπιστοσύνη μεταξύ των αγνώστων κόμβων του συστήματος και όλα τα δεδομένα συναλλαγών παραμένουν αμετάβλητα. Οι αλγόριθμοι συναίνεσης πρέπει να είναι άτρωτοι σε οποιαδήποτε κακόβουλη ενέργεια. Οι δύο πιο γνωστοί αλγόριθμοι συναίνεσης είναι η απόδειξη εργασίας (Proof of Work) και η απόδειξη συμμετοχής (Proof of Stake). Πριν αναλύσουμε τους δύο αυτούς αλγόριθμους θα πρέπει να αποσαφηνίσουμε τη διαδικασία της εξόρυξης (mining).

Η διαδικασία της εξόρυξης είναι μια υπολογιστική διαδικασία peer-to-peer και χρησιμοποιείται για την ασφάλεια και την επαλήθευση συναλλαγών στο δίκτυο του Blockchain. Το mining περιλαμβάνει τους ανθρακωρύχους (miners) που προσθέτουν δεδομένα συναλλαγών στο παγκόσμιο δημόσιο καθολικό συναλλαγών (ledger) των προηγούμενων συναλλαγών της αλυσίδας. Στα καθολικά αυτά, τα blocks ασφαλίζονται από τους ίδιους τους miners και συνδέονται μεταξύ τους σχηματίζοντας μια αλυσίδα.

Ο miner είναι ένας κόμβος ο οποίος ομαδοποιεί τις συναλλαγές σε block και συναγωνίζεται τους υπόλοιπους miners για το ποιος θα λύσει πιο γρήγορα ένα κρυπτογραφικό "παζλ" προκειμένου να προστεθεί το δικό του μπλοκ στην αλυσίδα. Εφόσον επιλέξει μέσα από μια "πισίνα" ανεξακρίβωτων συναλλαγών (pool of unconfirmed transactions) ποιες συναλλαγές θέλει να προσθέσει στο block, τότε θα πρέπει να παράγει μία μοναδική κατακερματισμένη τιμή (hash) για το block αυτό. Αυτό επιτυγχάνεται όταν ο miner μεταβάλλει μία τιμή nonce που περιέχεται στην επικεφαλίδα του block μέχρι αυτή να γίνει μικρότερη από μία συγκεκριμένη τιμή που λέγεται difficulty target.

Proof of Work

Ο αλγόριθμος Proof of Work προτάθηκε αρχικά το 1993 από δύο ερευνητές, την Cynthia Dwork και τον Moni Naor, ως τρόπος σύνδεσης υπολογιστικού κόστους σε αιτήσεις κατανομής πόρων. Έχει επίσης προταθεί ως μέθοδος ελέγχου των ανεπιθύμητων μηνυμάτων ηλεκτρονικού ταχυδρομείου (spam), ως αποτρεπτικό μέσο για τις επιθέσεις άρνησης υπηρεσίας (denial-of-service attacks), για την επαλήθευση υπολογισμών, για την εφαρμογή καθυστερήσεων όπου κρίνεται απαραίτητο και σε πολλές ακόμα υπηρεσίες.

Έκτοτε η ιδέα αυτή χρησιμοποιήθηκε πρακτικά στο γνωστό White Paper του Bitcoin το 2008 ως λύση στο πρόβλημα της διπλής δαπάνης. Δηλαδή, εάν η Alice έχει μόνο 5 νομίσματα, μπορεί να μεταφέρει αυτά τα 5 νομίσματα στον Bob. Εάν αργότερα η Alice αρνήθηκε τη μεταφορά και αμέσως μετέφερε 5 νομίσματα στον Charly,

προφανώς θα ξόδεψε δύο φορές τα νομίσματά της. Η τρέχουσα κατάσταση του δημόσιου καθολικού συναλλαγών (ledger) που παρακολουθεί αυτά τα νομίσματα θα ήταν εσωτερικά συνεπής, δεδομένου ότι η μεταφορά στον Bob θα ξεχαστεί, αλλά ιστορικά ανακριβής.

Όπως έχουμε πει ξανά, οι συναλλαγές στο Blockchain είναι ουσιαστικά αλλαγές στην τρέχουσα κατάσταση του αποκεντρωμένου καθολικού συναλλαγών που περιγράφει την αλυσίδα. Οι συναλλαγές ομαδοποιούνται σε ένα block. Ο κατακερματισμός (hashing) του προηγούμενου block λειτουργεί ως μοναδικό αναγνωριστικό και προστίθεται στο τρέχων block, το οποίο συνδέει όλα τα blocks μεταξύ τους σχηματίζοντας μια αλυσίδα.

Η εσωτερική συνέπεια των συναλλαγών σε κάθε μπλοκ μπορεί εύκολα να επαληθευτεί από έναν έντιμο πελάτη (honest client). Ωστόσο, αυτός ο πελάτης δεν είναι πάντα δεδομένο ότι θα υπάρχει στο δίκτυο. Για να επιβεβαιωθεί λοιπόν ότι οι συναλλαγές είναι ουσιαστικά αμετάβλητες, η δημιουργία ενός block πρέπει να είναι "δύσκολη" και δαπανηρή. Η εισαγωγή ενός block που έρχεται σε αντίθεση ή αγνοεί μέρος της υπάρχουσας αλυσίδας (είτε κατά λάθος είτε επειδή η Alice έχει κακές προθέσεις) πρέπει να είναι πιο ακριβής από την πιθανή ανταμοιβή που κερδίζεται ακολουθώντας την υπάρχουσα συναίνεση του Blockchain.

Για τον λόγο αυτό οι κόμβοι ενός δικτύου Blockchain εκτελούν τον αλγόριθμο Proof of Work για τη δημιουργία ενός νέου block. Αυτός ο αλγόριθμος απαιτεί από κάθε miner να επιλέξει έναν τυχαίο αριθμό που ονομάζεται nonce, να τον προσθέσει στη λίστα συναλλαγών που σχηματίζουν ένα block, να κατακερματιστεί το μπλοκ (hashing) με τη χρήση του αλγορίθμου SHA-256 και να ελέγξει εάν το προκύπτον hash ξεκινά με έναν καθορισμένο αριθμό μηδενικών. Εάν όχι, επιλέγει έναν νέο τυχαίο αριθμό nonce και προσπαθεί ξανά. Ο επεξεργαστής του μηχανήματος κόμβων, δηλαδή του υπολογιστή του miner, θα παράγει πολλούς κατακερματισμούς έως ότου πετύχει, γεγονός που το καθιστά απευθείας πρόβλημα επιπέδου επεξεργαστή CPU.

Λόγω των ιδιοτήτων των συναρτήσεων κατακερματισμού, κάθε επιλογή αριθμού nonce έχει την ίδια πιθανότητα να έχει ως αποτέλεσμα την απαιτούμενη τιμή. Η δημιουργία ενός block μετατρέπεται σε ανταγωνισμό μεταξύ όλων των κόμβων του δικτύου. Είναι σαν μια λαχειοφόρο αγορά. Οι κόμβοι με μεγαλύτερη πιθανότητα εύρεσης του σωστού nonce είναι αυτοί που μπορούν να δοκιμάσουν περισσότερες nonces ταυτόχρονα. Αυτό είναι και το mining που αναφέραμε προηγουμένως.

Οι κόμβοι (miners) που κατάφεραν να βρουν το σωστό nonce κερδίζουν μια ανταμοιβή, η οποία συνήθως προέρχεται από τις προμήθειες που καταβάλλουν οι αποστολείς των συναλλαγών, το λεγόμενο gas. Όταν ένα έγκυρο block εξορύσσεται, όλοι οι κόμβοι ενημερώνουν την κατάστασή τους για να αποδεχτούν τις συναλλαγές που περιλαμβάνει. Στη συνέχεια συνεχίζουν το mining, προσπαθώντας να σχηματίσουν ένα νέο έγκυρο block από όλες τις συναλλαγές που εκκρεμούν.

Επίσης, λόγω των ιδιοτήτων των συναρτήσεων κατακερματισμού, είναι δυνατό δύο κόμβοι να βρουν ένα έγκυρο nonce την ίδια στιγμή. Εάν συμβεί αυτό, θα υπάρχουν δύο ανταγωνιστικές αλυσίδες με διαφορετικά μονοπάτια. Οι άλλοι κόμβοι στο δίκτυο θα πρέπει να επιλέξουν μία από αυτές πριν συνεχίσουν τις εργασίες εξόρυξης, επειδή το block που προσπαθούν να δημιουργήσουν πρέπει να περιλαμβάνει το κατακερματισμό του block στην άκρη της αλυσίδας. Ως γενικός κανόνας εδώ επιλέγεται η αλυσίδα που είναι σαφώς μεγαλύτερη. Όλοι οι κόμβοι που εξακολουθούν να εργάζονται στη μικρότερη αλυσίδα θα μεταβούν στη μεγαλύτερη αλυσίδα, προσθέτοντας όλα τα blocks που εξορύσσονται στην αλυσίδα τους.

Ο αλγόριθμος Proof of Work βασίζεται στην αρχή ότι κανένας κόμβος στο δίκτυο δε πρέπει να κατέχει περισσότερο από το 50% της συνολικής υπολογιστικής δύναμης, καθώς αυτός θα έχει τη δυνατότητα να ελέγχει αποτελεσματικά το σύστημα (51% επίθεση). Όταν διαχειρίζεται το δίκτυο ένας μεγάλος αριθμός χρηστών, τότε αυτό είναι σχεδόν ακατόρθωτο. Όσο μεγαλύτερο είναι το δίκτυο, τόσο πιο ανθεκτικό είναι σε τέτοιου είδους επιθέσεις.

Τον αλγόριθμο Proof of Work, εκτός του Bitcoin, τον έχει υιοθετήσει και η πλατφόρμα Ethereum, πάνω στην οποία αναπτύσσεται η εφαρμογή της διπλωματικής. Λόγω όμως της τεράστιας ενέργειας (σε υπολογιστική δύναμη) που απαιτεί ο αλγόριθμος Proof of Work για να επαληθευτεί μια συναλλαγή, έχει αρχίσει να γίνεται πλέον πιθανή μια αλλαγή στο πρωτόκολλο συναίνεσης του Ethereum. Είναι δηλαδή πολύ πιθανό τους επόμενους μήνες ο αλγόριθμος Proof of Work που χρησιμοποιεί το Ethereum να αντικατασταθεί από τον αλγόριθμο απόδειξης συμμετοχής ή αλλιώς Proof of Stake.

Proof of Stake

Ο αλγόριθμος Proof of Stake προσπαθεί να λύσει τα προβλήματα ενέργειας που δημιουργεί ο αλγόριθμος Proof of Work. Για να το κάνει αυτό, αντικαθιστά τον ανταγωνισμό του Proof of Work επιλέγοντας τυχαία τους συμμετέχοντες, οι οποίοι θα λάβουν μέρος στη διαδικασία επαλήθευσης συναλλαγών σε ένα block και στην μετέπειτα εισαγωγή του στο Blockchain.

Στον συγκεκριμένο αλγόριθμο οι ενδιαφερόμενοι που θέλουν να συμμετάσχουν στη διαδικασία επικύρωσης, υποχρεούνται να "κλειδώσουν" ένα συγκεκριμένο ποσό νομισμάτων στο δίκτυο ως ποντάρισμα τους. Ένας κάτοχος ποντάρισματος (μέτοχος) ενός δεδομένου Blockchain είναι ένα άτομο που κρατά μερικά εγγενή νομίσματα αυτού του Blockchain, και το ποντάρισμα αναφέρεται στην κατοχή τέτοιων νομισμάτων ενός μετόχου. Για παράδειγμα, στο δίκτυο Ethereum, το ποντάρισμα θα είναι το ποσό ETH που συγκρατείται από έναν κόμβο.

Το ύψος του ποντάρισματος καθορίζει τις πιθανότητες να επιλεγεί ένας κόμβος ως ο επόμενος επικυρωτής, δηλαδή όσο μεγαλύτερο είναι το ποντάρισμα, τόσο μεγαλύτερες είναι οι πιθανότητες. Θεωρητικά, οι επικυρωτές με υψηλότερα ποντάρισματα είναι πιθανότερο να επιλεγούν για την επικύρωση της φήμης τους.

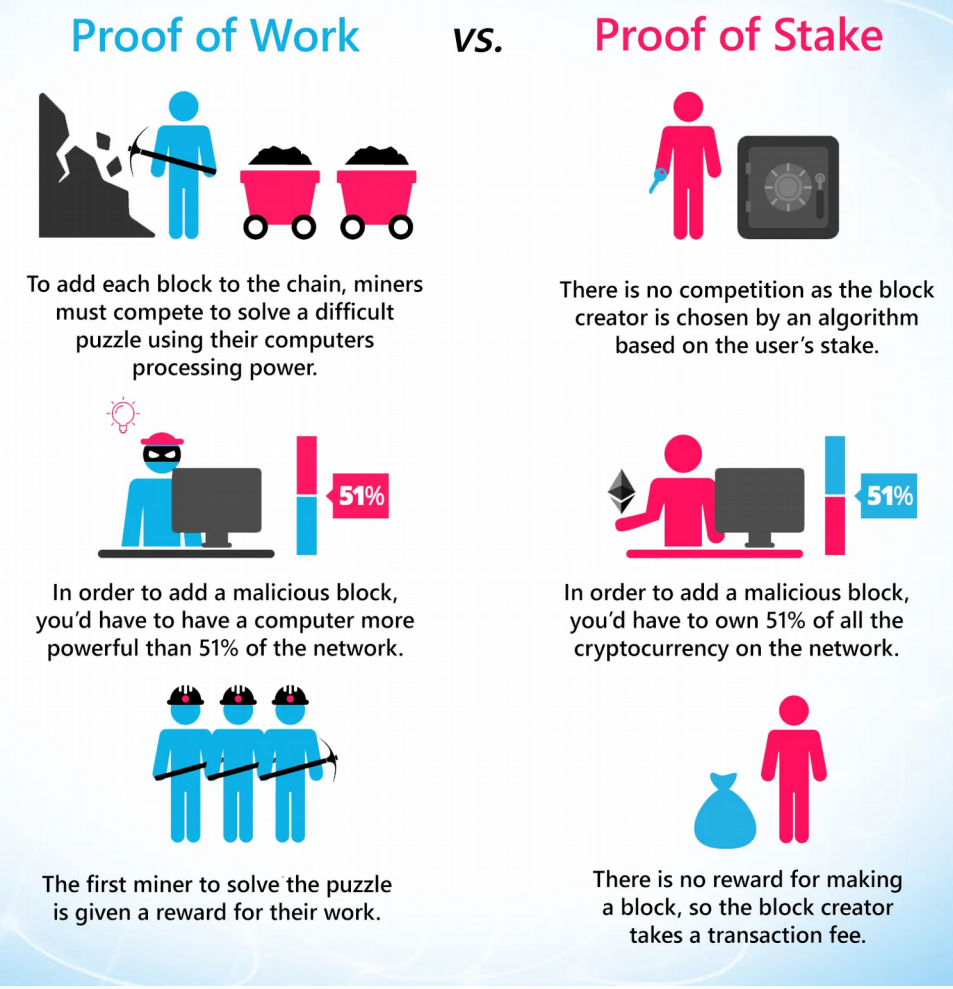
Εάν ένας υποψήφιος επικυρωτής αποφασίσει να ενεργήσει κακόβουλα, το δίκτυο θα έχει έναν μηχανισμό για να τον τιμωρήσει και θα χάσει πολύ περισσότερα χρήματα από αυτούς που κάνουν χαμηλότερα ποντάρισματα. Έτσι, για να διασφαλιστεί ότι οι επικυρωτές δεν ενεργούν κακόβουλα, ορισμένα Blockchain έχουν μια οντότητα που λέγεται fisherman ή στα ελληνικά ψαράς. Αυτός ο ψαράς είναι ουσιαστικά η "αστυνομία" του συστήματος, η οποία παρακολουθεί τους επικυρωτές και ελέγχει ύποπτες συναλλαγές ή επικυρώσεις. Εάν ο κατηγορούμενος επικυρωτής αποδειχθεί κακόβουλος, τιμωρείται από το δίκτυο. Αυτή η τιμωρία ποικίλλει μεταξύ των δικτύων. Μπορεί να έχει ως ποινή τον αποκλεισμό του κόμβου επικύρωσης από το δίκτυο ή την εισαγωγή του σε μαύρη λίστα για την επικύρωση στο μέλλον ή την απώλεια χρημάτων του.

Η επιλογή του κόμβου ως του κατάλληλου για επικυρωτή βασίζεται σε ένα συνδυασμό παραγόντων που περιλαμβάνουν την ηλικία ποντάρισματος (ο αριθμός των ημερών που το χρηματικό ποσό έχει διατεθεί), την τυχαιοποίηση και τον πλούτο (συνολικό ποσό ιδιοκτησίας) του κόμβου. Εάν ένας κόμβος έχει επιλεχτεί ως

επικυρωτής για το επόμενο block, θα ελέγξει εάν οι συναλλαγές στο block είναι έγκυρες, θα υπογράψει το block και θα το προσθέσει στο Blockchain. Σε αντάλλαγμα, ο κόμβος παίρνει ένα φόρο συναλλαγής ως ανταμοιβή για τη δημιουργία του νέου block. Όταν ένας κόμβος αποφασίσει να σταματήσει να είναι επικυρωτής, το στοίχημα του μαζί με τις κερδισμένες ανταμοιβές αποδεσμεύονται μετά από ένα ορισμένο χρονικό διάστημα, δίνοντας στο δίκτυο χρόνο να επαληθεύσει ότι δεν προστέθηκαν ψευδή blocks στο Blockchain από αυτόν τον κόμβο.

Κλείνοντας, θα αναφέρουμε δύο βασικά πλεονεκτήματα τα οποία είναι και οι λόγοι που το Ethereum πρωτόκολλο θέλει να υιοθετήσει την Proof of Stake συναίνεση:

- Εξοικονόμηση ενέργειας
- Ένα ασφαλέστερο δίκτυο, καθώς οι επιθέσεις γίνονται πιο ακριβές: εάν ένας χάκερ θέλει να αγοράσει το 51% του συνολικού αριθμού νομισμάτων, η αγορά αντιδρά με τη γρήγορη ανατίμηση των τιμών.



Σχήμα 2.3 – Proof of Work vs Proof of Stake

2.2 Κρυπτογραφία στο Blockchain

Αν κανείς "κοιτάξει" πίσω στην απαρχή γέννησης της ανθρωπότητας θα βρει στοιχεία που υποδεικνύουν ότι οι άνθρωποι της εποχής αναζητούσαν από τότε τρόπους για διατήρηση της μυστικότητας. Η τέχνη λοιπόν για αναζήτηση της μυστικότητας οδήγησε στην κρυπτογραφία. Η κρυπτογραφία εμφανίστηκε ως έννοια και ιδεολογική προσέγγιση, θα έλεγε κανείς, σχεδόν μαζί την εμφάνιση της γραφής. Αυτό γίνεται και αντιληπτό αν αναλύσει κανείς ετυμολογικά τη λέξη (κρυπτός + γράφω). Ως κρυπτογραφία ορίζουμε πλέον την επιστήμη που ασχολείται με τη μελέτη, την ανάπτυξη και τη χρήση τεχνικών κρυπτογράφησης και αποκρυπτογράφησης με σκοπό την απόκρυψη του περιεχομένου των μηνυμάτων.

Η πρώτη επίσημη καταγραφή χρήσης κρυπτογραφίας τοποθετείται στα 1500-1700 π .Χ. πάνω σε μία πήλινη πινακίδα που βρέθηκε στον ποταμό Τίγρη. Αυτή έφερε πάνω της γραμμένη μία συνταγή γραμμένη σε σφηνοειδή γραφή κάνοντας χρήση δυσνόητων και σπάνιων συμβόλων, κάτι που υποδεικνύει ότι είχε "κρυπτογραφηθεί" ώστε το μυστικό παραγωγής να μη διαρρεύσει. Έκτοτε έχουμε πολλές εμφανίσεις της κρυπτογραφίας στην ιστορία. Αναφορά σε αυτή γίνεται από τον Όμηρο στην Ιλιάδα όπως επίσης και στην Αρχαία Σπάρτη τον 5ο π.Χ. αιώνα με τις περίφημες σκυτάλες.

Η κρυπτογραφία ιστορικά μπορεί να διακριθεί σε τρία στάδια. Στο πρώτο στάδιο οι διαδικασίες κρυπτογράφησης αφορούσαν τον τρόπο της έντυπης απεικόνισης , δηλαδή μολύβι και χαρτί. Εδώ εφευρέθηκαν πολλοί αλγόριθμοι που χρησιμοποιούσαν ως δομικές πράξεις την αντικατάσταση και την μετάθεση χαρακτήρων. Ένας από τους γνωστότερους αλγόριθμους αντικατάστασης είναι ο κρυπταλγόριθμος του Καίσαρα. Ενδεικτικά θα αναφέρουμε εδώ, ότι για την επικοινωνία μεταξύ των λεγεώνων, ο Ιούλιος Καίσαρας χρησιμοποίησε αυτόν τον κρυπταλγόριθμο στον οποίο κάθε γράμμα του μηνύματος αντικαθίσταται από το γράμμα που βρίσκεται τρεις θέσεις πιο μετά στο αλφάβητο. Σαν δεύτερο στάδιο αναφέρεται αυτό των κρυπτογραφικών μηχανών. Εδώ είναι γνωστή η μηχανή κρυπτογράφησης Enigma (1918) η οποία χρησιμοποιήθηκε στον Δεύτερο Παγκόσμιο Πόλεμο από τους Γερμανούς και τους Ιταλούς καθ' όλη τη διάρκεια του πολέμου. Το τεράστιο πλήθος πιθανών κρυπτογραφικών κλειδιών που παράγει αυτή η μηχανή, οδήγησε τον Alan Turing να κατασκευάσει τον πρώτο ηλεκτρονικό υπολογιστή, ώστε να βοηθηθεί η προσπάθεια για το "σπάσιμο" της Enigma. Τρίτο και τελευταίο στάδιο θεωρείται το σύγχρονο κρυπτογραφικό σύστημα, ως απόρροια της αμοιβαίας αλληλεπίδρασης των υπολογιστών και των μαθηματικών. Πιο συγκεκριμένα, οι υπολογιστές επέτρεψαν τη χρήση πολυπλοκότερων συστημάτων κρυπτογράφησης και τα μαθηματικά προσέφεραν τον σχεδιασμό.

Στην σύγχρονη κρυπτογραφία (τέλη Β' Παγκοσμίου και έπειτα) διακρίνονται τρία βασικά κρυπτογραφικά αρχέτυπα στα οποία κατηγοριοποιούνται τα υπάρχοντα κρυπτογραφικά εργαλεία:

- **Τα αρχέτυπα δίχως κλειδί**: Σε αυτά ανήκουν οι συναρτήσεις κατακερματισμού (hash functions) τις οποίες θα τις δούμε αναλυτικά στη συνέχεια. Επίσης στην κατηγορία αυτή ανήκουν και οι μονόδρομες συναρτήσεις και οι τυχαίες συναρτήσεις.
- **Τα αρχέτυπα συμμετρικού ή μυστικού κλειδιού**: Τα αρχέτυπα αυτά διαφοροποιούν την έξοδο τους ανάλογα με το κλειδί που χρησιμοποιείται, δηλαδή, το ίδιο κείμενο κρυπτογραφημένο με διαφορετικό κλειδί δίνει διαφορετική έξοδο. Σε αυτά ανήκουν οι αλγόριθμοι κρυπτογράφησης τμήματος, κρυπτογράφησης

ροής και οι κώδικες αυθεντικοποίησης μηνύματος.

- **Τα αρχέτυπα ασύμμετρου ή δημόσιου κλειδιού**: Και σε αυτά η έξοδος εξαρτάται από το κλειδί, όμως εδώ το κλειδί κρυπτογράφησης διαφέρει από το κλειδί αποκρυπτογράφησης. Σε αυτά ανήκουν οι αλγόριθμοι κρυπτογράφησης δημόσιου κλειδιού. Υπάρχουν τρεις κύριες κατηγορίες αλγορίθμων κρυπτογράφησης δημόσιου κλειδιού:

  i. *Παραγοντοποίησης ακέραιου* : *οι* μεγάλοι ακέραιοι αριθμοί χρησιμοποιούνται για την εκτέλεση λειτουργιών ασύμμετρης κρυπτογραφίας . Ο RSA είναι ο κύριος αλγόριθμος κρυπτογραφίας που περιλαμβάνεται σε αυτήν την κατηγορία.

  ii. *Διακριτού λογαρίθμου* : η modular αριθμητική χρησιμοποιείται για την εκτέλεση λειτουργιών ασύμμετρης κρυπτογραφίας. Η λειτουργία modulo είναι μια μονόδρομη λειτουργία, οπότε είναι πολύ δύσκολο να βρεθεί η είσοδος ξεκινώντας από το αποτέλεσμα. Ο αλγόριθμος Diffie-Hellman είναι το πιο γνωστό παράδειγμα για αυτό το είδος ασύμμετρης κρυπτογραφίας.

  iii. *Ελλειπτικών καμπυλών* : βασίζεται στο διακριτό λογάριθμο αλλά στο πλαίσιο των ελλειπτικών καμπυλών. Η ελλειπτική καμπύλη είναι μια αλγεβρική κυβική καμπύλη (δηλαδή έχει πολυωνυμική εξίσωση $3_{ου}$ βαθμού) πάνω από ένα πεδίο που ορίζεται από μια εξίσωση. Η κρυπτογραφία ελλειπτικής καμπύλης χρειάζεται μικρού μεγέθους κλειδί, σε σύγκριση με το RSA, και προσφέρει το ίδιο επίπεδο ασφαλείας. Παραδείγματα αλγορίθμων που βασίζονται σε ελλειπτικές καμπύλες είναι ο αλγόριθμος ψηφιακής υπογραφής ελλειπτικών καμπυλών (Elliptic Curve Digital Signature Algorithm – ECDSA) και ο αλγόριθμος ανταλλαγής κλειδιού ελλειπτικών καμπυλών Diffie-Hellman (ECDH).

Η κρυπτογράφηση είναι ένα βασικό συστατικό της τεχνολογίας Blockchain. Η ασύμμετρη κρυπτογραφία είναι αυτή που χρησιμοποιείται για την ασφάλεια σε ένα δίκτυο που βασίζεται στο Blockchain. Στην ασύμμετρη κρυπτογραφία ένα ζεύγος κλειδιών χρησιμοποιείται για την εκτέλεση κρυπτογράφησης / αποκρυπτογράφησης. Το κλειδί που χρησιμοποιείται για την κρυπτογράφηση των δεδομένων διαφέρει από το κλειδί που χρησιμοποιείται για την αποκρυπτογράφηση αυτών, όπως αναφέραμε και παραπάνω. Η κρυπτογραφία δημόσιου κλειδιού προσφέρει όλους εκείνους τους μηχανισμούς που είναι απαραίτητοι στο Blockchain για τη δημιουργία κλειδιών (πρωτόκολλα για τη δημιουργία κλειδιών πάνω από ένα μη ασφαλές κανάλι), ψηφιακών υπογραφών και ταυτοποίησης.

Παρακάτω θα αναλύσουμε περαιτέρω την κρυπτογραφία ελλειπτικών καμπυλών (Elliptic Curve Cryptography – ECC) διότι η πλειοψηφία των δικτύων Blockchain (όπως και το Ethereum που αναπτύχθηκε η εφαρμογή της διπλωματικής) βασίζονται σε αυτό το σχήμα κρυπτογράφησης.

2.2.1 Κρυπτογραφία Ελλειπτικών Καμπυλών

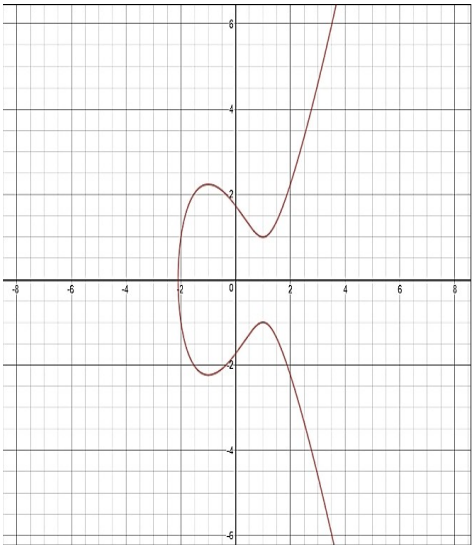
Ένας βασικός παράγοντας που κάνει τις αλυσίδες τύπου Blockchain εξαιρετικά ελκυστικές είναι η ασφάλεια που προσφέρουν κατά την εκτέλεση των συναλλαγών στο δίκτυο του. Το Bitcoin και το Ethereum χρησιμοποιούν κρυπτογραφία ελλειπτικών καμπυλών (Elliptic curve cryptography - ECC) για την υπογραφή των συναλλαγών των χρηστών του.

Η κρυπτογραφία ελλειπτικών καμπυλών (ECC) ανακαλύφθηκε το 1985 από τους Victor Miller (IBM) και Neil Koblitz (Πανεπιστήμιο της Ουάσιγκτον) ως εναλλακτικός μηχανισμός για την εφαρμογή κρυπτογράφησης δημόσιου κλειδιού. Οι αλγόριθμοι δημόσιου κλειδιού, να υπενθυμίσουμε, δημιουργούν έναν μηχανισμό για την κοινή χρήση κλειδιών μεταξύ μεγάλου αριθμού συμμετεχόντων ή οντοτήτων σε ένα σύνθετο σύστημα πληροφοριών. Σε αντίθεση με άλλους δημοφιλείς αλγόριθμους όπως ο RSA, η ECC βασίζεται στον διακριτό λογάριθμο και για αυτό είναι πολύ πιο δύσκολο να παραβιαστεί σε ισοδύναμα μήκη κλειδιού.

Η ελλειπτική καμπύλη βασίζεται στην εξίσωση που δημιουργεί μια καμπύλη πάνω από ένα πεπερασμένο σώμα F. Θεωρούμε ότι το F είναι είτε το ίδιο το σύνολο των πραγματικών αριθμών $\mathbb{R}$, είτε το πεπερασμένο σώμα $GF(q)$ όπου $q = p^r$

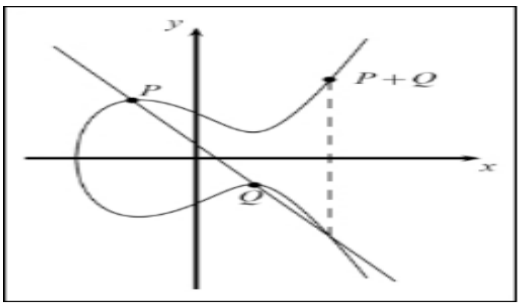
και p πρώτος. Η ελλειπτική καμπύλη ορίζεται ως εξίσωση με την εξής μορφή: $y^2 = x^3 + A \cdot x + B(modp)$,όπου τα A,B ανήκουν στο F και το p είναι πρώτος. Μαζί με τα σημεία (x,y) που ικανοποιούν την παραπάνω εξίσωση εντάσσεται και ένα στοιχείο O, το οποίο ονομάζουμε "σημείο στο άπειρο".

Στο παρακάτω *σχήμα 2.4* βλέπουμε την αναπαράσταση μιας τυχαίας ελλειπτικής καμπύλης πάνω στο σύνολο των πραγματικών αριθμών $\mathbb{R}$: $y^2 = x^3 + A \cdot x + B$, όπου τα x,y,A,B ανήκουν στο $\mathbb{R}$.



Σχήμα 2.4 – Ελλειπτική καμπύλη πάνω στο $\mathbb{R}$

Έστω τώρα μια τυχαία ευθεία που τέμνει την ελλειπτική καμπύλη σε τρία διαφορετικά σημεία P,Q,R. (*σχήμα 2.5*). Ορίζουμε ως άθροισμα των P,Q το παρακάτω: Αν αθροίσουμε τα σημεία P,Q αυτό θα έχει ως αποτέλεσμα ένα καινούργιο σημείο $P+Q=-R$ πάνω στην ελλειπτική καμπύλη,όπου R είναι το τρίτο και τελευταίο σημείο τομής της καμπύλης με την ευθεία. Το άθροισμα των συντεταγμένων των P,Q γίνεται ως εξής: $(x1, y1)+(x2, y2)=(x3, y3)$, όπου το P έχει συντεταγμένες *(x1,y1)* και το Q έχει συντεταγμένες *(x2,y2).*



Σχήμα 2.5 – Πρόσθεση των P,Q

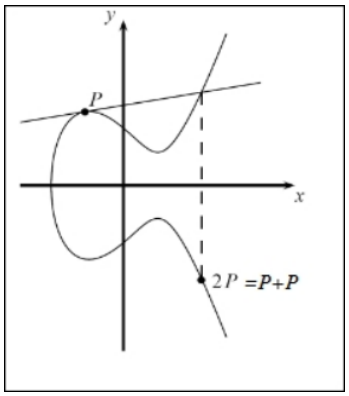
Αν θεωρήσουμε *s* την γραμμή μεταξύ των *P,Q* τότε έπειτα από πράξεις θα προκύψει:

$$s=\frac{(y2-y1)}{(x2-x1)}\,modp$$

Αν θεωρήσουμε εξίσωση ευθείας και κάνουμε πράξεις θα καταλήξουμε στην εξής έκφραση για τις συντεταγμένες του σημείου *P+Q*: $x3=s^2-x1-x2(modp)$

$$y3=s\cdot(x1-x3)-y1(modp)$$

Έστω τώρα μια τυχαία ευθεία που εφάπτεται της ελλειπτικής καμπύλης σε <u>ένα</u> μόνο σημείο *P* και την τέμνει σε ένα σημείο *R*. (*σχήμα 2.6*). Ορίζουμε ως διπλασιασμό του *P* (αντίστοιχο της πρόσθεσης των *P,Q* στη προηγούμενη περίπτωση) το παρακάτω: Αν διπλασιάσουμε το σημείο *P* αυτό θα έχει ως αποτέλεσμα ένα καινούργιο σημείο *P+P=2P=-R* πάνω στην ελλειπτική καμπύλη,όπου *R* είναι το σημείο τομής της καμπύλης με την ευθεία (αφού είναι εφαπτόμενη στο *P*). Ο διπλασιασμός των συντεταγμένων του *P* γίνεται ως εξής: $(x1,y1)+(x1,y1)=2\cdot(x1,y1)=(x2,y2)$, όπου το *P* έχει συντεταγμένες *(x1,y1)*.



Σχήμα 2.6– Διπλασιασμός του P

Αν θεωρήσουμε *s* την κλίση της εφαπτομένης που διέρχεται απ' το *P* τότε έπειτα από πράξεις θα προκύψει: $s=\frac{3\cdot x1^2+A}{2\cdot y1}(modp)$

Αν θεωρήσουμε εξίσωση ευθείας και κάνουμε πράξεις θα καταλήξουμε στην εξής έκφραση για τις συντεταγμένες του σημείου *2P*:

$$x2=s^2-2\cdot x1(modp)$$

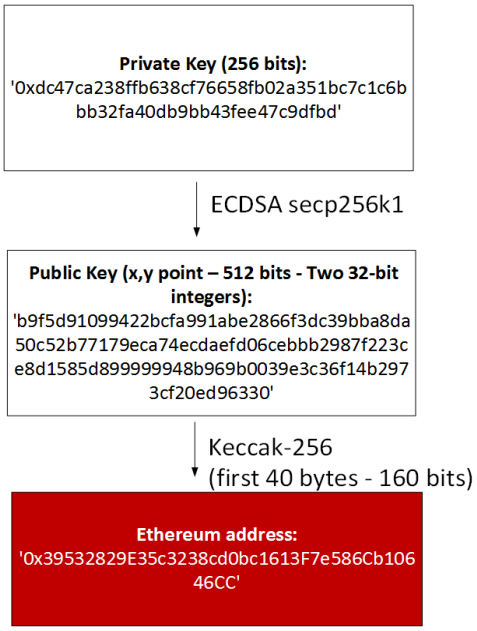
$$y2=s\cdot(x1-x2)-y1(modp)$$

Στην παραπάνω πρόσθεση σημείων που κάναμε θεωρήσαμε το πεπερασμένο σώμα μας Ғ να είναι το ℝ ώστε να γίνει πιο εύκολα αντιληπτός ο γεωμετρικός ορισμός για την πρόσθεση των σημείων της καμπύλης. Υπό ορισμένες συνθήκες όλα τα σημεία σε μια ελλειπτική καμπύλη σχηματίζουν μια κυκλική (αβελιανή) ομάδα. Η απόδειξη αυτής της πρότασης είναι αρκετά δύσκολη και δε θα τη πραγματευτούμε στην παρούσα διπλωματική.

Έτσι, στην κρυπτογραφία ελλειπτικών καμπυλών, το δημόσιο κλειδί είναι ένα τυχαίο πολλαπλάσιο του γεννήτορα της ομάδας που σχηματίζουν τα σημεία της καμπύλης. Το ιδιωτικό κλειδί είναι ένας τυχαία επιλεγμένος ακέραιος που χρησιμοποιείται για τη δημιουργία του πολλαπλάσιου του γεννήτορα. Άρα, το δημόσιο κλειδί είναι ένα σημείο στην ελλειπτική καμπύλη και το ιδιωτικό κλειδί είναι ένας τυχαία επιλεγμένος ακέραιος. Σύμφωνα με αυτή την επιλογή δημόσιου-ιδιωτικού κλειδιού, συμπεραίνουμε ότι είναι σχεδόν αδύνατο να παραβιαστεί ένα σύστημα που χρησιμοποιεί την ECC ως επίπεδο ασφάλειας τους. Κάτι τέτοιο θα μπορούσε να γίνει μόνο με μια ραγδαία ανάπτυξη στην κβαντική υπολογιστική (Quantum Computing), κάτι το οποίο απέχει μάλλον πολύ από την σημερινή εποχή.

Στο Ethereum, χρησιμοποιείται η ECC για να δημιουργηθεί ένα δημόσιο κλειδί και ένα ιδιωτικό κλειδί. Το ιδιωτικό κλειδί διατηρείται ασφαλές και, στη συνέχεια, το δημόσιο κλειδί χρησιμοποιείται για την "παραγωγή" της διεύθυνσης Ethereum του καθενός. Όταν υπογράφουμε μια συναλλαγή, χρησιμοποιούμε το ιδιωτικό μας κλειδί για να δημιουργήσουμε μια υπογραφή, η οποία ελέγχεται αυτόματα στη διεύθυνση Ethereum που μας έχει αποδοθεί. Κατά τη δημιουργία των κλειδιών, δημιουργείται ένα ιδιωτικό κλειδί 256-bit. Επίσης το δημόσιο κλειδί είναι είναι ένα σημείο στην secp256k1 ECDSA καμπύλη (x,y σημείο). Δηλαδή, ένα δημόσιο κλειδί Ethereum είναι δύο αριθμοί ενωμένοι. Αυτοί οι αριθμοί παράγονται από το ιδιωτικό κλειδί με έναν υπολογισμό που είναι μονής κατεύθυνσης. Αυτό σημαίνει ότι είναι ασήμαντο να υπολογίζετε ένα δημόσιο κλειδί εάν έχετε το ιδιωτικό κλειδί και απίθανο να μπορείτε να υπολογίσετε το ιδιωτικό κλειδί από το δημόσιο κλειδί. Το δημόσιο κλειδί υπολογίζεται από το ιδιωτικό κλειδί χρησιμοποιώντας κρυπτογραφία ελλειπτικής καμπύλης, η οποία είναι πρακτικά μη αναστρέψιμη (μονής κατεύθυνσης από το k στο K): **K = k * G**, όπου το k είναι το ιδιωτικό κλειδί, το G είναι ένα σταθερό σημείο που ονομάζεται γεννήτορας, το K είναι το προκύπτον δημόσιο κλειδί και * είναι το ειδικό σύμβολο «πολλαπλασιασμού» ελλειπτικής καμπύλης.

Το δημόσιο κλειδί στη συνέχεια κατακερματίζεται χρησιμοποιώντας την Keccak-256 (γνωστή και ως SHA-3) και τα πρώτα 160 bit γίνονται η δημόσια διεύθυνση Ethereum. Το ιδιωτικό κλειδί πρέπει να είναι εντελώς τυχαίο και υπάρχουν 2^{256} διαφορετικά κλειδιά που μπορούν να δημιουργηθούν, οπότε η πιθανότητα να ταιριάξουμε ένα ιδιωτικό κλειδί με ένα δημόσιο κλειδί είναι 1 στις 1,550 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 ... 000 (75 μηδενικά).



Σχήμα 2.7– Ethereum Address Generation

2.2.2 Συναρτήσεις Κατακερματισμού

Μία συνάρτηση κατακερματισμού (hash function), είναι μια συνάρτηση, η οποία χρησιμοποιείται για να μετατρέψει (κρυπτογραφήσει) μία οποιουδήποτε μήκους συμβολοσειρά, την οποία δέχεται ως είσοδο (input), σε μία νέα, σταθερού μήκους συμβολοσειρά, την οποία προσδίδει ως έξοδο (hash value).

Οι συναρτήσεις κατακερματισμού αποτελούν αναπόσπαστο στοιχείο της δομικής λειτουργίας του Ethereum, και γενικότερα κάθε αλυσίδας τύπου Blockchain . Στην πραγματικότητα, οι hash functions χρησιμοποιούνται εκτενώς σε σχεδόν όλα τα κρυπτογραφικά συστήματα – κάτι το οποίο καταγράφηκε από τον διάσημο κρυπτογράφο *Bruce Schneier*, ο οποίος είπε: «Πολύ περισσότερο από τους αλγόριθμους κρυπτογράφησης, οι μονόδρομες συναρτήσεις κατακερματισμού είναι οι λειτουργίες της σύγχρονης κρυπτογραφίας».

Ενδεικτικά αναφέραμε μια χρήση τέτοιας συνάρτησης προηγουμένως (Keccak-256), κατά την μεταμόρφωση ενός δημόσιου κλειδιού σε διεύθυνση Ethereum. Πέραν αυτής της διαδικασίας, μπορούν επίσης να χρησιμοποιηθούν για τη δημιουργία ψηφιακών δακτυλικών αποτυπωμάτων, τα οποία βοηθούν στην επαλήθευση των δεδομένων.

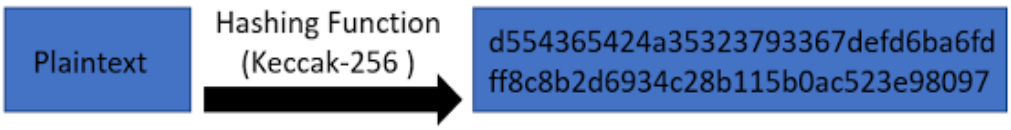
Οι συναρτήσεις κατακερματισμού που θα μας απασχολήσουν είναι οι κρυπτογραφικές συναρτήσεις κατακερματισμού. Μια τέτοια συνάρτηση είναι μια μονόδρομη (one-way) συνάρτηση κατακερματισμού που μετατρέπει δεδομένα αυθαίρετου μεγέθους σε μια συμβολοσειρά bit μόνιμου μεγέθους. Η «μονόδρομη» ιδιότητα σημαίνει ότι είναι υπολογιστικά αδύνατη η επαναδημιουργία των δεδομένων εισόδου, εάν κάποιος γνωρίζει μόνο τον κατακερματισμό εξόδου. Ο μόνος τρόπος για να προσδιοριστεί μια πιθανή είσοδος είναι να διεξαχθεί μια brute-force επίθεση, ελέγχοντας κάθε πιθανή είσοδο για μια αντίστοιχη έξοδο. Δεδομένου ότι ο χώρος αναζήτησης είναι ουσιαστικά άπειρος, καταλαβαίνουμε εύκολα ότι μια τέτοια έρευνα είναι πρακτικά αδύνατη. Ακόμα κι αν βρεθούν κάποια δεδομένα εισόδου που δημιουργούν ένα ταιριαστό κατακερματισμό, αυτή η είσοδος μπορεί να μην είναι η

ίδια με την οποία λειτούργησε αρχικά η συνάρτηση. Η εύρεση δύο συνόλων δεδομένων εισόδου που οδηγούν με κατακερματισμό στην ίδια έξοδο καλείται εύρεση σύγκρουσης κατακερματισμού (finding a hash collision). Σε γενικές γραμμές, όσο καλύτερη είναι η συνάρτηση κατακερματισμού, τόσο πιο σπάνιες είναι οι συγκρούσεις κατακερματισμού. Για το Ethereum, η εύρεση τέτοιων συγκρούσεων είναι ουσιαστικά αδύνατη. Έχει υπολογιστεί ότι ο SHA-3 (Keccak-256), που χρησιμοποιεί το Ethereum, έχει μια πολυπλοκότητα επίθεσης περίπου 2^{128} υπολογισμοί (αποδεικνύεται ότι αν *n* είναι το μήκος εξόδου μιας hash function τότε περίπου η πολυπλοκότητα επίθεσης της συνάρτησης είναι $2^{n/2}$ υπολογισμοί και το security level του αλγορίθμου είναι τα *n/2* bits).

Keccak-256

Το Ethereum χρησιμοποιεί τη συνάρτηση κρυπτογραφικού κατακερματισμού Keccak-256 σε πολλές λειτουργίες του. Η συνάρτηση αυτή σχεδιάστηκε ως υποψήφια για να πάρει τη θέση της SHA-3 στον διαγωνισμό "Cryptographic Hash Function" που έγινε το 2007 από το Εθνικό Ινστιτούτο Επιστήμης και Τεχνολογίας, κάτι το οποίο το κατάφερε με επιτυχία. Λόγω του ότι υπήρξαν κάποιες επιπλοκές στην άμεση καθιέρωση του αλγορίθμου Keccak-256 ως SHA-3, κατά την περίοδο κατασκευής του Ethereum που δεν είχε ακόμα πάρει την τελική του μορφή ως SHA-3, οι προγραμματιστές της πλατφόρμας τον χρησιμοποίησαν όπως ακριβώς είχε προταθεί στον διαγωνισμό εκείνο.

Μια λειτουργία του Keccak-256 στο Ethereum όπως είδαμε πριν είναι για την απόδοση διευθύνσεων Ethereum στους χρήστες του, αφού πρώτα έχει βρεθεί το δημόσιο κλειδί μέσω ιδιωτικού κλειδιού και της κρυπτογραφίας ελλειπτικών καμπυλών. Έχοντας το δημόσιο κλειδί (συμβολοσειρά μεταβλητού μήκους), εφαρμόζουμε σε αυτό την συνάρτηση Keccak-256 η οποία παράγει το hash της εξόδου μας. Από το hash αυτό κρατάμε τα τελευταία 20 bytes (λιγότερο σημαντικά ψηφία) τα οποία συνήθως μαζί με το πρόθεμα 0x στην αρχή τους μας δίνουν την αναγνωριστική διεύθυνση Ethereum που μας αναλογεί.



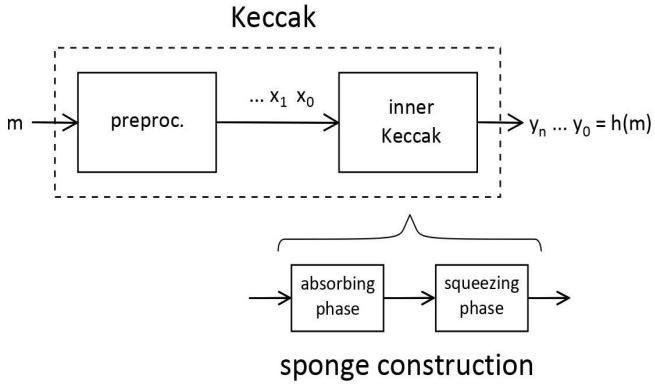
Σχήμα 2.8– Παράδειγμα συνάρτησης κατακερματισμού

Ας δούμε πως ακριβώς δουλεύει ο αλγόριθμος Keccak όπως προτάθηκε το 2007.

Μια βασική απαίτηση για τη καθιέρωση της συνάρτησης κατακερματισμού Keccak ως SHA-3 ήταν η υποστήριξη της των ακόλουθων μηκών εξόδου: 224 bits, 256 bits, 384 bits και 512 bits. Να σημειώσουμε εδώ ότι το Blockchain του Ethereum χρησιμοποιεί μήκος κατακερματισμού εξόδου 256 bits (για αυτό και το όνομα Keccak-256). Η συνάρτηση κατακερματισμού Keccak βασίζεται σε αυτό που ονομάζεται "κατασκευή σφουγγαριού". Μετά την προ-επεξεργασία (pre-processing),η οποία χωρίζει το μήνυμα m σε blocks x0,x1,x2… συγκεκριμένου μεγέθους και παρέχει επένδυση (padding) σε αυτά, ακολουθεί η "κατασκευή σφουγγαριού" (inner Keccak) που αποτελείται από δύο φάσεις:

- **Φάση απορρόφησης (ή εισόδου)**: Το block μηνυμάτων xi, για i=0,1,2…, μεταβιβάζεται στον αλγόριθμο και επεξεργάζεται.

- **Φάση συμπίεσης (ή εξόδου)**: Υπολογίζεται μια έξοδος σταθερού μήκους y0. (στο σχήμα παρακάτω φαίνεται πως είναι η έξοδος πολλών κατακερματισμών)



Σχήμα 2.9– High-level προσέγγιση του Keccak

Υπάρχουν πολλές παράμετροι με τις οποίες τα μεγέθη εισόδου και εξόδου καθώς και το επίπεδο ασφάλειας (security level) του Keccak μπορεί να διαμορφωθεί. Οι αντίστοιχες παράμετροι είναι:

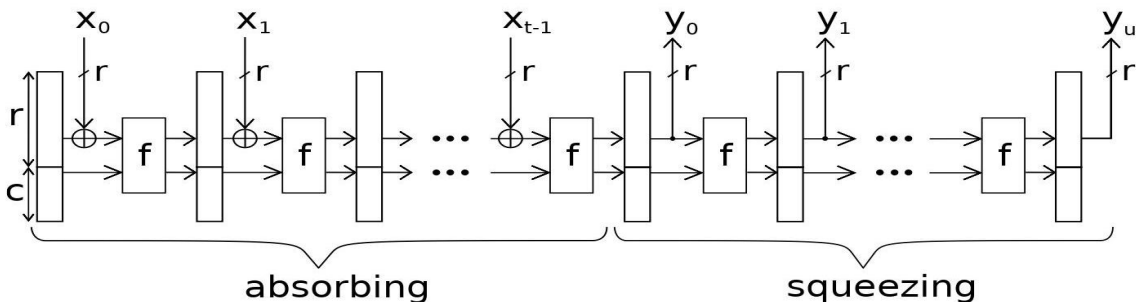
i. **b**: είναι το μέγεθος του μηνύματος που θα κατακερματιστεί ή αλλιώς το πλάτος της κατάστασης όπως αναφέρεται στη βιβλιογραφία. Το πλάτος αυτό εξαρτάται από έναν εκθέτη l και μπορεί να λάβει τις ακόλουθες τιμές: $b = 25 \cdot 2^l$ για l = 0,1, ..., 6. Αυτό σημαίνει ότι η κατάσταση μπορεί να έχει πλάτος b $\in$ {25,50,100,200,400,800,1600}. Για τον SHA-3 χρησιμοποιήθηκε επίσημα μια κατάσταση πλάτους b = 1600 bits (για l=6). (*πίνακας 2.1*)

ii. **r**: Αντιπροσωπεύει το μέγεθος των τμημάτων, στα οποία θα πρέπει να χωριστεί το μήνυμα στο pre-processing, ώστε τα τμήματα αυτά να αποτελούν σωστές εισόδους στα διάφορα τμήματα του Keccak. Για τον SHA-3 επιτρέπεται μόνο r=1344 bits και r=1088 bits.

iii. **c**: Αντιπροσωπεύει την χωρητικότητα, δηλαδή τον ελεύθερο χώρο που θα απομείνει στην κατάσταση b, αφότου ένα μέρος αυτού καταληφθεί από το εκάστοτε block. Για τον SHA-3 επιτρέπεται μόνο c=256 bits και r=512 bits.

Πρέπει να ισχύει πάντα ότι το r + c είναι ένα έγκυρο πλάτος κατάστασης, δηλαδή

$r + c = b \in \{25,50,100,200,400,800,1600\}$

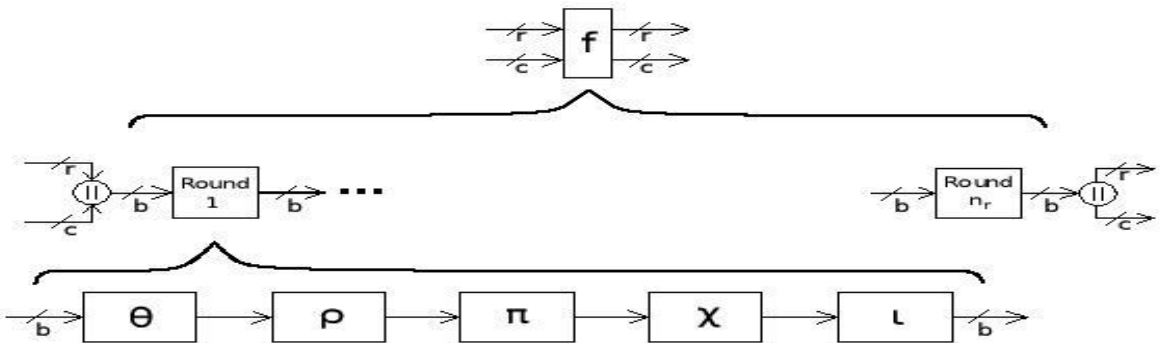


Σχήμα 2.10– Φάση απορρόφησης και αποσυμπίεσης στην "κατασκευή σφουγγαριού"

b (state) [bits]	*r* [bits]	*c* [bits]	security level [bits]	hash output [bits]
1600	1344	256	128	224
1600	1344	256	128	256
1600	1088	512	256	384
1600	1088	512	256	512

Πίνακας 2.1– Παράμετροι του SHA-3

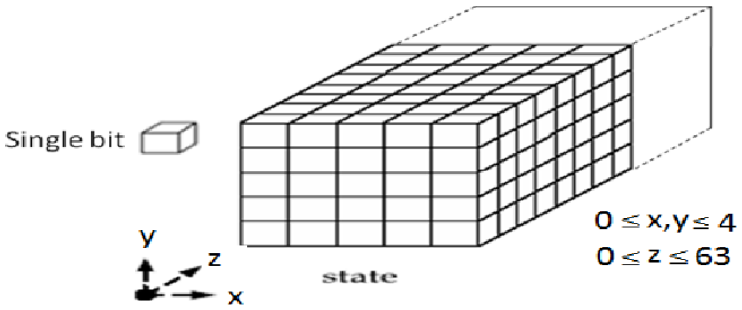
Η συνάρτηση Keccak-f είναι το κεντρικό σημείο της συνάρτησης κατακερματισμού και χρησιμοποιείται και στις δύο φάσεις της "κατασκευής σφουγγαριού". Η συνάρτηση εκτελείται σε n_r γύρους. Κάθε γύρος έχει μια είσοδο που αποτελείται από $b = r + c$ *bits*. Ο αριθμός των γύρων εξαρτάται από την παράμετρο l: $n_r = 12 + 2l$. Στον SHA-3 όπως είδαμε πριν χρησιμοποιούμε l=6, άρα ο αριθμός των γύρων θα είναι n_r=24. Ας εξετάσουμε τώρα την εσωτερική δομή της Keccak-f, η οποία απεικονίζεται στο παρακάτω σχήμα.



Σχήμα 2.11– Εσωτερικό της Keccak-f συνάρτησης

Τα δεδομένα χωρίζονται σε r και c bits στο κάθε βήμα. Αρχικά όλα τα bits έχουν την τιμή 0, ενώ έπειτα αρχίζει η φάση της απορρόφησης, όπου τα τμήματα του μηνύματος δίνονται ανά r bits. Πραγματοποιείται η πράξη xor (❑⊕❑) μεταξύ των υπαρχόντων και των νέων bits. Στη συνέχεια, τα δεδομένα εισόδου δίνονται στη συνάρτηση f, που αποτελείται από 5 βήματα-συναρτήσεις, οι οποίες εκτελούνται για πολλούς γύρους. Η έξοδος της συναρτήσεως αυτής δίνεται ως είσοδος στο επόμενο βήμα και η διαδικασία επαναλαμβάνεται μέχρις ότου και το τελευταίο μέρος του

μηνύματος υποστεί επεξεργασία. Ως έξοδος, δεν δίνονται και τα 1600 bits που παράγει ο αλγόριθμος, αλλά επιλέγονται τα πιο σημαντικά (most significant bits) του r 'κομματιού' του, έως ότου συμπληρωθεί το μήκος της τιμής εξόδου που έχει επιλεγεί . Εντός της f, εκτελούνται $n_r = 24$ γύροι, καθένας εκ των οποίων αποτελείται από την εκτέλεση 5 συναρτήσεων. Σε κάθε γύρο δίνεται είσοδος b bits και εξάγεται έξοδος πάλι b bits, η οποία λειτουργεί ως είσοδος στον επόμενο γύρο, με τη διαδικασία να επαναλαμβάνεται μέχρι να ολοκληρωθούν οι γύροι. Τα 5 βήματα-συναρτήσεις που εκτελούνται μες στην f με την ακόλουθη σειρά είναι τα εξής (δόθηκαν από τους δημιουργούς ελληνικά ονόματα): Theta (θ) Step, Rho (ρ) Step, Pi (π) Step, Chi (χ) Step και Iota (ι) Step. Τα b=1600 bits αναπαριστάνονται ως ένας τρισδιάστατος πίνακας 5x5x64 όπως φαίνεται παρακάτω:



Σχήμα 2.12– Πίνακας κατάστασης του SHA-3

Theta (θ) Step

Έστω ότι δοθέντος (x,y) συντεταγμένων με x,y=0,1,2,3,4 (αν φανταστούμε τον παραπάνω πίνακα ως έναν πίνακα A[] 5x5), ορίσουμε ως w=64bits (lane) το βάθος του τρισδιάστατου πίνακα και κάθε τέτοιο βάθος το λέμε λέξη. Έστω επίσης C[x] και D[x] μονοδιάστατα διανύσματα που περιέχουν πέντε λέξεις μήκους w bits, το σύμβολο ❑⊕❑ είναι ο δυαδικός τελεστής xor μεταξύ των w bits και rot (C [], 1) δηλώνει μια περιστροφή του τελεστή κατά ένα bit. Αυτή η περιστροφή είναι προς την κατεύθυνση του z άξονα. Το θ step είναι το εξής:

$$C[x] = A[x,0] \oplus A[x,1] \oplus A[x,2] \oplus A[x,3] \oplus A[x,4]$$

$$D[x] = C[(x\text{-}1) \bmod 5] \oplus \mathrm{rot}(C[(x+1 \bmod 5)], 1)$$

$$A[x, y] = A[x, y] \oplus D[x]$$

Rho (ρ) και Pi (π) Steps

Τα επόμενα δύο βήματα υπολογίζουν έναν βοηθητικό πίνακα B[] 5x5 από τον πίνακα κατάστασης A. Και τα δύο βήματα μπορούν να εκφραστούν από κοινού από τον ακόλουθο ψευδοκώδικα: $B\,[y,\ 2x + 3y] = \mathrm{rot}\,(A\,[x,\ y], r\,[x,\ y])$ με x, y = 0,1,2,3,4. Στα βήματα αυτά κάθε λέξη A[x,y] περιστρέφεται κατά έναν συγκεκριμένο αριθμό r[x,y], ο οποίος καθορίζεται στον διπλανό πίνακα, και τοποθετείται στον νέο πίνακα B.

	x = 3	x = 4	x = 0	x = 1	x = 2
y=2	25	39	3	10	43
y=1	55	20	36	44	6
y=0	28	27	0	1	62
y=4	56	14	18	2	61
y=3	21	8	41	45	15

Πίνακας 2.2– Τιμές σταθερών r[x,y]

Chi (x) Step

Στο βήμα αυτό δίνεται είσοδος ο πίνακας B[x,y] των προηγούμενων βημάτων, όπου x, y = 0,1,2,3,4 και παράγεται έξοδος C[x,y] ως εξής:

$C[x,y] = B[x,y] \oplus (B'[(x+1) \bmod 5, y] \wedge B[(x+2) \bmod 5, y])$, όπου B' είναι το συμπλήρωμα του B και ο τελεστής $\wedge$ είναι ο δυαδικός τελεστής AND.

Iota (ι) Step

Στο τελευταίο αυτό βήμα προστίθεται μια προκαθορισμένη λέξη w bits στη θέση [0,0] του πίνακα C:

C [0,0] = C [0,0] $\oplus$ RC [i], όπου η σταθερά RC [i] διαφέρει για κάθε γύρο i=0,1, …,23 ως εξής:

RC[0] = 0x0000000000000001
RC[1] = 0x0000000000008082
RC[2] = 0x800000000000808A
RC[3] = 0x8000000080008000
RC[4] = 0x000000000000808B
RC[5] = 0x0000000080000001
RC[6] = 0x8000000080008081
RC[7] = 0x8000000000008009
RC[8] = 0x000000000000008A
RC[9] = 0x0000000000000088
RC[10] = 0x0000000080008009
RC[11] = 0x000000008000000A
RC[12] = 0x000000008000808B
RC[13] = 0x800000000000008B
RC[14] = 0x8000000000008089
RC[15] = 0x8000000000008003
RC[16] = 0x8000000000008002
RC[17] = 0x8000000000000080
RC[18] = 0x000000000000800A
RC[19] = 0x800000008000000A
RC[20] = 0x8000000080008081
RC[21] = 0x8000000000008080
RC[22] = 0x0000000080000001
RC[23] = 0x8000000080008008

2.2.3 Merkle trees

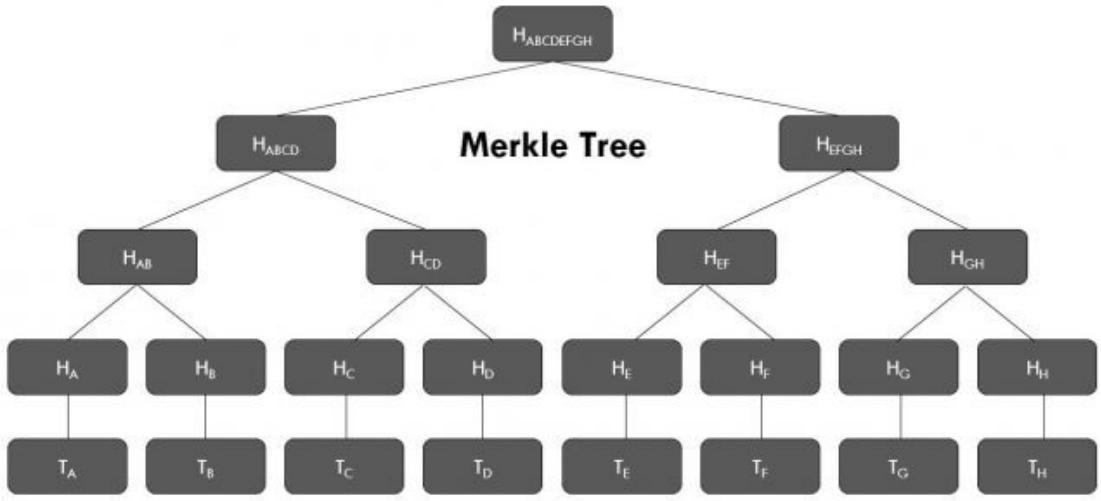
Τα Merkle trees αποτελούν ένα βασικό συστατικό της τεχνολογίας Blockchain, οπότε δε θα μπορούσαμε να μην αναφερθούμε σε αυτά. Ένα Merkle Tree είναι μία δομή δεδομένων, η οποία αναπαρίσταται ως ένα δέντρο και κάθε φύλλο του δέντρου αναπαριστά ένα κρυπτογραφημένο μήνυμα, κάθε κλαδί του αναπαριστά τη συνδυασμένη τιμή κατακερματισμού εξόδου (hash value) των παιδιών της, βάσει μιας συνάρτησης κατακερματισμού (hash function), ενώ η ρίζα του (Merkle root) αποτελείται από μία μόνο τιμή, η οποία αναπαριστά τη συνολική συνδυαστική τιμή κατακερματισμού όλου του δέντρου. Ένα Merkle Tree ουσιαστικά επιτρέπει σε μεγάλο μέρος πληροφοριών (συναλλαγών) να επαληθευτούν ως προς την ακρίβεια τους εξαιρετικά γρήγορα και αποτελεσματικά.

Η ιστορία των Merkle Trees ξεκινά από το 1979 όταν ο Ralph Merkle φοιτούσε στο Πανεπιστήμιο του Στάνφορντ και έγραψε μια ακαδημαϊκή εργασία με τίτλο «A Certified Digital Signature». Σε αυτή την εργασία ο Merkle περιέγραψε μια μέθοδο δημιουργίας ψηφιακών υπογραφών και καθιέρωσε μια νέα, εξαιρετικά αποτελεσματική μέθοδο δημιουργίας κρυπτογραφικών αποδείξεων. Με άλλα λόγια, σχεδίασε μια διαδικασία για την επαλήθευση δεδομένων που θα επέτρεπε στους υπολογιστές να κάνουν τη δουλειά τους πολύ πιο γρήγορα από ποτέ. Ο Merkle ονόμασε την ιδέα του "Tree Signatures" ή "Tree Authentication". Σήμερα, αυτή η ιδέα είναι πια γνωστή ως Merkle Tree λόγω του εφευρέτη της. Τα Merkle Trees αναφέρονται επανειλημμένα στο white-paper του Satoshi Nakamoto το 2008 που εισήγαγε το Bitcoin στον κόσμο. Πλέον χρησιμοποιούνται εκτενώς στο πρωτόκολλο

Bitcoin και κυρίως στο πρωτόκολλο του Ethereum με την μορφή των Merkle Patricia Tries. Ας δούμε πως ακριβώς δουλεύουν τα Merkle trees γενικά.

Κάθε συναλλαγή σε ένα Blockchain έχει το δικό της μοναδικό αναγνωριστικό συναλλαγής (id). Για τα περισσότερα Blockchain, ένα αναγνωριστικό είναι ένας κωδικός 64 χαρακτήρων που καταλαμβάνει μνήμη 32 bytes. Αν αναλογιστούμε ότι κάθε αλυσίδα τύπου Blockchain αποτελείται από χιλιάδες blocks και ότι κάθε block μπορεί να περιέχει έως και χιλιάδες συναλλαγές, καθίσταται σαφές ότι η μνήμη και η υπολογιστική ισχύς είναι δύο μεγάλα ζητήματα. Για το λόγο αυτό, θα πρέπει να χρησιμοποιηθούν όσο το δυνατόν λιγότερα δεδομένα κατά την επεξεργασία και την επαλήθευση συναλλαγών. Έτσι θα μειωθούν οι χρόνοι επεξεργασίας της CPU μνήμης και θα διασφαλιστεί υψηλότερο επίπεδο ασφάλειας. Και αυτό ακριβώς κάνουν το Merkle Trees. Ουσιαστικά, ένα Merkle Tree παίρνει έναν τεράστιο αριθμό αναγνωριστικών συναλλαγών (id's) και τα τρέχει μέσω μιας μαθηματικής διαδικασίας που οδηγεί σε έναν κωδικό 64 χαρακτήρων, ο οποίος ονομάζεται Merkle Root. Το Merkle Root είναι εξαιρετικά σημαντικό επειδή επιτρέπει σε οποιονδήποτε υπολογιστή να επαληθεύσει γρήγορα ότι μια συγκεκριμένη συναλλαγή πραγματοποιήθηκε σε ένα συγκεκριμένο block όσο το δυνατόν ακριβέστερα.

Τα Merkle Trees χρησιμοποιούν πάντα ζευγάρια συναλλαγών. Αν ο αριθμός τους είναι περιττός, τότε δημιουργείται ένας κλώνος της τελευταίας συναλλαγής και γίνεται ζευγάρι με τον εαυτό της, καθώς και όταν κατά πλάτος ενός επιπέδου, το σύνολο των κλαδιών που προκύπτουν είναι περιττό, τότε και πάλι δημιουργείται κλώνος του τελευταίου κλαδιού και γίνεται ζευγάρι με τον εαυτό του. Σε κάθε γύρο το σύνολο των νέων κλαδιών ή φύλλων περνάει σε ζευγάρια από μια συνάρτηση κατακερματισμού (hash function), από την οποία προκύπτει το επόμενο επίπεδο, το οποίο περιέχει τη συνδυασμένη τιμή κατακερματισμού των δύο και η διαδικασία συνεχίζεται, έως ότου προκύψει η ρίζα του δέντρου Merkle root. Το παρακάτω σχήμα δίνει μια οπτική στα Merkle trees. Ως H συμβολίζεται η hash function και T_A,$T_{B,...,}T_H$ είναι οι συναλλαγές που πρέπει να μπουν στο block.



Σχήμα 2.13– Merkle tree

Για να αποδείξει κάποιος ότι μία συναλλαγή περιέχεται στο δέντρο πρέπει να το ψάξει όλο. Επίσης, για να αποδειχτεί ότι κάθε συναλλαγή δεν είναι ψεύτικη, πάλι πρέπει να ψάξει όλο το δέντρο. Ωστόσο, αν κάποιος θέλει να "χειραγωγήσει" μια συναλλαγή, ώστε να προκαλέσει πρόβλημα στο Blockchain, αυτό γίνεται αμέσως αντιληπτό, καθώς η ρίζα του δέντρου αλλάζει τελείως. Οπότε, τα Merkle Trees προσφέρουν ασφάλεια, αλλά έχουν πολύ μεγάλο χρόνο αναζήτησης. Ωστόσο υπάρχει

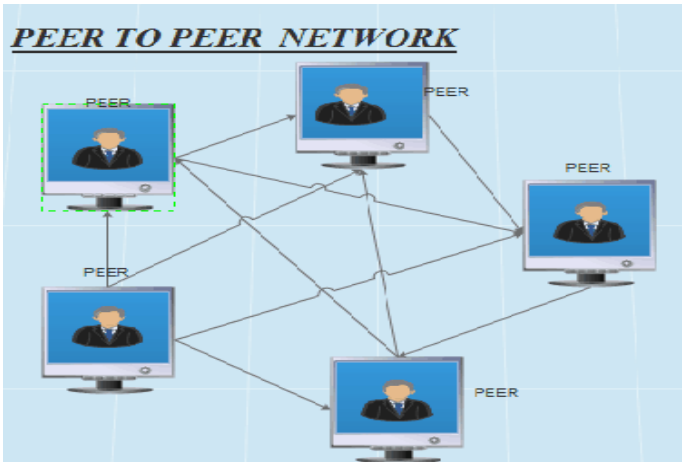
ένας "γρήγορος" τρόπος, ώστε κάποιος που απλά θέλει να ελέγξει αν μια συναλλαγή που τον αφορά περιέχεται στο συγκεκριμένο block χωρίς να ελέγξει όλες τις συναλλαγές. Συγκεκριμένα, δίνονται μόνο κάποιες απαραίτητες πληροφορίες που θα χρειαστεί σε κάθε επίπεδο του δέντρου (ένα υποσύνολο των τιμών H_i του δέντρου για i=A,..,H σύμφωνα με το σχήμα), ώστε βάζοντας τη συναλλαγή (άρα και τον κατακερματισμό της) να μπορεί να σχηματίσει τη ρίζα του δέντρου και να δει αν αυτή ταυτίζεται με την πραγματική. Αν ναι, η συναλλαγή περιέχεται όντως στο block.

2.3 Δίκτυα Ομότιμων Κόμβων

Η αρχιτεκτονική ομότιμων κόμβων (peer-to-peer ή P2P) αποτελεί την θεμελιώδη αρχιτεκτονική πάνω στην οποία είναι βασισμένη η τεχνολογία του Blockchain.

Στο Διαδίκτυο, όπως είναι δομημένο σήμερα, δύο είναι τα κυρίαρχα μοντέλα δικτυακών εφαρμογών: η αρχιτεκτονική πελάτη-εξυπηρετητή (client-server) και η αρχιτεκτονική ομότιμων. Στην αρχιτεκτονική πελάτη-εξυπηρετητή υπάρχει μονίμως ένας ενεργός υπολογιστής, ο εξυπηρετητής, ο οποίος εξυπηρετεί αιτήσεις για παροχή υπηρεσιών από άλλους υπολογιστές, τους πελάτες. Γίνεται έτσι αντιληπτό πως ο ρόλος του εξυπηρετητή για την λειτουργία τέτοιας αρχιτεκτονικής δικτύων είναι καθοριστικός. Αν λοιπόν ο εξυπηρετητής αφαιρεθεί, τότε το δίκτυο καταρρέει.

Αντιθέτως, στην αρχιτεκτονική ομότιμων κόμβων υπάρχει μικρή ή και καμία στήριξη σε αποκλειστικούς εξυπηρετητές σε κέντρα δεδομένων. Σε ένα δίκτυο ομότιμων κόμβων οι "ομότιμοι" είναι συστήματα υπολογιστών που συνδέονται μεταξύ τους μέσω του Διαδικτύου. Μεταξύ των υπολογιστών αυτών είναι δυνατός ο διαμοιρασμός αρχείων και πληροφοριών χωρίς την ανάγκη ύπαρξης κεντρικού διακομιστή (εξυπηρετητή). Με άλλα λόγια, κάθε υπολογιστής σε ένα δίκτυο P2P γίνεται διακομιστής αρχείων αλλά και πελάτης. Στο δίκτυο αυτό κάθε υπολογιστής (κόμβος) διαχειρίζεται την ύπαρξη του χωρίς την ανάγκη των υπολοίπων και όλοι έχουν ίσα δικαιώματα για την αποκατάσταση της επικοινωνίας μεταξύ τους, την κοινή χρήση πόρων και την επικύρωση νέων χρηστών στο δίκτυο.



Σχήμα 2.14– Δίκτυο ομότιμων κόμβων

Σε ένα P2P δίκτυο οι χρήστες αποθηκεύουν αρχεία στους δικούς τους υπολογιστές και είναι υπεύθυνοι για τη διασφάλιση της σωστής δημιουργίας αντιγράφων ασφαλείας αυτών των αρχείων. Κάθε υπολογιστής εκτελεί συνήθως λογισμικό και πελάτη και εξυπηρετητή και μπορεί να χρησιμοποιηθεί για τη διάθεση πόρων σε άλλους χρήστες ή για πρόσβαση σε κοινόχρηστους πόρους στο δίκτυο. Στα πλεονεκτήματα της αρχιτεκτονικής ομότιμων θα μπορούσαμε να επισημάνουμε τα

παρακάτω:

- Μη ανάγκη ύπαρξης κεντρικού λειτουργικού συστήματος για το δίκτυο.
- Μη ανάγκη ύπαρξης ακριβού διακομιστή επειδή μεμονωμένοι σταθμοί εργασίας χρησιμοποιούνται για πρόσβαση στα αρχεία.
- Μη ανάγκη ύπαρξης εξειδικευμένου προσωπικού, όπως τεχνικοί δικτύου, επειδή κάθε χρήστης ορίζει τα δικά του δικαιώματα για τα αρχεία που είναι διατεθειμένος να μοιραστεί.
- Πολύ ευκολότερη εγκατάσταση από ένα δίκτυο με αρχιτεκτονική πελάτη-εξυπηρετητή .
- Εάν ένας υπολογιστής σταματήσει τη λειτουργία του, δεν θα διακόψει το δίκτυο. Απλά τα αρχεία που παρείχε δεν θα είναι διαθέσιμα σε άλλους χρήστες εκείνη τη στιγμή.

Μερικές από τις βασικές προκλήσεις που αντιμετωπίζουν οι εφαρμογές αρχιτεκτονικής ομότιμων είναι:

- *Μη φιλικότητα προς τους ISP (Internet Service Providers):* Οι περισσότεροι ISP έχουν διασταθιοποιηθεί για ασύμμετρη χρησιμοποίηση του εύρους ζώνης, δηλαδή για περισσότερη συρρευματική, παρά αντιρρευματική κίνηση, δυσκολεύοντας έτσι την παροχή πόρων προς το σύστημα από τους ομότιμους κόμβους.
- *Ασφάλεια:* Λόγω της κατανεμημένης και ανοικτής φύσης τους, οι εφαρμογές P2P μπορούν να δημιουργήσουν προβλήματα ασφαλείας. Στο Blockchain αυτό λύνεται με τη χρήση κρυπτογραφίας όπως αναφέραμε νωρίτερα.
- *Κίνητρα:* Ποια κίνητρα έχουν οι κόμβοι, ώστε να παρέχουν πόρους στο σύστημα. Και εδώ στο Blockchain (π.χ Ethereum) παρέχονται χρηματικά κίνητρα (gas) στους κόμβους που υποστηρίζουν την λειτουργία του δικτύου (miners).

Με βάση τον τρόπο με τον οποίο οι κόμβοι συνδέονται μεταξύ τους εντός του δικτύου και τον τρόπο που οι πόροι εντοπίζονται απ' τον κάθε χρήστη, μπορούμε να ταξινομήσουμε τα δίκτυα ως αδόμητα ή δομημένα.

Τα αδόμητα δίκτυα ομότιμων κόμβων δεν επιβάλλουν συγκεκριμένη δομή στο δίκτυο ως προς το σχεδιασμό τους, αλλά αποτελούνται από κόμβους που σχηματίζουν τυχαία συνδέσεις μεταξύ τους (π.χ Gnutella, Kazaa κλπ.). Επειδή δεν υπάρχει συγκεκριμένη δομή σε αυτά, τα αδόμητα δίκτυα δημιουργούνται εύκολα και επιτρέπουν τοπικές βελτιστοποιήσεις σε διαφορετικές περιοχές της επικάλυψης του δικτύου. Επίσης, επειδή ο ρόλος όλων των "ομότιμων" στο δίκτυο είναι ο ίδιος, τα αδόμητα δίκτυα είναι πολύ ισχυρά ενόψει των υψηλών ποσοστών αναχωρούντων κόμβων από το δίκτυο (churn). Ορισμένα μειονεκτήματά τους είναι ο πλημμυρισμός του δικτύου σε κάθε ερώτημα αναζήτησης δεδομένων. Όταν συμβαίνει αυτό προκαλείται αυξημένη κίνηση και κατασπατάληση των πόρων του δικτύου λόγω της δομής του και επίσης δεν εξασφαλίζεται ότι θα ζητηθεί από τον σωστό κόμβο ο προς αναζήτηση πόρος.

Στα δομημένα δίκτυα ομότιμων κόμβων, η επικάλυψη του δικτύου οργανώνεται βάσει συγκεκριμένης τοπολογίας και το πρωτόκολλο διασφαλίζει ότι οποιοσδήποτε κόμβος μπορεί αποτελεσματικά να αναζητήσει στο δίκτυο ένα πόρο, ακόμα και αν ο πόρος είναι εξαιρετικά σπάνιος/δυσεύρετος. Ο πιο κοινός τύπος δομημένων δικτύων P2P υλοποιεί έναν κατανεμημένο πίνακα κατακερματισμού (DHT), στον οποίο χρησιμοποιείται μια παραλλαγή της συνεπούς συνάρτησης κατακερματισμού (consistent hashing) για την εκχώρηση ιδιοκτησίας κάθε πόρου σε ένα συγκεκριμένο

κόμβο. Αυτό επιτρέπει στους "ομότιμους" να αναζητήσουν πόρους στο δίκτυο χρησιμοποιώντας έναν πίνακα κατακερματισμού. Δηλαδή, ζεύγη (key,value) αποθηκεύονται στο DHT και οποιοσδήποτε κόμβος μπορεί να ανακτήσει αποτελεσματικά την τιμή που σχετίζεται με ένα δεδομένο κλειδί. Ωστόσο, επειδή οι κόμβοι σε ένα δομημένο δίκτυο πρέπει να διατηρούν λίστες γειτόνων που πληρούν συγκεκριμένα κριτήρια, αυτό τα καθιστά λιγότερο ισχυρά έναντι δικτύων με μεγάλο αριθμό κόμβων που συχνά συνδέονται και αποχωρούν από το δίκτυο. Ορισμένα αξιοσημείωτα ερευνητικά έργα που αξιοποιούν τέτοιου είδους δίκτυα είναι το Chord project, το Kademlia και το P-Grid.

Στο δίκτυο του Ethereum, η επικοινωνία μεταξύ των ομότιμων κόμβων του γίνεται σύμφωνα με το RLPx Transport Protocol που βασίζεται σε ένα πίνακα κατακερματισμού (DHT) Kademlia. Πρόκειται για ένα πρωτόκολλο μεταφοράς TCP ώστε να επικοινωνούν οι κόμβοι στο εσωτερικό του Ethereum.

3.Ethereum

3.1 Εισαγωγή

Θα προσεγγίσουμε αρχικά το Ethereum με μια σύγκριση του με το Bitcoin μιας και οι περισσότεροι πιθανόν να έχουν ακούσει στο παρελθόν περισσότερα για το Bitcoin παρά για το Ethereum. Η καινοτομία του Ethereum σε σχέση με το Bitcoin, είναι πως το Ethereum αποτελεί μία πιο ευέλικτη και προσαρμόσιμη πλατφόρμα, πάνω στην οποία μπορούν να δημιουργηθούν και να λειτουργήσουν με ασφάλεια αποκεντρωμένες εφαρμογές (Dapps), ενώ το Bitcoin παρέχει κυρίως την δυνατότητα (οικονομικών) συναλλαγών του κρυπτονομίσματος (Bitcoin). Ας δούμε όμως ακριβώς τι είναι το Ethereum.

Το Ethereum είναι μια ανοιχτή πλατφόρμα Blockchain που επιτρέπει σε όλους να δημιουργούν και να χρησιμοποιούν αποκεντρωμένες εφαρμογές που λειτουργούν με χρήση της τεχνολογίας Blockchain. Όπως το Bitcoin, κανείς δεν ελέγχει ούτε έχει δικαιώματα ιδιοκτησίας του Ethereum. Πρόκειται για ένα έργο ανοιχτού κώδικα που έχει κατασκευαστεί από πολλούς ανθρώπους ανά τον κόσμο. Το Ethereum είναι δηλαδή ένα προγραμματιζόμενο Blockchain. Αντί να παρέχει στους χρήστες ένα σύνολο προκαθορισμένων λειτουργιών (π.χ. συναλλαγές Bitcoin), το Ethereum επιτρέπει στους χρήστες να δημιουργούν τις δικές τους λειτουργίες οποιασδήποτε πολυπλοκότητας που επιθυμούν, μέσω των έξυπνων συμβολαίων (προγράμματα κώδικα). Με αυτόν τον τρόπο, χρησιμεύει ως πλατφόρμα για πολλούς διαφορετικούς τύπους αποκεντρωμένων εφαρμογών Blockchain, ένας εκ των οποίων ενδεικτικά αποτελεί το ίδιο το κρυπτονόμισμα του Ethereum ETH. Εδώ αξίζει να αναφέρουμε ότι η αποκεντρωμένη εφαρμογή ηλεκτρονικής ψηφοφορίας που δημιουργήθηκε στα πλαίσια της διπλωματικής "τρέχει" στο Blockchain του Ethereum.

Μερικές βασικές αρχές που κρύβονται πίσω από το Ethereum είναι οι ακόλουθες :

- **Απλότητα**: Το πρωτόκολλο Ethereum θα πρέπει να είναι όσο το δυνατόν απλούστερο, ακόμη και με κόστος αποθήκευσης δεδομένων ή χρόνου. Ένας μέσος προγραμματιστής θα πρέπει ιδανικά να μπορεί να ακολουθεί και να εφαρμόζει όλες τις δυνατότητες που του προσφέρει η πλατφόρμα, προκειμένου να αξιοποιηθεί πλήρως το πρωτοφανές δυναμικό εκδημοκρατισμού που φέρνει η

κρυπτογράφηση και να προωθήσει το όραμα του Ethereum ως πρωτοκόλλου που είναι ανοιχτό σε όλους. Οποιαδήποτε βελτιστοποίηση που προσθέτει πολυπλοκότητα δεν πρέπει να συμπεριλαμβάνεται εκτός εάν η βελτιστοποίηση παρέχει πολύ σημαντικό όφελος.

- **Καθολικότητα**: Ένα θεμελιώδες μέρος της σχεδιαστικής φιλοσοφίας του Ethereum είναι ότι το Ethereum δεν έχει "χαρακτηριστικά". Αντί αυτού, το Ethereum παρέχει μια εσωτερική γλώσσα δέσμης ενεργειών Turing Complete που τρέχει στην Ethereum Virtual Machine, την οποία ένας προγραμματιστής μπορεί να χρησιμοποιήσει για να κατασκευάσει οποιοδήποτε έξυπνο συμβόλαιο ή τύπο συναλλαγής που μπορεί να καθοριστεί μαθηματικά.
- **Αρθρωτότητα**: Τα μέρη του πρωτοκόλλου Ethereum πρέπει να είναι σχεδιασμένα ώστε να είναι όσο το δυνατόν πιο αρθρωτά και διαχωρίσιμα. Κατά τη διάρκεια της ανάπτυξης, ο στόχος μας είναι να δημιουργήσουμε ένα πρόγραμμα όπου εάν κάποιος έκανε μια μικρή τροποποίηση πρωτοκόλλου σε ένα μέρος, η στοίβα εφαρμογών θα συνέχιζε να λειτουργεί χωρίς καμία περαιτέρω τροποποίηση. Έτσι όταν οι δυνατότητες του Ethereum αξιοποιούνται στο μέγιστο, τότε ωφελείται ολόκληρο το "οικοσύστημα" κρυπτογράφησης και όχι μόνο το ίδιο το Ethereum.
- **Ευελιξία**: Οι λεπτομέρειες του πρωτοκόλλου Ethereum δεν είναι απλές και λιτές. Ωστόσο ο κάθε προγραμματιστής που θα ασχοληθεί με αυτό μπορεί να κάνει ορισμένες τροποποιήσεις, π.χ. στην αρχιτεκτονική πρωτοκόλλου ή στην Ethereum Virtual Machine (EVM), που θα βελτιώσουν ουσιαστικά την επεκτασιμότητα ή την ασφάλεια του πρωτοκόλλου.
- **Μη διάκριση και μη λογοκρισία**: Το πρωτόκολλο δεν πρέπει να προσπαθεί να περιορίσει ενεργά ή να αποτρέψει συγκεκριμένες κατηγορίες χρήσης. Όλοι οι ρυθμιστικοί μηχανισμοί του πρωτοκόλλου πρέπει να είναι σχεδιασμένοι ώστε να ρυθμίζουν άμεσα τη βλάβη και να μην επιχειρούν να αντιταχθούν σε συγκεκριμένες ανεπιθύμητες εφαρμογές. Ένας προγραμματιστής μπορεί ακόμη και να εκτελέσει ένα σενάριο άπειρου βρόχου πάνω από το Ethereum για όσο διάστημα είναι πρόθυμοι να συνεχίσουν να πληρώνουν το τέλος συναλλαγής ανά υπολογιστικό βήμα.

Ethereum Virtual Machine

Το Ethereum είναι ουσιαστικά μια σουίτα πρωτοκόλλων που ορίζουν μια πλατφόρμα για αποκεντρωμένες εφαρμογές. Στο επίκεντρο του βρίσκεται η Ethereum Virtual Machine (EVM), η οποία μπορεί να εκτελέσει κώδικα αυθαίρετης αλγοριθμικής πολυπλοκότητας. Από την σκοπιά της επιστήμης των υπολογιστών, το Ethereum είναι Turing complete. Οι προγραμματιστές μπορούν να δημιουργήσουν εφαρμογές που εκτελούνται στην EVM χρησιμοποιώντας είτε τη κύρια γλώσσα προγραμματισμού του Ethereum, την Solidity είτε φιλικές γλώσσες προγραμματισμού που διαμορφώνονται σε υπάρχουσες γλώσσες όπως η JavaScript και η Python. Εμείς για το backend κομμάτι της εφαρμογής μας που εκτελείται στην EVM χρησιμοποιήσαμε την Solidity. Όπως και τα άλλα Blockchain, το Ethereum περιλαμβάνει ένα πρωτόκολλο δικτύου ομότιμων κόμβων. Το πρωτόκολλο αυτό είναι υπεύθυνο για τον συντονισμό των συνδεδεμένων κόμβων, με σκοπό την απρόσκοπτη λειτουργία του δικτύου. Κάθε κόμβος «τρέχει» την EVM και εκτελεί τις ίδιες εντολές. Λόγω αυτού, το Ethereum αναφέρεται συχνά και ως «παγκόσμιος υπολογιστής». Αυτή η μεγάλη και ταυτόχρονη εκτέλεση υπολογισμών στο δίκτυο του Ethereum δεν γίνεται για επίτευξη μεγαλύτερης αποτελεσματικότητας. Ίσα ίσα, αυτή η διαδικασία καθιστά τους υπολογισμούς πολύ πιο αργούς και ακριβούς από ότι σε έναν παραδοσιακό υπολογιστή. Αντίθετα, κάθε κόμβος του Ethereum εκτελεί την EVM προκειμένου να διατηρήσει την ύπαρξη ομοφωνίας σε όλο το Blockchain. Η αποκεντρωμένη

συναίνεση δίνει στο Ethereum τεράστια επίπεδα ανοχής σφαλμάτων, εξασφαλίζει μηδενικό χρόνο διακοπής λειτουργίας και καθιστά τα δεδομένα που είναι αποθηκευμένα στο Blockchain για πάντα αμετάβλητα.

Πως λειτουργεί το Ethereum;

Ενώ το Blockchain του Bitcoin έχει ως βασική μονάδα τις συναλλαγές, η βασική μονάδα του Ethereum είναι ο λογαριασμός (account). Το Blockchain του Ethereum παρακολουθεί την κατάσταση κάθε λογαριασμού και όλες οι αλλαγές κατάστασης που πραγματοποιούνται είναι μεταφορές αξίας και πληροφοριών μεταξύ λογαριασμών. Κάθε λογαριασμός αναγνωρίζεται από μία 20-byte Ethereum address και κάθε κατάσταση λογαριασμού (account state) περιέχει 4 πεδία:

- Το nonce, έναν μετρητή που χρησιμοποιείται για να επιβεβαιώνεται ότι κάθε συναλλαγή μπορεί να υποβληθεί σε επεξεργασία μόνο μία φορά
- Το τρέχον υπόλοιπο ETH του λογαριασμού
- Ο κωδικός συμβολαίου του λογαριασμού, εάν υπάρχει
- Ο αποθηκευτικός χώρος του λογαριασμού (κενός από προεπιλογή)

Υπάρχουν δύο τύποι λογαριασμών:

- Λογαριασμοί εξωτερικής ιδιοκτησίας (Externally Owned Accounts ή EOAs), οι οποίοι ελέγχονται από ιδιωτικά κλειδιά
- Λογαριασμοί συμβολαίου (Contract Accounts), οι οποίοι ελέγχονται από τον κώδικα του έξυπνου συμβολαίου τους και μπορούν να «ενεργοποιηθούν» μόνο από έναν EOA.

Η βασική διαφορά μεταξύ αυτών των δύο τύπων λογαριασμών βρίσκεται στο ότι οι άνθρωποι ελέγχουν τους EOAs επειδή μπορούν να ελέγχουν τα ιδιωτικά κλειδιά, ενώ οι λογαριασμοί συμβολαίου, από την άλλη πλευρά, διέπονται από τον εσωτερικό τους κώδικα. Εάν λέγαμε ότι «ελέγχονται» από έναν ανθρώπινο χρήστη, αυτό θα οφείλεται στο γεγονός ότι έχουν προγραμματιστεί να ελέγχονται από ένα EOA με συγκεκριμένη διεύθυνση, το οποίο με τη σειρά του ελέγχεται από όποιον κατέχει τα ιδιωτικά κλειδιά που ελέγχουν αυτόν το EOA. Άρα οι λογαριασμοί συμβολαίου εκτελούν μια λειτουργία μόνο όταν τους ζητηθεί από έναν EOA. Ο δημοφιλής όρος «έξυπνα συμβόλαια» αναφέρεται στον κώδικα σε έναν λογαριασμό συμβολαίου - προγράμματα που εκτελούνται όταν αποστέλλεται μια συναλλαγή σε αυτόν τον λογαριασμό. Οι χρήστες μπορούν να δημιουργήσουν νέα συμβόλαια με την ανάπτυξη κώδικα στο Blockchain.

Οι λογαριασμοί EOAs στέλνουν συναλλαγές στο δίκτυο του Ethereum, υπογράφοντας τα δεδομένα της συναλλαγής με το ιδιωτικό τους κλειδί, χρησιμοποιώντας την κρυπτογραφία ελλειπτικών καμπυλών (ECDSA – Elliptic Curve Digital Signature Algorithm). Μία συναλλαγή είναι έγκυρη μόνο εάν είναι υπογεγραμμένη από τον αποστολέα της (από το ιδιωτικό του κλειδί). Ως αποτέλεσμα, το δίκτυο είναι σίγουρο ότι ο αποστολέας της συναλλαγής είναι αυτός που ισχυρίζεται και όχι κάποιος κακόβουλος χρήστης. Οι χρήστες στο δίκτυο του Ethereum πρέπει να πληρώνουν και κάποια μικρά τέλη συναλλαγών στο δίκτυο. Αυτό προστατεύει το Blockchain από κακόβουλες υπολογιστικές εργασίες, όπως επιθέσεις 51% ή άπειρους βρόχους. Ο αποστολέας μιας συναλλαγής λοιπόν πρέπει να πληρώσει για κάθε βήμα του «προγράμματος» που ενεργοποίησε, συμπεριλαμβανομένου του υπολογισμού και της αποθήκευσης μνήμης. Αυτά τα τέλη καταβάλλονται σε ποσά του κρυπτονομίσματος του Ethereum, το ETH και λέγονται

gas.

Εν συνεχεία, αυτά τα τέλη συναλλαγής συλλέγονται από τους κόμβους που επικυρώνουν το δίκτυο,τους γνωστούς miners. Αυτοί οι miners είναι κόμβοι στο δίκτυο Ethereum που λαμβάνουν, διαδίδουν, επαληθεύουν και εκτελούν συναλλαγές. Οι miners ομαδοποιούν έπειτα τις συναλλαγές - που περιλαμβάνουν ενημερώσεις για την κατάσταση λογαριασμών στο Ethereum - σε blocks και ανταγωνίζονται ο ένας τον άλλο για το ποιό θα είναι το επόμενο block που θα προστεθεί το Blockchain. Οι miners ανταμείβονται με ETH για κάθε επιτυχημένο block που προσθέτουν. Η αμοιβή αυτή είναι το gas που αναφέραμε. Αυτό παρέχει το οικονομικό κίνητρο για τους ανθρώπους να αφιερώσουν υλικό (CPU) και μνήμη στο δίκτυο Ethereum. Οι miners, όπως έχουμε αναφέρει σε προηγούμενο κεφάλαιο, είναι επιφορτισμένοι με την επίλυση ενός πολύπλοκου μαθηματικού προβλήματος προκειμένου να εξορύξουν με επιτυχία ένα block, το Proof of Work.

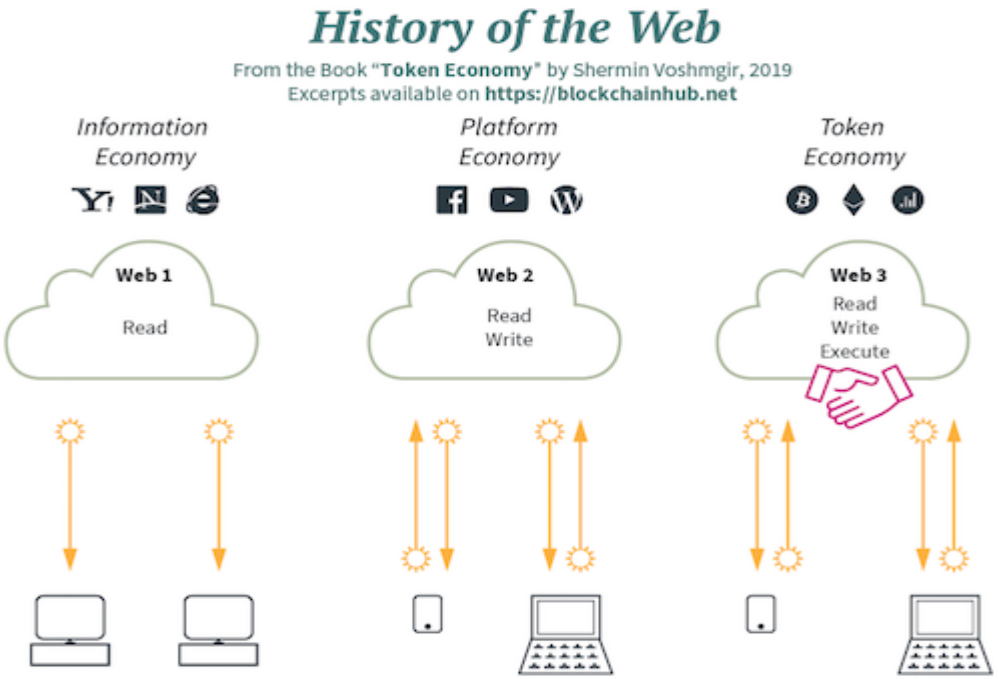
3.2 Web3

Στο Διαδίκτυο όπως είναι διαμορφωμένο στις μέρες μας δεν έχουμε τον πλήρη έλεγχο των δεδομένων μας. Τριάντα χρόνια μετά τη μαζική υιοθέτηση του Διαδικτύου και οι αρχιτεκτονικές δεδομένων μας εξακολουθούν να βασίζονται στην έννοια των αυτόνομων υπολογιστών, όπου τα δεδομένα αποθηκεύονται κεντρικά, διαχειρίζονται σε έναν διακομιστή και αποστέλλονται ή ανακτώνται από έναν πελάτη. Κάθε φορά που αλληλεπιδρούμε μέσω του Διαδικτύου, αντίγραφα των δεδομένων μας αποστέλλονται στον διακομιστή ενός παρόχου υπηρεσιών και κάθε φορά που συμβαίνει αυτό, χάνουμε αυτόματα τον έλεγχο των δεδομένων μας. Ακόμα κι αν ζούμε σε έναν "συνδεδεμένο" κόσμο, με όλο και περισσότερες συσκευές να συνδέονται με το Διαδίκτυο - συμπεριλαμβανομένων των ρολογιών, των αυτοκινήτων, των τηλεοράσεων ακόμη και των ψυγείων - τα δεδομένα μας εξακολουθούν να αποθηκεύονται κεντρικά: στους υπολογιστές μας ή σε άλλες συσκευές, στο USB stick ή και ακόμη και στις υπηρεσίες Cloud. Αυτό εγείρει άμεσα ζητήματα εμπιστοσύνης. Το σύγχρονο Internet λοιπόν με την αρχιτεκτονική πελάτη-εξυπηρετητή που υποστηρίζει έχει πολλά σημεία αποτυχίας, όπως γίνεται αντιληπτό από τις επαναλαμβανόμενες παραβιάσεις δεδομένων των διαδικτυακών παρόχων υπηρεσιών. Παράγει επιπλέον υψηλό κόστος διαχείρισης εγγράφων, καθώς και δεν κρίνεται απόλυτα διαφανές κατά μήκος της αλυσίδας εφοδιασμού αγαθών και υπηρεσιών που παρέχει.

Το Web1.0 ή αλλιώς WWW αποτελεί την πρώτη μορφή του Διαδικτύου που έγινε διαδεδομένη στους χρήστες των ηλεκτρονικών υπολογιστών γύρω στο 1990. Σε αυτή τη μορφή του το Internet έδινε στους χρήστες μόνο τη δυνατότητα ανάγνωσης (read). Μια δεκαετία αργότερα και λόγω του Dot-com bubble (γνωστό και ως Internet bubble) το Διαδίκτυο "ωρίμασε" και έγινε πια προγραμματιζόμενο. Είδαμε την άνοδο του λεγόμενου Web2, το οποίο μας έφερε πλατφόρμες κοινωνικών μέσων και ηλεκτρονικού εμπορίου (Facebook,YouTube και Wikipedia). Η άνοδος αυτής της μορφής του διαδικτύου έδωσε τη δυνατότητα στους χρήστες τόσο της εγγραφής όσο και της ανάγνωσης (read-write). Το Web2 έφερε επανάσταση στις κοινωνικές αλληλεπιδράσεις, φέρνοντας τους παραγωγούς και τους καταναλωτές πληροφοριών, αγαθών και υπηρεσιών πιο κοντά, και μας επέτρεψε να απολαύσουμε τις αλληλεπιδράσεις P2P σε παγκόσμια κλίμακα, αλλά πάντα με έναν μεσάζοντα: μια πλατφόρμα που λειτουργεί ως αξιόπιστος διαμεσολαβητής μεταξύ δύο ατόμων που

δεν γνωρίζονται ούτε εμπιστεύονται ο ένας τον άλλον. Ενώ αυτές οι πλατφόρμες έχουν κάνει φανταστική δουλειά για τη δημιουργία μιας οικονομίας P2P με ένα εξελιγμένο επίπεδο ανακάλυψης περιεχομένου , υπαγορεύουν όλους τους κανόνες των συναλλαγών και ελέγχουν όλα τα δεδομένα των χρηστών τους.

Το Blockchain φαίνεται να είναι ο κινητήριος μοχλός του Διαδικτύου επόμενης γενιάς, το οποίο είναι γνωστό ως Web3. Το Blockchain επαναπροσδιορίζει τον τρόπο αποθήκευσης και διαχείρισης των δεδομένων στο Διαδίκτυο. Παρέχει ένα μοναδικό σύνολο δεδομένων το οποίο διαχειρίζεται συλλογικά. Αυτό το μοναδικό σύνολο δεδομένων επιτρέπει για πρώτη φορά ένα επίπεδο διακανονισμού τιμών για το Διαδίκτυο. Μας επιτρέπει να στέλνουμε αρχεία με τρόπο που προστατεύεται από αντιγραφή, επιτρέποντας πραγματικές συναλλαγές P2P χωρίς μεσάζοντες. Στο Web3, τα δεδομένα αποθηκεύονται σε πολλά αντίγραφα ενός δικτύου P2P. Οι κανόνες διαχείρισης τυποποιούνται σε ένα πρωτόκολλο συναίνεσης και διασφαλίζονται με την ύπαρξη ομοφωνίας όλων των συμμετεχόντων στο δίκτυο. Το Blockchain, ως η ραχοκοκαλιά του Web3, επαναπροσδιορίζει τις δομές δεδομένων στο backend κομμάτι του Web, τώρα που ζούμε σε έναν "συνδεδεμένο" κόσμο. Εισάγει ένα επίπεδο διακυβέρνησης που τρέχει πάνω από το τρέχον Διαδίκτυο, το οποίο επιτρέπει σε δύο άτομα που δεν γνωρίζουν ή δεν εμπιστεύονται ο ένας τον άλλο να συνάψουν και να διευθετήσουν συμφωνίες μέσω του Διαδικτύου. Ενώ το Web2 ήταν μια επανάσταση στο frontend κομμάτι του διαδικτύου (UI), το Web3 είναι μια επανάσταση στο backend κομμάτι του. Μια επανάσταση που έγγειται στην λογική της ανάπτυξης κώδικα μέσω των έξυπνων συμβολαίων που ήρθαν στο φως με την έλευση του Ethereum, τα οποία αποτελούν τη δομική ύλη για την ανάπτυξη αποκεντρωμένων εφαρμογών (Dapps).



Σχήμα 3.1– History of the Web

3.3 Βασικά χαρακτηριστικά

Αφού κάναμε μια εισαγωγή στο βασικό "σενάριο" που κρύβεται πίσω από το Ethereum και στις μορφές δικτύων Web3 που κάνουν σιγά σιγά την εμφάνιση τους, ας δούμε μερικά βασικά χαρακτηριστικά στη δομή του Ethereum.

3.3.1 World State

Το World State είναι ένα mapping (αντιστοίχηση) μεταξύ διευθύνσεων Ethereum και της κατάστασης λογαριασμού (δομή δεδομένων) που είδαμε στο κεφάλαιο 3.1. Αν και δεν είναι αποθηκευμένο στο Blockchain, θεωρείται ότι το World State διατηρείται μέσα σε ένα modified Merkle Patricia tree (trie).

Το modified Merkle Patricia tree ή trie παρέχει μια μόνιμη δομή δεδομένων για το mapping μεταξύ δυαδικών δεδομένων αυθαίρετου μήκους. Είναι ένας συνδυασμός Merkle Tree, που παρουσιάστηκε στην προηγούμενη ενότητα και Patricia Trie. Το Patricia Trie έχει το πλεονέκτημα ότι είναι πολύ πιο γρήγορο στην εύρεση δεδομένων, αλλά λιγότερο ασφαλές. Ενδεικτικά αναφέρουμε ότι το Patricia Trie είναι μια δομή δεδομένων, η οποία χρησιμοποιεί προθέματα, ώστε να κάνει την ανάκτηση των δεδομένων πολύ πιο γρήγορη. Οπότε, ένας συνδυασμός των δύο αυτών δομών δεδομένων προσφέρει μεγάλη ασφάλεια και ευελιξία. Ορίζεται με όρους μεταβλητής δομής δεδομένων για την αντιστοίχιση μεταξύ δυαδικών δεδομένων 256-bit και αυθαίρετων δυαδικών δεδομένων, που συνήθως εφαρμόζονται ως βάση δεδομένων. Ο πυρήνας του trie είναι η παροχή μιας αποκλειστικής τιμής που προσδιορίζει ένα δεδομένο σύνολο ζευγών κλειδιών-τιμών (key-value), που μπορεί να είναι είτε μια ακολουθία 32-byte είτε κενή ακολουθία. Οι ρίζες των δέντρων αυτών αντικατοπτρίζονται μέσα στα blocks (root hash) και το υπόλοιπο κομμάτι των δέντρων που δεν ανήκει στο Blockchain λέγεται state trie.

Προφανώς, καθώς αλλάζουν τα δεδομένα των λογαριασμών μέσω των συναλλαγών, αλλάζει και το World State. Αν μπορούσαμε να φανταστούμε το δίκτυο του Ethereum ως έναν αποκεντρωμένο υπολογιστή, το World state θα ήταν ο σκληρός του δίσκος. Οπότε, στο Ethereum δεν έχουμε όλα τα δεδομένα αποθηκευμένα μέσα στο Blockchain, αλλά πολλά είναι εξωτερικά, που, όμως, οι αλλαγές σε αυτά πραγματοποιούνται μόνο αν μια συναλλαγή που τα επηρεάζει, μπει στο Blockchain.

3.3.2 Transactions

Μια συναλλαγή (transaction ή συμβολικά T) είναι μια απλή κρυπτογραφικά υπογεγραμμένη οδηγία που κατασκευάστηκε από έναν χρήστη στο πεδίο εφαρμογής του Ethereum. Υπάρχουν δύο τύποι συναλλαγών:

1. Συναλλαγές μεταξύ ήδη υπαρχόντων λογαριασμών μέσω μηνυμάτων.
2. Δημιουργία ενός νέου λογαριασμού συμβολαίου, δηλαδή εκτέλεση ενός νέου smart contract.

Και οι δύο αυτοί τύποι συναλλαγών καθορίζονται από ένα συγκεκριμένο αριθμό πεδίων:
nonce: Αριθμός συναλλαγών που έχουν σταλεί από τον λογαριασμό που δημιούργησε τη συγκεκριμένη συναλλαγή, συμβολίζεται και ως T_n.
gasPrice: Αξία σε Wei (10^{-18} ETH) του gas που σπαταλήθηκε για να πληρωθούν τα υπολογιστικά κόστη εκτέλεσης της συναλλαγής, συμβολίζεται και ως T_p.

gasLimit: Η μέγιστη ποσότητα gas, που είναι δυνατό να χρησιμοποιηθεί για την εκτέλεση της συναλλαγής, συμβολίζεται και ως T_g.
to: Αν αφορά τις συναλλαγές μεταξύ ήδη υπαρχόντων λογαριασμών, το πεδίο αυτό είναι η 160-bit διεύθυνση του λογαριασμού στον οποίο θα μεταφερθούν τα χρήματα. Αυτή η διεύθυνση μπορεί να είναι και διεύθυνση έξυπνου συμβολαίου. Αν αφορά τη δημιουργία ενός νέου λογαριασμού συμβολαίου, το πεδίο αυτό είναι πάντα κενό. Συμβολίζεται ως T_t.
value: Αν αφορά τις συναλλαγές μεταξύ ήδη υπαρχόντων λογαριασμών, τότε το πεδίο αυτό είναι το ποσό σε Wei, που πρόκειται να μεταφερθεί. Αν η συναλλαγή αυτή αφορά την κλήση ενός συμβολαίου, τότε το πεδίο αυτό είναι το ποσό σε Wei, που θα πληρωθεί από τον λογαριασμό συμβολαίου, που λαμβάνει το μήνυμα. Αν αφορά τη δημιουργία ενός νέου λογαριασμού συμβολαίου, τότε το πεδίο αυτό είναι το ποσό σε Wei που θα προστεθεί στο υπόλοιπο του νέου συμβολαίου. Συμβολίζεται ως T_v.
v,r,s: Τιμές οι οποίες χρησιμοποιούνται στην κρυπτογραφημένη υπογραφή της συναλλαγής, ώστε να μπορεί να προσδιοριστεί ο λογαριασμός που έστειλε τη συναλλαγή. Συμβολίζεται ως T_w,T_r και T_s.
init: Το πεδίο αυτό αφορά αποκλειστικά τη δημιουργία ενός νέου λογαριασμού συμβολαίου. Περιέχει τον κώδικα EVM που χρησιμοποιήθηκε για την αρχικοποίηση του έξυπνου συμβολαίου, αποθηκευμένο σε ένα byte διάνυσμα απεριόριστου μεγέθους. Συμβολίζεται ως T_i.
data: Το πεδίο αυτό αφορά αποκλειστικά τις συναλλαγές μεταξύ ήδη υπαρχόντων λογαριασμών. Περιέχει τα δεδομένα εισόδου της συναλλαγής, αποθηκευμένα σε ένα byte διάνυσμα απεριόριστου μεγέθους. Συμβολίζεται ως T_d.

Όταν μια συναλλαγή εκτελεστεί με επιτυχία και το block, στο οποίο περιέχεται, γίνει μέρος του Ethereum Blockchain, δημιουργείται μία απόδειξη συναλλαγής (transaction receipt). Επίσης υπάρχουν δύο Modified Merkle Patricia Trees, τα οποία δημιουργούνται για τις συναλλαγές. Τα δέντρα αυτά είναι το transaction trie και το transaction receipt trie και δημιουργούνται για κάθε block της αλυσίδας ξεχωριστά. Το transaction trie περιέχει όλες τις πληροφορίες για τις συναλλαγές του συγκεκριμένου block που αναφέρθηκαν παραπάνω. Η τιμή κατακερματισμού της ρίζας του δέντρου (hash root) αυτού αποτελεί πεδίο του block, όπως θα δούμε παρακάτω. Το transaction receipt trie περιέχει αποδείξεις των συναλλαγών που συμπεριλήφθηκαν στο συγκεκριμένο block. Η τιμή κατακερματισμού της ρίζας του δέντρου αυτού (hash root) αποτελεί, επίσης, πεδίο του block.

3.3.3 Blocks

Τα blocks στο Ethereum είναι μια συλλογή σχετικών πληροφοριών (γνωστή ως κεφαλίδα του block) H, μαζί με πληροφορίες που αντιστοιχούν στις περιλαμβανόμενες συναλλαγές του block T και ένα σύνολο άλλων κεφαλίδων από blocks, U, που έχουν τον ίδιο "γονέα" με αυτόν του παρόντος block (τέτοια block είναι γνωστά ως ommers). Η κεφαλίδα του block B_H στο Ethereum περιέχει πολλά παραπάνω πεδία από την παραδοσιακή δομή που έχει ένα block σε ένα Blockchain άλλου κρυπτονομίσματος. Αυτά είναι τα παρακάτω:

parentHash: Ο κατακερματισμός Keccak 256-bit της κεφαλίδας του "γονέα" του παρόντος block, στο σύνολό του, συμβολίζεται και ως H_p.

ommersHash: Ο κατακερματισμός Keccak 256-bit του τμήματος της λίστας των ommers blocks (που έχουν τον ίδιο γονέα) αυτού του block. συμβολίζεται και ως H_o.

beneficiary: Η διεύθυνση 160-bit στην οποία έχουν καταβληθεί όλα τα τέλη για επιτυχή εξόρυξη αυτού του block, δηλαδή η διεύθυνση του miner. Συμβολίζεται και ως H_c.

stateRoot: Ο κατακερματισμός Keccak 256-bit του κόμβου-ρίζας του state trie, αφού όλες οι συναλλαγές εκτελέστηκαν και οριστικοποιήθηκαν. Συμβολίζεται και ως H_r.

transactionsRoot: Ο κατακερματισμός Keccak 256-bit του κόμβου-ρίζας του transaction trie, που όπως είπαμε και προηγουμένως, περιέχει όλες τις συναλλαγές που αντιστοιχούν στο τρέχων block. Συμβολίζεται και ως H_t.

receiptsRoot: Ο κατακερματισμός Keccak 256-bit του κόμβου-ρίζας του transaction receipt trie, που όπως είπαμε και προηγουμένως, περιέχει όλες τις αποδείξεις συναλλαγών που αντιστοιχούν στο τρέχων block. Συμβολίζεται και ως H_e.

logsBloom: Πρόκειται για ένα φίλτρο, που λέγεται Bloom, το οποίο αποτελείται από πληροφορίες ευρετηρίου (διεύθυνση καταγραφής και θέματα καταγραφής), από τις οποίες αποτελούνται οι διάφορες συναλλαγές του τρέχοντος block. Συμβολίζεται και ως H_b.

difficulty: Μια βαθμωτή τιμή που αντιστοιχεί στο επίπεδο δυσκολίας αυτού του block. Αυτό μπορεί να υπολογιστεί από το επίπεδο δυσκολίας του προηγούμενου μπλοκ και το timestamp του τρέχοντος block. Όσο μεγαλύτερο είναι το difficulty, τόσο περισσότερες δοκιμές nonce πρέπει να γίνουν, ώστε να φτάσει ένας miner στο επιθυμητό αποτέλεσμα. Συμβολίζεται και ως H_d.

number: Μια βαθμωτή τιμή ίση με τον αριθμό των προηγούμενων block στην αλυσίδα. Το genesis block (block 0) έχει το πεδίο αυτό ίσο με 0. Συμβολίζεται και ως H_i.

gasLimit: Μια βαθμωτή τιμή ίση με το όριο δαπάνης gas ανά block, συμβολίζεται και ως H_l.

gasUsed: Μια βαθμωτή τιμή ίση με το συνολικό gas που χρησιμοποιήθηκε για συναλλαγές σε αυτό το block, συμβολίζεται και ως H_g.

timestamp: Μια βαθμωτή τιμή ίση με την έξοδο της εντολής time του Unix, κατά τη δημιουργία του block. Συμβολίζεται και ως H_s.

extraData: Ένας πίνακας αυθαίρετου μεγέθους byte που περιέχει δεδομένα σχετικά με αυτό το block. Πρέπει να είναι μεγέθους 32 byte ή λιγότερα. Συμβολίζεται και ως H_x.

mixHash: Ένας κατακερματισμός 256-bit ο οποίος, σε συνδυασμό με το nonce παρακάτω, αποδεικνύει ότι έχει πραγματοποιηθεί επαρκή σύνολο υπολογισμών σε αυτό το block. Συμβολίζεται και ως H_m.

nonce: Μια τυχαία τιμή 64-bit η οποία, σε συνδυασμό με το mixHash, αποδεικνύει ότι έχει πραγματοποιηθεί επαρκή σύνολο υπολογισμών σε αυτό το block. Η τιμή nonce χρησιμοποιείται από τους miners και αλλάζει συνεχώς μέχρις ότου φτάσει ένας miner να βρει μια τιμή μικρότερη της τιμής-στόχου (η τιμή-στόχος καθορίζεται μέσω ενός αλγορίθμου ο οποίος ανάλογα με τη κινητικότητα του δικτύου αλλάζει και την τιμή). Συμβολίζεται και ως H_n.

Τα άλλα δύο στοιχεία του block, όπως είπαμε, είναι απλά μια λίστα των κεφαλίδων των ommer blocks (με την ίδια μορφή όπως τα παραπάνω) B_U και μια σειρά συναλλαγών B_T. Επομένως, μπορούμε να αναφερόμαστε σε ένα block B ως εξής:

$$B=(B_H, B_T, B_U)$$

3.3.4 Έξυπνα Συμβόλαια

Παρόλο που ο όρος "έξυπνο συμβόλαιο" (smart contract) έχει αναπτυχθεί με την έλευση της τεχνολογίας του Blockchain, σαν όρος εμφανίστηκε στην πραγματικότητα πριν από περίπου τριάντα χρόνια. Ο Nick Szabo, επιστήμονας υπολογιστών και κρυπτογράφος, έγραψε ένα άρθρο για τα έξυπνα συμβόλαια το 1995. Η ιδέα που παρουσίασε ο Szabo αντιστοιχεί ακριβώς σε αυτό που προσφέρουν τα έξυπνα συμβόλαια σήμερα, συμπεριλαμβανομένης της εφαρμογής και αποθήκευσης τους σε ένα κατανεμημένο καθολικό. Πιο συγκεκριμένα τα καθόρισε ως «ένα σύνολο υποσχέσεων, που καθορίζονται σε ψηφιακή μορφή, συμπεριλαμβανομένων των πρωτοκόλλων στα οποία τα μέρη εκτελούν τις άλλες υποσχέσεις».

Στις μέρες μας, όλη η διαδικασία δημιουργίας της πλατφόρμας του Ethereum βασίστηκε στο πώς θα μπορεί να αποτελέσει τη βάση για τη δημιουργία αποκεντρωμένων εφαρμογών. Γέφυρα του Blockchain για τη δημιουργία αποκεντρωμένων εφαρμογών αποτέλεσε η χρήση των έξυπνων συμβολαίων.
Τα έξυπνα συμβόλαια είναι στην ουσία προγράμματα κώδικα τα οποία ενεργοποιούνται και εκτελούνται όταν ικανοποιηθούν συγκεκριμένες συνθήκες αυτόματα. Τα έξυπνα συμβόλαια αυτά ενεργοποιούνται αυτόματα από τους υπολογιστές και οι κινήσεις καταγράφονται στο Blockchain καθιστώντας την πληροφορία αμετάβλητη και αδιαμφισβήτητη. Αν και τα συμβόλαια αυτά υπάρχουν εδώ και πολλά χρόνια στην πιο απλή μορφή τους, όπως για παράδειγμα στην περίπτωση ενός αυτόματου πωλητή, η ενσωμάτωσή της λειτουργίας τους μέσα από την τεχνολογία Blockchain τους δίνει νέες δυνατότητες.
Τα έξυπνα συμβόλαια σχεδιάζονται και υλοποιούνται σε ένα Blockchain, και ως εκ τούτου κληρονομούν ορισμένες από τις ιδιότητες του:

i. **Είναι αμετάβλητα** (immutable), πράγμα που σημαίνει ότι ένα έξυπνο συμβόλαιο δεν μπορεί ποτέ να αλλάξει και κανείς δεν μπορεί να παραβιάσει ή να παραβεί ένα συμβόλαιο.
ii. **Είναι αποκεντρωμένα** (decentralized), δηλαδή όταν κάποιος θέλει να εκτελέσει μια συνάρτηση ενός έξυπνου συμβολαίου, θα πρέπει, για να γίνουν αλλαγές στην κατάστασή του, η πλειοψηφία των miners του Ethereum, να εγκρίνει τη συναλλαγή.

Αυτά τα δύο χαρακτηριστικά προσφέρουν ασφάλεια, καθώς αν συμφωνηθεί κάτι, ούτε αλλάζει ποτέ, ούτε τα κριτήρια που έχουν οριστεί μπορούν να προσπεραστούν, καθώς δε θα υπάρξει έγκριση.
Τα έξυπνα συμβόλαια στο Ethereum συντάσσονται συνήθως σε μια γλώσσα υψηλού επιπέδου, όπως η Solidity. Ωστόσο, για να εκτελεστούν, πρέπει να μεταγλωττιστούν πρώτα σε μια χαμηλού επιπέδου γλώσσα (bytecode) που εκτελείται στην EVM (Ethereum Virtual Machine). Μόλις μεταγλωττιστούν, αναπτύσσονται στην πλατφόρμα Ethereum μέσω μιας ειδικής συναλλαγής δημιουργίας συμβολαίου.

Κάθε συμβόλαιο προσδιορίζεται από μια διεύθυνση Ethereum, η οποία προέρχεται από τη συναλλαγή δημιουργίας συμβολαίου ως συνάρτηση του λογαριασμού προέλευσης. Η διεύθυνση Ethereum ενός smart contract μπορεί να χρησιμοποιηθεί σε μια συναλλαγή ως παραλήπτης, στέλνοντας χρήματα στο συμβόλαιο ή καλώντας μια από τις λειτουργίες του συμβολαίου. Σε αντίθεση με τους EOA's (Externally Owned Accounts), δεν υπάρχουν κλειδιά που σχετίζονται με έναν λογαριασμό που δημιουργήθηκε για ένα νέο έξυπνο συμβόλαιο. Ως δημιουργός συμβολαίου, δεν έχετε ειδικά προνόμια σε επίπεδο πρωτοκόλλου (αν και μπορείτε να τα κωδικοποιήσετε ρητά στο έξυπνο συμβόλαιο). Σίγουρα δεν λαμβάνετε το ιδιωτικό κλειδί για το λογαριασμό της σύμβασης, το οποίο στην πραγματικότητα δεν υπάρχει -μπορούμε να

πούμε ότι οι λογαριασμοί έξυπνων συμβολαίων είναι τα ίδια τα έξυπνα συμβόλαια.

Για να μπορέσει κάποιος να εκτελέσει ή να καλέσει ένα έξυπνο συμβόλαιο θα πρέπει, να πληρώσει ένα ποσό, ίσο με αυτό που θα δαπανηθεί σε υπολογιστική ενέργεια από τους miners για την πραγματοποίηση του συγκεκριμένου σκοπού. Το ποσό αυτό, όπως έχουμε αναφέρει και προηγουμένως, είναι το λεγόμενο gas και η ποσότητα του gas η οποία σπαταλάται για τη χρήση κάθε εντολής της EVM είναι καθορισμένη από τους δημιουργούς του Ethereum.

Κάθε χρήστης, ο οποίος δημιουργεί ή εκτελεί ένα έξυπνο συμβόλαιο αποφασίζει τι ποσό θα πληρώσει ανά gas (gas price) που θα χρησιμοποιηθεί για τη συναλλαγή αυτή, αναλόγως με το πόσο γρήγορα θέλει να εκτελεστεί. Αυτό συμβαίνει, διότι, οι miners επιδιώκουν όσο το δυνατόν μεγαλύτερο κέρδος λόγω της υπολογιστικής ενέργειας που σπαταλούν, οπότε προφανώς θα εισάγουν ευκολότερα στο block τους εκτελέσεις συμβολαίων με μεγάλη τιμή gas. Επίσης, ο χρήστης καθορίζει και τη μέγιστη ποσότητα gas (gas limit), την οποία είναι διατεθειμένος να σπαταλήσει για την εκτέλεση ενός τέτοιου συμβολαίου.

Ο χρήστης πληρώνει ένα κόστος εκτέλεσης συναλλαγών ίσο με το γινόμενο: $gas\,price \times gas\,limit$. Αν ικανοποιηθούν τα κριτήρια που το συμβόλαιο ορίζει και ο miner το εξορύξει επιτυχώς, τότε, αν δε χρησιμοποιηθεί όλο το gas, που ορίζεται από το gas limit, τα χρήματα που απομένουν επιστρέφονται στον χρήστη. Αν, όμως, δεν ικανοποιηθούν όλα τα κριτήρια του συμβολαίου, σπαταλάται όλο το ποσό, ως επιβάρυνση στον χρήστη που έκανε εσφαλμένη κλήση συμβολαίου. Αυτή η ενδεχόμενη απώλεια σημαντικών ποσών από τη πλευρά των χρηστών, καθιστά τα έξυπνα συμβόλαια ακόμη πιο ασφαλή, καθώς δεν επιτρέπεται η αβέβαιη κλήση ενός συμβολαίου.

3.3.5 Mining

Το mining, ή αλλιώς εξόρυξη, στο Ethereum είναι η διαδικασία δημιουργίας των blocks που θα προστεθούν στο Blockchain του Ethereum. Το Ethereum, όπως το Bitcoin, χρησιμοποιεί (προς το παρόν) τον αλγόριθμο συναίνεσης Proof of Work. Η εξόρυξη είναι η "καρδιά" του αλγορίθμου αυτού. Οι miners - υπολογιστές που χρησιμοποιούν λογισμικό - χρησιμοποιούν το χρόνο και τη δύναμη υπολογισμού τους για την επεξεργασία συναλλαγών και την παραγωγή blocks. Ας δούμε πως λειτουργεί το mining στο Ethereum:

Ένας χρήστης δημιουργεί και υπογράφει μια αίτηση συναλλαγής με το ιδιωτικό του κλειδί. Ο χρήστης μεταδίδει το αίτημα συναλλαγής σε ολόκληρο το δίκτυο Ethereum από κάποιο κόμβο. Μόλις ακούσει για το νέο αίτημα συναλλαγής, κάθε κόμβος στο δίκτυο Ethereum προσθέτει το αίτημα στο τοπικό mempool του, μια λίστα με όλα τα αιτήματα συναλλαγών που έχουν ακούσει τα οποία δεν έχουν ακόμη εγκριθεί για εισαγωγή στο Blockchain. Έτσι, ένας κόμβος εξόρυξης συγκεντρώνει αρκετές δεκάδες ή εκατοντάδες αιτήσεις συναλλαγών σε ένα πιθανό block, με τρόπο που μεγιστοποιούνται τα τέλη συναλλαγής που θα κερδίσει. Ο κόμβος εξόρυξης τότε: Επαληθεύει την εγκυρότητα κάθε αιτήματος συναλλαγής και στη συνέχεια εκτελεί τον κώδικα του αιτήματος, τροποποιώντας την τρέχουσα κατάσταση του τοπικού του αντιγράφου της EVM. Ο miner επιβραβεύεται με ένα τέλος συναλλαγής για κάθε τέτοιο αίτημα συναλλαγής στον δικό του λογαριασμό. Έπειτα ξεκινά τη διαδικασία παραγωγής του «πιστοποιητικού απόδειξης εργασίας» του Proof of Work για το πιθανό μπλοκ, αφού όλα τα αιτήματα συναλλαγών στο block έχουν επαληθευτεί και εκτελεστεί στο τοπικό αντίγραφο του EVM. Τελικά, όταν ένας miner θα ολοκληρώσει την παραγωγή ενός πιστοποιητικού για ένα block που περιλαμβάνει το

συγκεκριμένο αίτημα συναλλαγής μας, θα μεταδώσει το ολοκληρωμένο block, το οποίο περιλαμβάνει το πιστοποιητικό και ένα άθροισμα ελέγχου της νέας κατάστασης της EVM. Τότε οι άλλοι κόμβοι ακούνε για το νέο block. Επαληθεύουν το πιστοποιητικό, εκτελούν όλες τις συναλλαγές στο block (συμπεριλαμβανομένης της συναλλαγής που μεταδόθηκε αρχικά από τον χρήστη μας) και επαληθεύουν ότι το άθροισμα ελέγχου της νέας κατάστασης EVM μετά την εκτέλεση όλων των συναλλαγών ταιριάζει με το άθροισμα ελέγχου της κατάστασης που διεκδικείται από το block που προτείνει ο miner . Μόνο τότε αυτοί οι κόμβοι προσαρτούν αυτό το block στην αλυσίδα τους και αποδέχονται τη νέα κατάσταση της EVM ως κανονική κατάσταση. Κάθε κόμβος αφαιρεί όλες τις συναλλαγές στο νέο block από το τοπικό mempool των ανεκπλήρωτων αιτημάτων συναλλαγής τους. Οι νέοι κόμβοι που συμμετέχουν στο δίκτυο κατεβάζουν όλα τα blocks στη σειρά, συμπεριλαμβανομένου του block που περιέχει τη συναλλαγή που τους ενδιαφέρει. Αρχικοποιούν ένα τοπικό αντίγραφο EVM και στη συνέχεια περνούν στη διαδικασία εκτέλεσης κάθε συναλλαγής σε κάθε block πάνω από το τοπικό αντίγραφο του EVM, επαληθεύοντας τα αθροίσματα ελέγχου κατάστασης σε κάθε block.
Προς το παρόν (Νοέμβριος 2020), όταν ένα block εξορύσσεται με επιτυχία στο Ethereum Blockchain, ο miner λαμβάνει 3 ETH ως ανταμοιβή, ενώ στο ποσό αυτό προστίθενται οι φόροι των συναλλαγών, είτε αυτές είναι κανονικές συναλλαγές, είτε είναι εκτελέσεις έξυπνων συμβολαίων.

3.4 Αποκεντρωμένες εφαρμογές

Οι αποκεντρωμένες εφαρμογές ή αλλιώς Dapps στο Ethereum είναι διαδικτυακές εφαρμογές (web applications) που υποστηρίζονται από έξυπνα συμβόλαια. Αντί να χρησιμοποιούν έναν κεντρικό διακομιστή ή μια βάση δεδομένων, αυτές οι εφαρμογές βασίζονται στο Blockchain ως backend για τη λογική και την αποθήκευση του προγράμματος. Αυτό οδηγεί σε δυνητικά ασταμάτητες εφαρμογές καθώς ο καθένας μπορεί να αναπτύξει ένα αντίγραφο του frontend και να το συνδέσει ελεύθερα στο δημόσιο δίκτυο Ethereum.

Μερικά αξιοσημείωτα χαρακτηριστικά των Dapps είναι τα εξής:

- **Ανοιχτή πηγή (Open source)**: Διέπονται από αυτονομία και όλες οι αλλαγές πρέπει να αποφασίζονται με τη συναίνεση ή την πλειοψηφία των χρηστών της. Η βάση κώδικα θα πρέπει να είναι διαθέσιμη για έλεγχο.
- **Αποκεντρωτικός χαρακτήρας:** Όλα τα αρχεία της λειτουργίας της εφαρμογής πρέπει να αποθηκεύονται σε ένα δημόσιο και αποκεντρωμένο Blockchain για να αποφευχθούν παγίδες συγκεντρωτισμού.
- **Ύπαρξη κινήτρων:** Οι επικυρωτές του Blockchain θα πρέπει να ενθαρρύνονται ανταμείβοντας τους με κρυπτονομίσματα.
- **Ύπαρξη πρωτοκόλλου:** Η κοινότητα εφαρμογών πρέπει να συμφωνήσει σε έναν κοινό κρυπτογραφικό αλγόριθμο συναίνεσης. Για παράδειγμα, το Ethereum χρησιμοποιεί επί του παρόντος τον Proof of Work.

Στην παρούσα διπλωματική έχει επιλεχτεί το περιβάλλον του Ethereum για την ανάπτυξη μιας τυπικής αποκεντρωμένης εφαρμογής ηλεκτρονικής ψηφοφορίας (Προεδρικές Εκλογές ΗΠΑ 2020). Στα επόμενα κεφάλαια θα γίνει μια προσπάθεια παρουσίασης αυτής της εφαρμογής. Ας δούμε τώρα κάποια αναγκαία εργαλεία και τεχνολογίες που έπρεπε να χρησιμοποιηθούν για να γίνει δυνατή η ανάπτυξη αυτής της εφαρμογής.

4. Εργαλεία και Τεχνολογίες

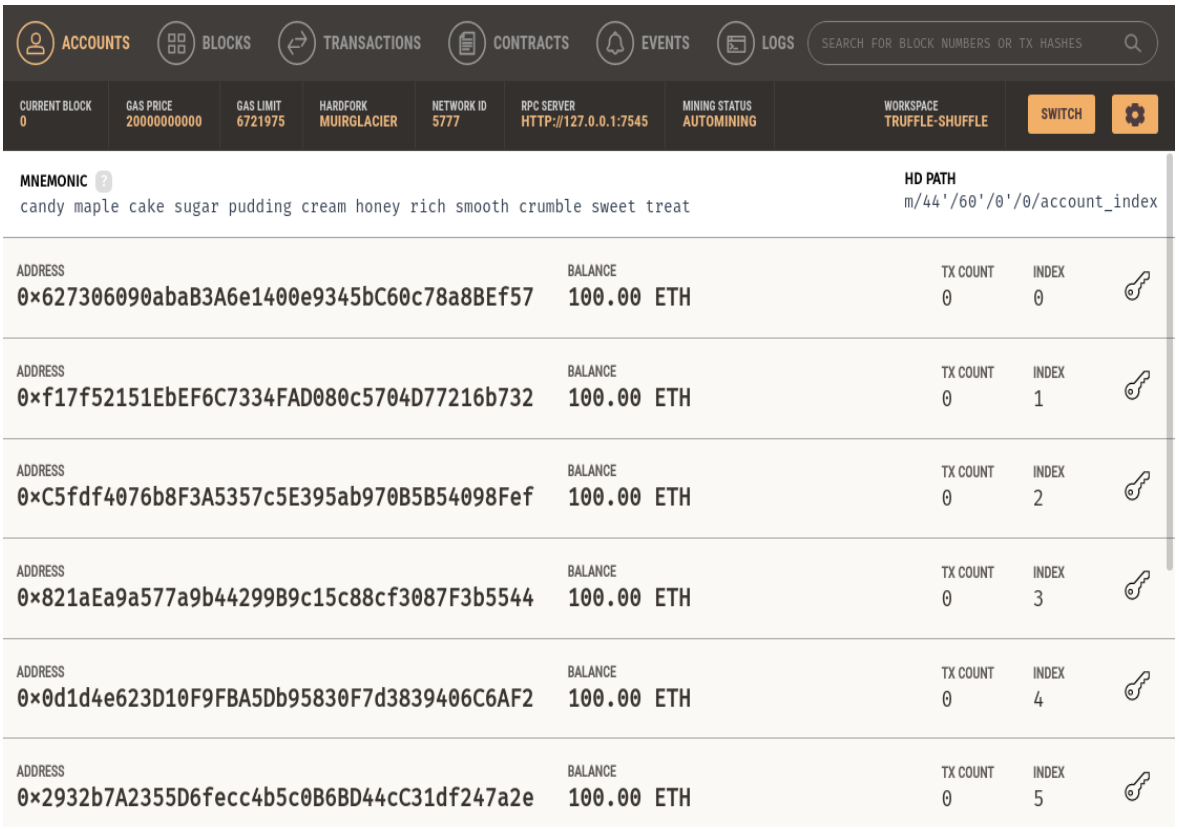
Για την ανάπτυξη και λειτουργία αποκεντρωμένων εφαρμογών στο Ethereum υπάρχουν πολλά διαφορετικά εργαλεία και τεχνολογίες που μπορεί ο καθένας να χρησιμοποιήσει. Όλα αυτά τα εργαλεία συνεχώς εξελίσσονται και αναβαθμίζονται με σκοπό την πιο ομαλή λειτουργία της Dapp που υποστηρίζουν. Θα αναπτύξουμε σε αυτό το κεφάλαιο όλα τα απαραίτητα εργαλεία, τεχνολογίες και γλώσσες προγραμματισμού που επιλέξαμε εμείς να χρησιμοποιήσουμε ώστε να δημιουργήσουμε την αποκεντρωμένη εφαρμογή της διπλωματικής εργασίας.

4.1 Truffle

Το Truffle είναι μια σουίτα πρωτοκόλλων ή αλλιώς ένα αναπτυξιακό περιβάλλον, πλαίσιο δοκιμών (testing framework) και μέσο επικοινωνίας με το Ethereum χρησιμοποιώντας την Ethereum Virtual Machine, που έχει σκοπό να βελτιώσει και να διευκολύνει τον τρόπο με τον οποίο ένας προγραμματιστής αλληλεπιδρά με αυτό. Χρησιμοποιώντας το Truffle ένας Blockchain προγραμματιστής έχει ένα μέσο το οποίο επικοινωνεί με το Ganache (θα δούμε στη συνέχεια τι είναι το Ganache) και καθίσταται δυνατή η μετατροπή των έξυπνων συμβολαίων, τα οποία είναι γραμμένα στην Solidity, στη γλώσσα που αντιλαμβάνεται το Blockchain. Μπορεί δηλαδή μέσω του Truffle να τα κάνει compile, να τα συνδέσει, να τα αναπτύξει (deploy) και να διαχειριστεί τον δυαδικό τους κώδικα. Επίσης, δίνεται η δυνατότητα στον προγραμματιστή να εκτελέσει όλα του τα συμβόλαια μαζί με μία εντολή αυτόματα . Το Truffle είναι ενσωματωμένο με τον npm (node package manager) και αναγνωρίζει τον φάκελο node_modules στο έργο της εφαρμογής μας εάν υπάρχει. Αυτό σημαίνει ότι είναι δυνατή η χρησιμοποίηση και διανομή έξυπνων συμβολαίων, dapps και βιβλιοθηκών μέσω της εντολής npm, καθιστώντας τον κώδικα του προγράμματος μας διαθέσιμο σε άλλους και τον κωδικό άλλων διαθέσιμο σε εμάς. Τέλος, να αναφερθεί ότι το Truffle δε χρησιμοποιείται στο πραγματικό Blockchain, αλλά αν συνδυαστεί με το Ganache και το web3 που θα εξεταστεί αμέσως μετά, δίνει τη δυνατότητα προσομοίωσης του κώδικα ενός έξυπνου συμβολαίου στο Ethereum Blockchain.

4.2 Ganache

Από τη σουίτα του Truffle επιλέχθηκε να χρησιμοποιηθεί ένα εργαλείο που δίνει τη δυνατότητα στο χρήστη να δημιουργήσει ένα τοπικό Ethereum Blockchain, το Ganache. Το Ganache κάνει αυτόματα όλες τις απαραίτητες ρυθμίσεις δικτύου που χρειάζεται (IP, θύρα δικτύου κλπ.) και δημιουργεί αυτόματα κάποιους λογαριασμούς χρηστών εφοδιασμένους με το ποσό των 100 ETH (*σχήμα 4.1*). Χρησιμοποιείται ώστε να δοκιμαστεί η λειτουργία των έξυπνων συμβολαίων ή της εφαρμογής που είναι χτισμένα πάνω στο Ethereum, πριν αυτά "ανέβουν" στο πραγματικό Blockchain. Δίνει τη δυνατότητα στον χρήστη να εισάγει όποιες επιλογές θέλει, ώστε να προσομοιώσει όπως ακριβώς θέλει τις εφαρμογές του.



Σχήμα 4.1– Ganache

4.3 Web3.js

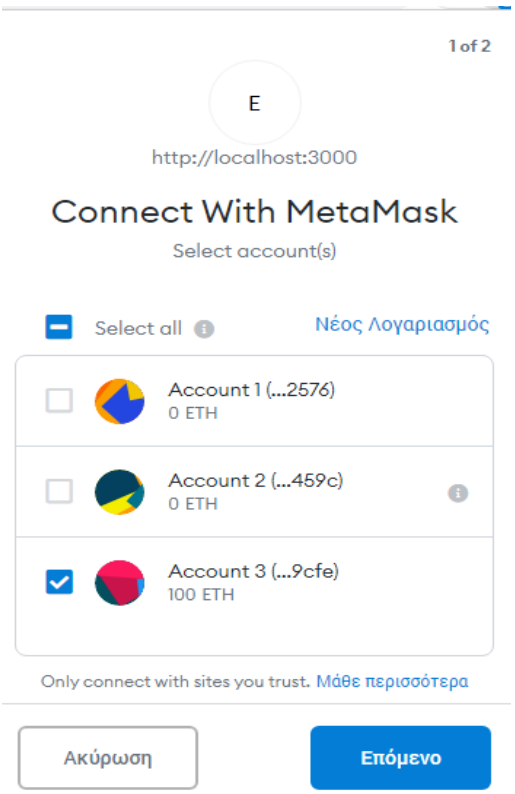
Το web3.js είναι το πιο σημαντικό npm πακέτο που χρησιμοποιεί η αποκεντρωμένη εφαρμογή μας. Το web3.js είναι μια συλλογή βιβλιοθηκών που επιτρέπει την αλληλεπίδραση με έναν τοπικό ή απομακρυσμένο κόμβο Ethereum χρησιμοποιώντας μια HTTP ή IPC σύνδεση ή το πρωτόκολλο WebSocket. Στην πραγματικότητα είναι η διεπαφή επικοινωνίας που χρειάζεται η εφαρμογή μας, για να επικοινωνήσει η JavaScript με το Blockchain. Όταν λέμε JavaScript εννοούμε το κομμάτι της Dapp που αποτελεί το frontend κομμάτι της. Έτσι μέσω του web3.js ο frontend κώδικας μας μπορεί να αξιοποιήσει αντικείμενα που «ζουν» μέσα στο Blockchain, όπως τα έξυπνα συμβόλαια (smart contracts), τα δεδομένα τους, τις συναρτήσεις τους, τις διευθύνσεις και τα υπόλοιπα λογαριασμών Ethereum.

4.4 Solidity

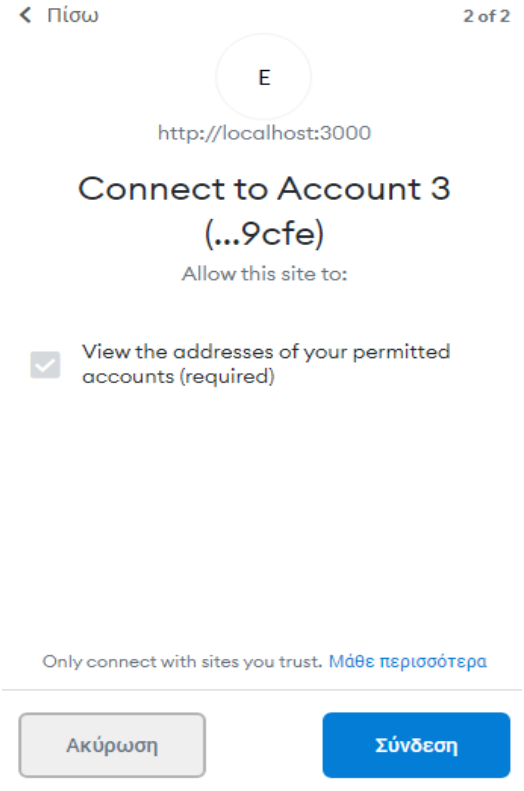
Η Solidity είναι μία αντικειμενοστραφής υψηλού επιπέδου γλώσσα προγραμματισμού που εκτελείται στο Ethereum Virtual Machine (EVM) και έχει δημιουργηθεί για να μεγεθύνει τις δυνατότητες της ιδεατής αυτής μηχανής. Το αντικείμενο (object) στην Solidity δεν είναι το αντικείμενο όπως το έχουμε στο μυαλό μας για την Java, αλλά το ίδιο το έξυπνο συμβόλαιο. Χρησιμοποιείται κυρίως για την δημιουργία τέτοιων έξυπνων συμβολαίων στο Ethereum Blockchain. Ο ίδιος ο πηγαίος κώδικας της γλώσσας του Ethereum είναι γραμμένος σε Solidity. Έχει παρόμοιο συντακτικό με αυτό της JavaScript, οπότε είναι εύκολα κατανοήσιμη από έναν μεγάλο αριθμό προγραμματιστών. Είναι μία στατικού τύπου γλώσσα. Υποστηρίζει την κληρονομικότητα με παρόμοιο τρόπο με άλλες γλώσσες προγραμματισμού (π.χ. C++).

4.5 MetaMask

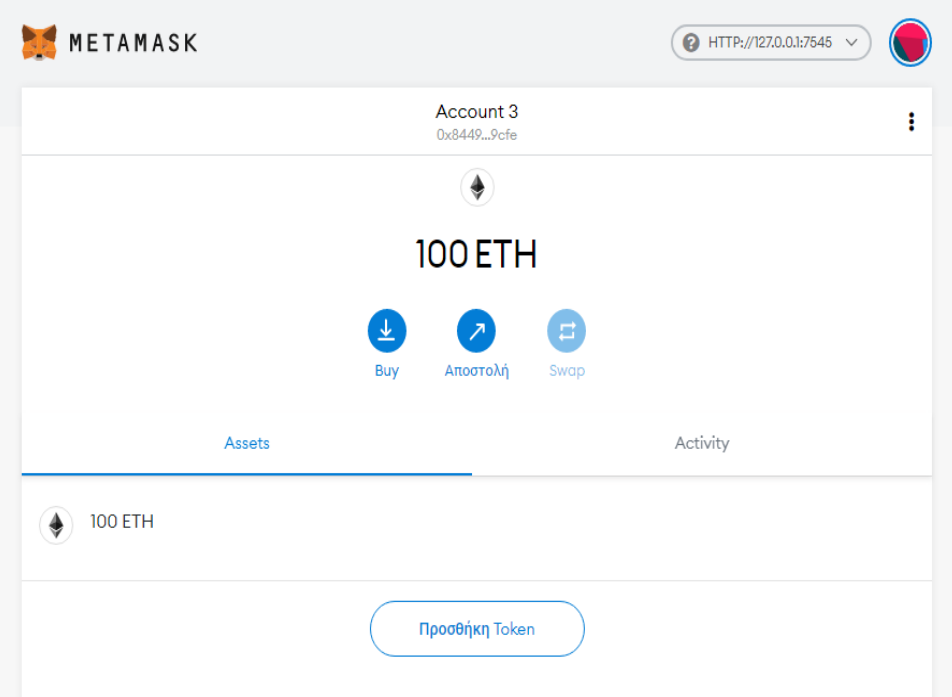
Το MetaMask είναι ένα plugin διαθέσιμο για τους φυλλομετρητές Google Chrome, Mozilla Firefox, Brave και Opera. Το MetaMask είναι στην ουσία μία "γέφυρα" η οποία δίνει στον browser πρόσβαση στις αποκεντρωμένες εφαρμογές (Dapps), που για την λειτουργία τους βασίζονται στο δίκτυο του Ethereum. Το μεγάλο πλεονέκτημα που προσφέρει είναι το γεγονός ότι με την χρήση του, η πρόσβαση στις εφαρμογές αυτές, δεν απαιτεί την εκτέλεση του πλήρους Ethereum κόμβου στο μηχάνημα του χρήστη, κάτι που κάνει ο φυλλομετρητής Mist και απαιτεί ειδικές γνώσεις για την διαχείριση του. Σύμφωνα με τους δημιουργούς του, ο σκοπός του είναι να κάνει το Ethereum προσβάσιμο σε όσο το δυνατόν περισσότερο κόσμο. Αυτό, έπειτα από την προσωπική μου χρήση, το πετυχαίνει σε μεγάλο βαθμό καθώς η εγκατάστασή του, η σύνδεση με το Blockchain και η αλληλεπίδραση με αυτό γίνεται αρκετά εύκολα. Συμπεριλαμβάνει μία ασφαλή κρύπτη και παρέχει στον χρήστη μία διεπαφή, μέσω της οποίας αυτός μπορεί να συνδέεται στο Ethereum Blockchain (*σχήματα 4.2.1,4.2.2*),να διαχειρίζεται τους Ethereum λογαριασμούς του, ώστε να αλληλοεπιδρά με τις ιστοσελίδες (*σχήμα 4.3*), να στέλνει ή και να υπογράφει συναλλαγές και δεδομένα (*σχήμα 4.4*).



Σχήμα 4.2.1-Επιλογή λογαριασμού για σύνδεση στο Ethereum Blockchain



Σχήμα 4.2.2-Επιβεβαίωση σύνδεσης λογαριασμού



Σχήμα 4.3-Διαχείριση λογαριασμού χρήστη

Σχήμα 4.4-Επιβεβαίωση/απόρριψη συναλλαγής

4.6 NPM

Το NPM – Node Package Manager είναι ένα πρόγραμμα διαχείρισης πακέτων της γλώσσας JavaScript και είναι το μεγαλύτερο μητρώο λογισμικού στον κόσμο. Διαχειριστής του είναι η εταιρία npm, Inc (θυγατρική της GitHub). Είναι το προκαθορισμένο πρόγραμμα διαχείρισης πακέτων του προγραμματιστικού περιβάλλοντος Node.js που θα αναφέρουμε αμέσως μετά. Εκτός από πακέτα, στο αρχείο του npm υπάρχουν και node modules τα οποία χρησιμοποιούνται στον frontend προγραμματισμό. Το πακέτο είναι ένα αρχείο ή ένα directory το οποίο περιγράφεται από ένα package.json, ενώ το module είναι ένα αρχείο ή ένα directory το οποίο φορτώνεται από το Node.js μέσω της εντολής require(). Αποτελείται από τρία διακριτά κομμάτια:

- Την ιστοσελίδα (https://www.npmjs.com/).
- Το Command Line Interface (CLI) μέσω του οποίου είτε λαμβάνονται τα διαθέσιμα πακέτα είτε δημοσιεύονται στο αρχείο πακέτα που ο χρήστης έχει δημιουργήσει. Το CLI τρέχει μέσω της γραμμής εντολών του χρήστη με χρήση της εντολής npm.
- Ένα αρχείο (registry), στο οποία είναι αποθηκευμένα τα JavaScript πακέτα και τα Node modules.

Είναι πολύ χρήσιμο στην κοινότητα ανάπτυξης λογισμικού, επειδή είναι δυνατή η ανταλλαγή έτοιμων πακέτων, δηλαδή κώδικα που λύνει ένα συγκεκριμένο πρόβλημα πολύ καλά και επίσης έχει θεσπίσει προδιαγραφές τις οποίες ακολουθούν όλα τα πακέτα που δημοσιεύονται στο registry, έτσι ώστε να υπάρχει ένας όσο γίνεται ενιαίος τρόπος χρησιμοποίησής τους από τους προγραμματιστές. Ένα npm πακέτο που χρησιμοποιήθηκε είναι το git, στην έκδοση 2.29.2. Αυτό είναι ένα σύστημα ελέγχου εκδόσεων με έμφαση στην ταχύτητα, στην ακεραιότητα των δεδομένων και στην υποστήριξη για κατανεμημένες μη γραμμικές ροές εργασίας. Είναι απαραίτητο για την ορθή λειτουργία του πακέτου web3.js.

4.7 Node.js

Το Node.js είναι ένα ανοιχτού κώδικα (open source), cross-platform περιβάλλον ανάπτυξης και εκτέλεσης της γλώσσας προγραμματισμού JavaScript, κατάλληλο για εύκολη και γρήγορη ανάπτυξη διαδικτυακών εφαρμογών. Το Node.js έχει αναπτυχθεί πάνω στην μηχανή JavaScript V8, τον πυρήνα του Google Chrome, έξω όμως από το πρόγραμμα περιήγησης. Επίσης, το Node.js παρέχει μία μεγάλη βιβλιοθήκη από JavaScript modules, γεγονός που απλοποιεί την διαδικασία ανάπτυξης διαδικτυακών εφαρμογών. Δηλαδή το Node.js είναι και περιβάλλον εκτέλεσης, αλλά και βιβλιοθήκη της JavaScript. Ένα κομμάτι αυτής της βιβλιοθήκης είναι και το web3.js που αναφέραμε προηγουμένως. Μέσω του Node.js και του web3.js, μπορεί κάποιος να εντοπίσει ένα έξυπνο συμβόλαιο, δίνοντας δύο πληροφορίες: το Abi του συμβολαίου (δυαδική διεπαφή του) και την διεύθυνση του λογαριασμού του και έπειτα να επικοινωνήσει μαζί του.

4.8 Ανάπτυξη ιστοσελίδας

Για τις ανάγκες παρουσίασης της εργασίας ήταν αναγκαία η ανάπτυξη μιας ιστοσελίδας, μιας και η αποκεντρωμένη εφαρμογή ηλεκτρονικής ψηφοφορίας που δημιουργήθηκε είναι μια web-app. Μέσω της ιστοσελίδας φαίνεται η αποτελεσματική λειτουργία τόσο του backend όσο και του frontend κώδικα μας. Προφανώς

χρησιμοποιήθηκε η γλώσσα HTML (Hypertext Markup Language), η οποία είναι η γλώσσα που χρησιμοποιείται σε ολόκληρο τον κόσμο για την δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Η HTML σε συνδυασμό με κάποια Cascading Style Sheets (CSS) και την JavaScript ολοκλήρωσαν το κομμάτι της επαφής του χρήστη με την εφαρμογή μας.

4.8.1 HTML

Η γλώσσα HTML (Hypertext Markup Language) είναι η βασική γλώσσα σήμανσης για τις ιστοσελίδες και τα στοιχεία που την συνθέτουν τα αποτελούν δομικά στοιχεία όλων των ιστοσελίδων που συναντάμε στους web browsers. Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags) οι οποίες περικλείονται μέσα στα σύμβολα "<>" και (όχι όλες οι ετικέτες) </>. Τέτοιες ετικέτες μπορεί να είναι κείμενα, παράγραφοι, εικόνες, υπερσύνδεσμοι κλπ. Ο τρόπος που ο browser αξιοποιεί τις ετικέτες αυτές δεν είναι να τις εμφανίζει αυτούσιες, αλλά να τις χρησιμοποιεί για να παρουσιάσει το περιεχόμενο της σελίδας. Τέλος, μπορεί να υποβοηθηθεί από τεχνολογίες όπως η Cascading Style Sheets (CSS) και γλώσσες προγραμματισμού όπως η JavaScript. Στην εφαρμογή μας υλοποιούμε το συνδυασμό αυτών των τεχνολογιών, όπως θα φανεί και στη παρουσίαση του frontend κώδικα.

4.8.2 CSS

Η CSS (Cascading Style Sheets) είναι μια γλώσσα υπολογιστή και πιο συγκεκριμένα, είναι μια γλώσσα ύφους φύλλων και χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μια γλώσσα σήμανσης, συνήθως HTML (ή XTHML). Ο ρόλος της CSS, που την κάνει και απαραίτητη για κάθε προγραμματιστή που επιθυμεί να δημιουργήσει μια όμορφη και καλοσχεδιασμένη ιστοσελίδα, είναι ότι αναπτύσσει στυλιστικά μια ιστοσελίδα, δηλαδή διαμορφώνει περισσότερα χαρακτηριστικά, χρώματα, στοίχιση και γενικά δίνει περισσότερες δυνατότητες απ' όσες δίνει η HTML από μόνη της. Στην παρούσα διαδικτυακή αποκεντρωμένη εφαρμογή που αναπτύξαμε, πέραν κάποιων προσωπικών αρχείων CSS που δημιουργήθηκαν, χρησιμοποιήθηκε και ένα έτοιμο πλαίσιο ανάπτυξης ιστοσελίδων, γνωστό ως Bootstrap. Αυτό με τα CSS που παρέχει, συμβάλλει στην δημιουργία προσαρμοστικών ιστοσελίδων, δηλαδή ιστοσελίδων οι οποίες δυναμικά αλλάζουν τον τρόπο το περιεχόμενό τους και τον τρόπο παρουσίασής του, ανάλογα το μέγεθος της οθόνης στην οποία προβάλλονται, πράγμα απαραίτητο, αφού όλο και περισσότεροι χρήστες χρησιμοποιούν κινητά και tablet και όχι μόνο υπολογιστές για την περιήγησή τους στο διαδίκτυο. Κάποια βασικά πλεονεκτήματα που προσφέρει το framework του Bootstrap είναι τα εξής:

- *Εύκολο στη χρήση*: Όποιος έχει βασικές γνώσεις HTML και CSS μπορεί να αρχίσει να χρησιμοποιεί το Bootstrap.
- *Αποκριτικές δυνατότητες*: Το ανταποκρίσιμο CSS του Bootstrap προσαρμόζεται σε τηλέφωνα, tablet και επιτραπέζιους υπολογιστές.
- *Προσέγγιση για κινητές συσκευές*: Στο Bootstrap, τα στυλ για κινητά αποτελούν μέρος του βασικού πλαισίου του.
- *Συμβατότητα με τα προγράμματα περιήγησης*: Το Bootstrap είναι συμβατό με όλα τα σύγχρονα προγράμματα περιήγησης (Chrome, Firefox, Internet Explorer, Edge, Safari και Opera).

4.8.3 JavaScript

Εκτός από το backend κομμάτι (κώδικας που εκτελείται στον εξυπηρετητή του Node.js), JavaScript χρησιμοποιήθηκε και για το frontend (κώδικας που εκτελείται στον browser του χρήστη-πελάτη – client-side JavaScript). Η JavaScript είναι μία από τις πιο διαδεδομένες δυναμικές γλώσσες προγραμματισμού, αφού χρησιμοποιείται κυρίως σε διαδικτυακές εφαρμογές (και όχι μόνο) και υποστηρίζεται από όλους τους μοντέρνους φυλλομετρητές. Είναι ο καλύτερος ίσως τρόπος να δώσουμε δυναμικό περιεχόμενο σε μία ιστοσελίδα κάνοντάς την να επιτελεί σύνθετες λειτουργίες. Μπορεί επίσης να χρησιμοποιηθεί και για την ανάπτυξη προγραμμάτων που δεν εκτελούνται σε φυλλομετρητές, όπως για παράδειγμα σε εξυπηρετητές, βάσεις δεδομένων, επεξεργαστές PDF εγγράφων, Desktop εφαρμογές, αλλά και εφαρμογές κινητών. Για την εκτέλεσή της είναι απαραίτητη κάποια μηχανή JavaScript (JavaScript engine), όπως είναι για παράδειγμα η Google V8 που αναφέραμε και προηγουμένως. Η μηχανή JavaScript είναι στην ουσία ένα πρόγραμμα ή διερμηνέας που εκτελεί κώδικα γραμμένο στην γλώσσα JavaScript. Συναντάται κυρίως στους φυλλομετρητές, όμως χρησιμοποιείται και από άλλες πλατφόρμες, όπως είναι το Node.js. Η JavaScript είναι μία υψηλού επιπέδου, δυναμική, weakly typed, prototype-based, multi-paradigm, interpreted γλώσσα προγραμματισμού. Ως multi-paradigm γλώσσα, η JavaScript υποστηρίζει τα event-driven, functional, and imperative προγραμματιστικά στυλ. Έχει έτοιμες προγραμματιστικές διεπαφές (APIs) για την επεξεργασία κειμένου, πινάκων, ημερομηνιών, καθώς και τον χειρισμό DOM (Document Object Model), όμως δεν υποστηρίζει λειτουργίες εισόδου εξόδου (I/O), όπως δικτύωση, αποθήκευση, γραφικά, παρά βασίζει την υλοποίηση των λειτουργιών αυτών στο περιβάλλον εκτέλεσής της. Μερικά από τα πλεονεκτήματα της χρήσης JavaScript στις διαδικτυακές εφαρμογές είναι :

- *Λιγότερη αλληλεπίδραση μεταξύ πελάτη – εξυπηρετητή*: Μπορείτε να επικυρώσετε την είσοδο χρήστη πριν αποστείλετε τη σελίδα στον διακομιστή. Αυτό εξοικονομεί "κίνηση" διακομιστή, που σημαίνει λιγότερη φόρτωση αρχείων στον διακομιστή σας.
- *Άμεση ανατροφοδότηση στους χρήστες*: Δεν χρειάζεται να περιμένουν την επαναφόρτωση μιας σελίδας για να δουν αν έχουν ξεχάσει να εισαγάγουν κάτι.
- *Αυξημένη διαδραστικότητα*: Μπορείτε να δημιουργήσετε διασυνδέσεις που αντιδρούν όταν ο χρήστης αιωρείται πάνω τους με ποντίκι ή τις ενεργοποιεί μέσω του πληκτρολογίου.
- *Πλούσιες διεπαφές*: Μπορείτε να χρησιμοποιήσετε τη JavaScript για να συμπεριλάβετε στοιχεία, όπως στοιχεία μεταφοράς και απόθεσης, για να δώσετε μια πλούσια διεπαφή στους επισκέπτες του ιστότοπού σας.

Για να εφαρμοστούν κάποια απαραίτητα Bootstrap JavaScript plugins στην εφαρμογή μας έγινε χρήση της (ίσως) πιο διαδεδομένης βιβλιοθήκης που προσφέρει η JavaScript, η jQuery. Η jQuery είναι μια γρήγορη, μικρή και πλούσια σε χαρακτηριστικά βιβλιοθήκη JavaScript. Κάνει πράγματα όπως η διασταύρωση και ο χειρισμός εγγράφων HTML, ο χειρισμός συμβάντων (event handling), η κινούμενη εικόνα και τις κλήσεις AJAX (Asynchronous JavaScript And XML) πολύ πιο απλές με ένα εύχρηστο API (Application Programming Interface) που λειτουργεί σε πολλά προγράμματα περιήγησης.

5.Σχεδιασμός και υλοποίηση εφαρμογής

Στο κεφάλαιο αυτό θα παρουσιαστούν όλες οι λεπτομέρειες (κώδικες και ανάλυση διαδικασιών τους) που αφορούν τον σχεδιασμό και την υλοποίηση της αποκεντρωμένης εφαρμογής ηλεκτρονικής ψηφοφορίας (Εκλογές ΗΠΑ 2020) που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας.

5.1 Αναλυτική παρουσίαση έξυπνου συμβολαίου USAelections

Όπως έχουμε ξαναπεί, τα έξυπνα συμβόλαια είναι αναπόσπαστο κομμάτι των αποκεντρωμένων εφαρμογών. Περιέχουν τον κώδικα που εκτελείται στην Ethereum Virtual Machine και χωρίς αυτά θα ήταν αδύνατη η ανάπτυξη οποιασδήποτε τέτοιας εφαρμογής. Ο κώδικας των έξυπνων συμβολαίων, από την στιγμή που θα υλοποιηθεί στο Blockchain, είναι οριστικός και δεν αλλάζει. Στην περίπτωση που βρεθεί κάποιο σφάλμα κατά της διάρκεια λειτουργίας της εφαρμογής, ο μόνος τρόπος για να το διορθώσουμε θα ήταν να υλοποιήσουμε εξαρχής ένα νέο συμβόλαιο. Για αυτόν τον λόγο δόθηκε ιδιαίτερη προσοχή από πλευρά μας ως προγραμματιστές στο στάδιο ανάπτυξης αυτού του συμβολαίου. Να επισημάνουμε βέβαια εδώ, ότι όταν λέμε να υλοποιηθεί ένα συμβόλαιο στο Blockchain, εννοούμε στο κύριο δίκτυο του Ethereum Blockchain. Στο στάδιο ανάπτυξης της εφαρμογής μας εργαστήκαμε σε ένα τοπικό δίκτυο (testnet) Blockchain που μας προσφέρει η Ganache, πράγμα που σημαίνει ότι η υλοποίηση νέων συμβολαίων δεν κοστίζουν πραγματικά Ethers.

Στο έξυπνο συμβόλαιο μας USAelections υλοποιείται η λειτουργικότητα μιας αποκεντρωμένης εφαρμογής ηλεκτρονικής ψηφοφορίας. Εισάγονται οι υποψήφιοι πρόεδροι και έπειτα αναπτύσσεται η διαδικασία υποβολής ψήφου από την μεριά των ψηφοφόρων. Ας δούμε πιο αναλυτικά τον κώδικα του συμβολαίου.

```
//// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.8.0;
```

Στην πρώτη αυτή γραμμή κώδικα του συμβολαίου μας καθορίζονται οι εκδόσεις της Solidity που μπορούν να χρησιμοποιηθούν από τον compiler.

```
//candidate informations
struct Candidate{
uint id;
string name;
uint voteCount;
string party;
}
```

Έπειτα μέσω ενός struct με το όνομα Candidate, δηλαδή μιας δομής που είναι προσαρμοσμένη ώστε να ομαδοποιεί πολλές μεταβλητές διαφορετικού τύπου, αποθηκεύουμε όλες τις απαραίτητες πληροφορίες για κάθε υποψήφιο πρόεδρο. Πιο συγκεκριμένα, ορίζουμε έναν αύξοντα αριθμό *id* για κάθε υποψήφιο, το όνομά του *name*, τον αριθμό των ψήφων που έχει συγκεντρώσει *voteCount* και το κόμμα στο οποίο ανήκει *party*.

```
//candidates in an array
Candidate[] public _candidates;
```

Αποθηκεύουμε τον κάθε υποψήφιο πρόεδρο (κάθε struct *Candidate*) σε ένα διάνυσμα *_candidates* μεγέθους ίσο με τον αριθμό των υποψηφίων προέδρων που θα εισαχθούν.

Παρατήρηση: Το προεπιλεγμένο είδος κάθε μεταβλητής στην Solidity ως προς την ορατότητα του (visibility) είναι το private. Επειδή όμως κάποιες μεταβλητές θέλουμε να "διαβάζονται" από άλλα smart contracts ή από μια αποκεντρωμένη εφαρμογή (όπως και στην περίπτωση μας) θα τις ορίζουμε εμείς ως public για να είναι εύκολα προσβάσιμες.

```
//given the account address that has voted => return true or false respectively
mapping(address=>bool) public isVoted;
//given an integer => return Candidate information
mapping (uint => Candidate) public candidates;
```

Εδώ δημιουργούμε δύο δομές mapping. Ένα *mapping(_KeyType=>_ValueType)* θα μπορούσαμε να το φανταστούμε ως ένα κατακερματισμό πινάκων αναζήτησης που ουσιαστικά έχει δημιουργηθεί έτσι ώστε να υπάρχει κάθε πιθανό κλειδί και να αντιστοιχίζεται σε μια τιμή. Η πρώτη δομή mapping *isVoted* που δημιουργήσαμε δέχεται ως index (key) την Ethereum address του ψηφοφόρου και αποθηκεύει στη δομή την δυαδική τιμή True ή False αν έχει ψηφίσει ή όχι αντίστοιχα. Η δεύτερη δομή mapping *candidates* που δημιουργήσαμε δέχεται ως index (key) τον αύξοντα αριθμό κάθε υποψήφιου προέδρου και αποθηκεύει στη δομή το struct με τις πληροφορίες του.

```
//save number of Candidates
uint public candidatesCounter;
```

Επίσης ορίζουμε μια public ακέραια μεταβλητή που θα κρατάει τον αριθμό των υποψήφιων προέδρων που έχουν εισαχθεί.

```
//declare vote event
event voteEvent(uint indexed _id);
```

Ορίζουμε ένα event με το όνομα *voteEvent*. Χρησιμοποιείται για την ενημέρωση των κόμβων του δικτύου του Blockchain ώστε να πληροφορηθούν ότι ο χρήστης-ψηφοφόρος επέλεξε να ψηφίσει τον πρόεδρο με αύξοντα αριθμό ίσο με το όρισμα που δέχεται το event *_id*.

```
//constructor function
constructor() public{
  addCandidate("Ντόναλντ Τραμπ","Ρεπουμπλικανικό");
  addCandidate("Τζο Μπάιντεν","Δημοκρατικό");
  }
```

Ο παραπάνω κατασκευαστής ή αλλιώς constructor του έξυπνου συμβολαίου μας εκτελείται αυτόματα μόλις το συμβόλαιο ενεργοποιηθεί στο Ethereum Blockchain. Αυτό που κάνει είναι να καλεί την συνάρτηση *addCandidate*, που θα ορίσουμε αμέσως μετά, ώστε να προσθέσει τους δύο υποψήφιους των προεδρικών εκλογών των ΗΠΑ το 2020, Ντόναλντ Τραμπ και Τζο Μπάιντεν. Λόγω του πως ορίζουμε παρακάτω την συνάρτηση *addCandidate,* στον κατασκευαστή χρειάζεται να θέσουμε μόνο το όνομα του υποψηφίου και το κόμμα στο οποίο ανήκει. Τα υπόλοιπα δύο

χαρακτηριστικά ενός struct *Candidate* (*id* και *voteCount*) ως ακέραιοι έχουν εξ'ορισμού τιμή ίση με 0.

```
function addCandidate(string memory _name,string memory _party) private {
  candidatesCounter ++;
  candidates[candidatesCounter]=Candidate(candidatesCounter,_name,0,_party);
  }
```

Εδώ ορίζουμε την συνάρτηση *addCandidate()* που χρησιμοποιεί ο κατασκευαστής. Η συνάρτηση αυτή ως προς την ορατότητα της (visibility) την ορίζουμε ως private, καθώς εμείς μόνο ως προγραμματιστές έχουμε δικαίωμα αλλαγής των υποψηφίων και κανείς άλλος που αλληλεπιδρά με την εφαρμογή μας. Αρχικά, δέχεται ως ορίσματα δύο strings, ένα για το όνομα του υποψηφίου και ένα για το κόμμα το οποίο εκπροσωπεί (στον constructor λαμβάνουν την επιθυμητή τιμή). Σε κάθε κλήση της συνάρτησης, δηλαδή σε κάθε εισαγωγή υποψηφίου, αυξάνεται ο αριθμός των υποψηφίων κατά 1 (αρχικά η μεταβλητή *candidatesCounter* έχει τιμή 0). Έπειτα στο mapping *candidates* που ορίσαμε και στην κατάλληλη θέση που αντιστοιχεί στην τιμή του *candidatesCounter*, εισάγουμε τις 4 απαραίτητες πληροφορίες του κάθε υποψήφιου προέδρου.

```
function vote(uint _id) public{
  //require that they haven't voted before
  require(!isVoted[msg.sender]);
  //requite that candidate exist
  require(_id>0 && _id<=candidatesCounter);
  //record that voter has succesfully submitted his vote
  isVoted[msg.sender]=true;

  //update candidates voteCount
  candidates[_id].voteCount ++;

  //emit voting event
  emit voteEvent(_id);
  }
```

Εδώ ορίζουμε την πιο βασική συνάρτηση όλου του συμβολαίου μας *vote()*. Την ορίζουμε ως public ώστε να μπορεί να κληθεί από άλλα συμβόλαια ή ,στην περίπτωση μας, μέσω web3 από τον JavaScript κώδικα μας. Αυτή η συνάρτηση μπορεί να εκτελείται από κάθε ψηφοφόρο, όποτε αυτός επιθυμήσει να συμμετάσχει στην εκλογική διαδικασία (φυσικά όμως μέσα στα χρονικά πλαίσια που καθορίζονται οι εκλογές). Οι ψηφοφόροι αυτό που πρέπει να κάνουν είναι να στείλουν απλά το αναγνωριστικό του υποψηφίου που θέλουν ψηφίσουν. Αυτό το αναγνωριστικό το δέχεται ως ακέραια παράμετρο η συνάρτηση μας, όποτε καλείται, με το όνομα *_id*. Μέσω της συνάρτησης αυτής καταγράφονται οι ψήφοι. Αρχικά, εντοπίζει ποιος αυτή τη στιγμή προσπαθεί να εκτελέσει αυτήν τη λειτουργία του έξυπνου συμβολαίου μας (*msg.sender*). Εάν το άτομο αυτό έχει δικαίωμα ψήφου, τότε ελέγχεται περαιτέρω εάν το αναγνωστικό *_id* που θέτει ως ψήφο είναι δεκτό ως προς τον υποψήφιο που αντιστοιχεί. Στη συνέχεια, το άτομο επισημαίνεται (μαρκάρεται) πως έχει ήδη ψηφίσει και ο αριθμός ψήφων του υποψηφίου προέδρου της επιλογής του αυξάνεται κατά έναν. Τέλος, εκπέμπεται ένα event *voteEvent* με παράμετρο το αναγνωριστικό *_id* του προέδρου που επιθυμεί να ψηφίσει αυτός που καλεί την συνάρτηση, ώστε να ενημερωθούν οι κόμβοι του δικτύου και η ιστοσελίδα της ψηφοφορίας, για να γίνει

ορατή σε όλους η καταγραφή της ψήφου.

Το έξυπνο συμβόλαιο μας σταματάει εδώ. Θα μπορούσαμε να εμπλουτίσουμε και άλλο το περιεχόμενο του με περισσότερες λειτουργίες. Επιλέξαμε όμως να κρατήσουμε το smart contract μας όσο πιο απλό γίνεται, διότι όσο πιο πολύπλοκο είναι ένα τέτοιο συμβόλαιο τόσο πιο πολύ gas θα κοστίζει στο δίκτυο του Ethereum. Αυτό συνεπάγεται μεγαλύτερο κόστος σε ETH από του χρήστες που επιλέγουν να το χρησιμοποιήσουν (ψηφοφόροι). Επομένως, διαδικασίες όπως εύρεση νικητή στο τέλος της ψηφοφορίας, ορισμού χρονικού πλαισίου εκλογικής διαδικασίας και πολλές άλλες λειτουργίες επιλέχτηκαν να γίνουν από το frontend προγραμματιστικό κομμάτι της εφαρμογής μας, μέσω της JavaScript.

5.2 Περιγραφή διαδικασιών

Στο κεφάλαιο αυτό θα περιγραφούν αναλυτικά κάποιες από τις βασικές λειτουργίες της εφαρμογής μας, που καλούνται από την ιστοσελίδα (*Παράρτημα ΙΙ*) της αποκεντρωμένης εφαρμογής μας μέσω της JavaScript (και κατ' επέκταση της Solidity). Υπάρχουν δύο αρχεία JavaScript (*.js) που επιτελούν το βασικό κομμάτι αυτών των λειτουργιών. Το ένα είναι το *time.js* και το άλλο είναι το *app.js*.

time.js

```
// Set the date we're counting down to
var countDownDate = new Date("Nov 28, 2020 20:49:00").getTime();
// Update the count down every 1 second
var x = setInterval(function () {
 // Get today's date and time
 var now = new Date().getTime();
 // Find the distance between now and the count down date
 var distance = countDownDate - now;
 // Time calculations for days, hours, minutes and seconds
 var days = Math.floor(distance / (1000 * 60 * 60 * 24));
 var hours = Math.floor((distance % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60));
 var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
 var seconds = Math.floor((distance % (1000 * 60)) / 1000);
 // Display the result in the element with id="demo"
 document.getElementById("time").innerHTML =
  days + "d " + hours + "h " + minutes + "m " + seconds + "s ";
 //If the count down isn't finished,don't display winner
 if (distance >= 0) {
  $("#winner").hide();
 }
 // If the count down is finished, write some text
 if (distance < 0) {
  clearInterval(x);
  document.getElementById("time").innerHTML = "ΛΗΞΗ ΧΡΟΝΟΥ";
  $("button").hide();
  $("label").hide();
  $("select").hide();
  $("#winner").show();
  document.getElementById("_time").innerHTML = "";
```

```
 }
}, 1000);
```

Στον παραπάνω κώδικα, που αποτελεί αναπόσπαστο κομμάτι της εφαρμογής μας, ορίζουμε έναν timer (χρονομετρητή) που μετράει αντίστροφα μέχρι το κλείσιμο των καλπών. Εμείς ως προγραμματιστές ορίζουμε πότε θα λήξει η εκλογική διαδικασία, καθώς εμείς θέτουμε την ακριβή ημερομηνία και ώρα που θα τελειώσει αυτή, όπως φαίνεται στην πρώτη γραμμή του κώδικα. Επίσης ορίζουμε μια συνάρτηση που επιτελεί όλους τους απαραίτητους μαθηματικούς υπολογισμούς και μετατροπές ώστε να γίνει δυνατή η επίτευξη του τελικού αποτελέσματος. Μέσα σε αυτή τη συνάρτηση, ανάλογα με το αν έχει λήξει ο χρόνος της ψηφοφορίας ή όχι, κρύβουμε και εμφανίζουμε κομμάτια της ιστοσελίδας αντίστοιχα. Για παράδειγμα, αν ακόμα δεν έχει λήξει ο χρόνος της ψηφοφορίας τότε δεν εμφανίζουμε τον νικητή στην ιστοσελίδα του χρήστη. Αν όμως έχει λήξει ο χρόνος της εκλογικής διαδικασίας, τότε αφαιρούμε από τον χρήστη τη δυνατότητα να υποβάλλει την ψήφο του και εμφανίζουμε αυτόματα τον νικητή και τις ψήφους που έχει συλλέξει. Όλες αυτές οι λειτουργίες απόκρυψης και εμφάνισης γίνονται σε άμεση επικοινωνία της HTML ιστοσελίδας μας με όλα τα αρχεία JavaScript που είναι απαραίτητα, όπως και το παραπάνω, μέσω των δυνατοτήτων που προσφέρει η βιβλιοθήκη jQuery.

app.js

Το αρχείο app.js αποτελεί τη καρδιά του frontend κομματιού της εφαρμογής μας. Σε αυτό τον κώδικα επιτελούνται εργασίες απαραίτητες για την υλοποίηση μιας αποκεντρωμένης εφαρμογής. Ας δούμε αναλυτικά τον κώδικα αυτό (“σπασμένο” σε ξεχωριστά μέρη για να γίνει ευκολότερη η ανάλυση του) και τις συναρτήσεις από τις οποίες αποτελείται.

```
App = {
web3Provider: null,
 contracts: {},
 account: "0x0",

 init: function () {
  return App.initWeb3();
 },

 initWeb3: function () {
  ethereum.enable();
  if (typeof web3 !== "undefined") {
   // If a web3 instance is already provided by MetaMask.
   App.web3Provider = web3.currentProvider;
   web3 = new Web3(web3.currentProvider);
  } else {
   // Specify default instance from Ganache
   App.web3Provider = new Web3.providers.HttpProvider(
    "http://localhost:7545"
   );
   web3 = new Web3(App.web3Provider);
  }
  return App.initContract();
```

```
},

initContract: function () {
 $.getJSON("USAelections.json", function (election) {
  // Instantiate a new truffle contract from the artifact
  App.contracts.USAelections = TruffleContract(election);
  // Connect provider to interact with contract
  App.contracts.USAelections.setProvider(App.web3Provider);
  App.listenForEvents();
  App.listenForAccountChange();
  return App.mainFun();
 });
},

//Listen for Account change
listenForAccountChange: function () {
 ethereum.on("accountsChanged", function (accounts) {
  App.account = accounts[0];
  App.mainFun();
 });
},

// Listen for events emitted from the contract
listenForEvents: function () {
 App.contracts.USAelections.deployed().then(function (instance) {
  // Restart Chrome if you are unable to receive this event
  instance
   .voteEvent(
    {},
    {
     fromBlock: "latest",
     toBlock: "latest",
    }
   )
   .watch(function (error, event) {
    console.log("event emitted", event);
    // Reload when a new vote is recorded
    App.mainFun();
   });
 });
},
```

Το πρώτο αυτό κομμάτι κώδικα της εφαρμογής μας αποτελεί ένα διαδικαστικό μέρος θα λέγαμε, ώστε να καταστεί δυνατή η σύνδεση της αποκεντρωμένης εφαρμογής με τα εργαλεία που χρειάζονται για την επιτυχή υλοποίηση της. Πιο συγκεκριμένα, υλοποιούν την σύνδεση ανάμεσα στην εφαρμογή και το plugin του MetaMask, ώστε να "φορτωθεί" ο λογαριασμός του ψηφοφόρου στην πλατφόρμα ψηφοφορίας. Επίσης ορίζεται ο localhost στον οποίο θα "ακούει" η εφαρμογή μας. Αυτός στα πλαίσια ανάπτυξης της Dapp μας δίνεται μέσω της Ganache και βρίσκεται στη θύρα 7545 συνήθως. Αφού έχει αποκατασταθεί η σύνδεση με τα εργαλεία που είναι απαραίτητα

για το ανέβασμα της εφαρμογής στο τοπικό Ethereum Blockchain, φορτώνεται το έξυπνο συμβόλαιο μας μέσω της συνάρτησης *initContract*. Αυτή λαμβάνει ένα αρχείο *.json που είναι το αρχείο κώδικα (lightweight μορφής για αποθήκευση και μεταφορά δεδομένων) που προκύπτει έπειτα από το compile και deploy του συμβολαίου μας μέσω του Truffle. Έπειτα συνδέει το συμβόλαιο μας με τον web3 provider που είναι το MetaMask στη προκειμένη περίπτωση. Με αυτό το τρόπο μπορούν να συνδεθούν οι λογαριασμοί Ethereum στο συμβόλαιο μας. Επίσης ορίζονται κάποιες συναρτήσεις ώστε να αντιλαμβάνεται η εφαρμογή μας την εκτέλεση κάθε event (υποβολής ψήφου) και την αλλαγή λογαριασμών στο σύστημα της ψηφοφορίας. Όταν γίνεται αντιληπτό κάτι τέτοιο η σελίδα ανανεώνεται αυτόματα και φορτώνει τα ενημερωμένα δεδομένα. Να σημειώσουμε εδώ ότι η αποκεντρωμένη εφαρμογή μας έχει σχεδιαστεί ώστε να τρέχει σε δοκιμαστικό επίπεδο (χωρίς να χρειάζεται η σπατάλη πραγματικών Ethers) μέσω του τοπικού Ethereum Blockchain που προσφέρει η Ganache. Εάν στο μέλλον η εφαρμογή χρειαστεί να τρέξει στο main net του Ethereum πρέπει να γίνουν σημαντικές αλλαγές στο κομμάτι αυτό του κώδικα.

```
mainFun: function () {
 var electionInstance;
 var loader = $("#loader");
 var content = $("#content");
 loader.show();
 content.hide();
// Load account data
 web3.eth.getCoinbase(function (err, account) {
  if (err === null) {
   App.account = account;
   $("#accountAddress").html(
    "Διεύθνση λογαριασμού τρέχοντος ψηφοφόρου: " + account
   );
  }
 });
// Load contract data
 App.contracts.USAelections.deployed()
  .then(function (instance) {
   electionInstance = instance;
   return electionInstance.candidatesCounter();
  })
  .then(function (candidatesCounter) {
   // Store all promises to get candidate info
   const promises = [];
   for (var i = 1; i <= candidatesCounter; i++) {
    promises.push(electionInstance.candidates(i));
   }
   // Once all candidates are received, add to dom
   Promise.all(promises).then((candidates) => {
    var candidatesResults = $("#candidatesResults");
    candidatesResults.empty();
    var candidatesSelect = $("#candidatesSelect");
    candidatesSelect.empty();
    candidatesSelect.append(
```

```
    $("<option>", {
     text: "Επιλέξτε πρόεδρο",
     disabled: "disabled",
     selected: "true",
     name: "choosePresident",
    })
   );
  var candidatesResults2 = $("#candidatesResults2");
  candidatesResults2.empty();
   var _voteCount = 0;
  var maxvoteId = 0;
  var i = 1;
  var psifoi = [];
  var candidatesWinner = $("#candidatesWinner");
  candidatesWinner.empty();
   candidates.forEach((candidate) => {
   var id = candidate[0];
   var name = candidate[1];
   var voteCount = candidate[2];
   var party = candidate[3];
   // Give candidate Result
   var candidateTemplate =
    "<tr><td>" +
    party +
    "</td><td>" +
    name +
    "</td><td>" +
    voteCount +
    "</td></tr>";
   candidatesResults.append(candidateTemplate);
   var candidateTemplate2 =
    "<tr><td>" + party + "</td><td>" + name + "</td></tr>";
   candidatesResults2.append(candidateTemplate2);
    // Give candidate ballot option
   var candidateOption =
    "<option value='" + id + "' >" + name + "</ option>";
   candidatesSelect.append(candidateOption);
    //winner
   if (i == 1) {
    psifoi.push(voteCount);
   }
   if (i == 2) {
    psifoi.push(voteCount);
   }
   if (voteCount > _voteCount) {
    _voteCount = voteCount;
    maxvoteId = i;
   }
   if (i == candidatesCounter) {
```

```
    if (psifoi[0].toString() == psifoi[1].toString()) {
     candidatesWinner.append("Ισοψηφία");
    } else {
     electionInstance
      .candidates(maxvoteId)
      .then(function (candidate) {
       candidatesWinner.append(candidate[1]);
      });
    }
   }
   i = i + 1;
  });
 });
return electionInstance.isVoted(App.account);
})
.then(function (hasVoted) {
 var now = new Date().getTime();
 var countDownDate = new Date("Nov 28, 2020 20:49:00").getTime();
 // Do not allow a user to vote
 if (hasVoted && countDownDate - now > 0) {
  $("#pinakas1").hide();
  $("#pinakas2").show();
  $("form").hide();
 alert("Η ψήφος σας υποβλήθηκε επιτυχώς");
 }
if (hasVoted && countDownDate - now < 0) {
  $("button").hide();
  $("label").hide();
  $("select").hide();
  $("#winner").show();
  $("#pinakas2").hide();
 }
 if (!hasVoted && countDownDate - now < 0) {
  alert("Δυστυχώς δεν προλάβατε να υποβάλλετε την ψήφο σας");
  $("button").hide();
  $("label").hide();
  $("select").hide();
  $("#winner").show();
  $("#pinakas2").hide();
 }
 if (!hasVoted && countDownDate - now > 0) {
  $("#pinakas1").hide();
  $("#pinakas2").show();
 }
  loader.hide();
 content.show();
})
.catch(function (error) {
 console.warn(error);
```

```
  });
 //state example:Value1: "Alabama",Value2: "Alaska",...
 var State = {...};
 var select_ = document.getElementById("voterState");
 for (index in State) {
  select_.options[select_.options.length] = new Option(State[index], index);
 }
},
```

Στο κομμάτι αυτό του κώδικα μας ορίζουμε την κεντρική συνάρτηση της εφαρμογής μας *mainFun*, για αυτό και επιλέξαμε αυτό το όνομα. Το πρώτο πράγμα που εξασφαλίζει η συνάρτηση αυτή είναι να αντανακλά στην ιστοσελίδα μας τον λογαριασμό Ethereum του τρέχοντος ψηφοφόρου που χρησιμοποιεί την πλατφόρμα μας. Να σημειωθεί εδώ ότι ο χρήστης πρέπει να επαληθεύσει την ταυτοποίηση της διεύθυνσης του με αυτή που εμφανίζεται στην ιστοσελίδα, διαφορετικά η διαδικασία υποβολής της ψήφου δε θα μπορέσει να συνεχιστεί. Αφού λοιπόν φορτωθεί ο κατάλληλος λογαριασμός Ethereum που αντιστοιχεί στον χρήστη, τότε έχει σειρά η φόρτωση των δεδομένων από το έξυπνο συμβόλαιο μας. Αλληλεπιδρούμε λοιπόν με το συμβόλαιο και αντλώντας από αυτό τον αριθμό των υποψηφίων προέδρων *candidatesCounter* ορίζουμε μια νέα υποσυνάρτηση που δέχεται ως όρισμα τον αριθμό αυτό. Μέσα στην υποσυνάρτηση αυτή η πρώτη κίνηση που γίνεται είναι να εισάγουμε σε ένα δικό μας διάνυσμα *promises* όλα τα στοιχεία των υποψηφίων προέδρων που βρίσκονται στη δομή mapping *candidates* του έξυπνου συμβολαίου μας. Σε αυτό το κομμάτι θα γίνει χρήση ενός object της JavaScript γνωστό ως Promise. Το αντικείμενο αυτό αποτελείται από ένα κομμάτι "παραγωγής κώδικα" και ένα κομμάτι "κατανάλωσης κώδικα". Εμείς εκμεταλλευόμαστε το κομμάτι "κατανάλωσης κώδικα" αυτών των objects και αυτός ενεργοποιείται όταν φορτωθούν όλα τα δεδομένα στο διάνυσμα *promises*. Έχουμε λοιπόν απορροφήσει από το έξυπνο συμβόλαιο μας όλα τα δεδομένα που χαρακτηρίζουν έναν υποψήφιο πρόεδρο. Ορίζουμε κάποιες μεταβλητές χρήσιμες για τη συνέχεια (μεταβλητή αποτελεσμάτων, μεταβλητή επιλογής προέδρου, μεταβλητή νικητή κλπ.) με τρόπο ώστε να μπορούν να κληθούν από τον HTML κώδικα μας. Για κάθε υποψήφιο ξεχωριστά αποθηκεύομε σε αυτές τις μεταβλητές το σύνολο των ψήφων που έχει συγκεντρώσει ο κάθε υποψήφιος, το κόμμα στο οποίο ανήκει κλπ. Αφού γίνει αυτή η διαδικασία για κάθε υποψήφιο ξεχωριστά είναι η ώρα να ελέγξουμε ποιος έχει τις περισσότερες ψήφους ή ακόμη και αν έχουμε ισοψηφία. Με χρήση του κατάλληλου κώδικα καταφέρνουμε να δούμε ποιος είναι ο νικητής των εκλογών και αυτό το αποτέλεσμα το αποθηκεύουμε στη μεταβλητή *candidatesWinner*. Εδώ τελειώνει η πρώτη υποσυνάρτηση. Αντλούμε στη συνέχεια από το έξυπνο συμβόλαιο, ως όρισμα σε μια νέα υποσυνάρτηση, το mapping *isVoted* που επιστρέφει True ή False αν έχει ψηφίσει ή όχι ο τρέχων ψηφοφόρος. Στην υποσυνάρτηση αυτή ελέγχοντας κάθε φορά αν ο χρήστης έχει ψηφίσει ή όχι και αν η χρονική προθεσμία για να ψηφίσει έχει λήξει ή όχι, κρύβουμε και εμφανίζουμε στοιχεία στην ιστοσελίδα μας αντίστοιχα. Δηλαδή, για παράδειγμα, αν ο χρήστης δεν έχει ψηφίσει αλλά η ψηφοφορία έχει λήξει και επιχειρήσει να μπει στην πλατφόρμα, δέχεται ένα μήνυμα ότι δεν πρόλαβε να υποβάλλει ψήφο και αντικρίζει τον νικητή των εκλογών και τις ψήφους που συγκέντρωσε ο καθένας. Αφού εξαντλήσουμε κάθε τέτοια πιθανή εκδοχή τελειώνει και η δεύτερη υποσυνάρτηση. Στο τέλος της κεντρικής μας συνάρτησης ορίζουμε τις πολιτείες των ΗΠΑ, ώστε στην συνέχεια να κληθεί ο ψηφοφόρος να επιλέξει την πολιτεία ανήκουν τα εκλογικά του δικαιώματα. Σε αυτό το κομμάτι να αναφέρουμε ότι σε μια μελλοντική επέκταση της εφαρμογής και έχοντας σε μια δομή δεδομένων

αποθηκευμένα στοιχεία για κάθε ψηφοφόρο (βλέπε κεφάλαιο *Παρατηρήσεις*), η επιλογή άλλης πολιτείας πέραν της δικαιοδοσίας του θα απαγορεύεται από την εφαρμογή ρητώς.

```
selecState: function (a) {
 var b = document.getElementById("candidatesSelect");
 if (
  a.options[a.selectedIndex].text == "Επιλέξτε πολιτεία" &&
  b.options[b.selectedIndex].text == "Επιλέξτε πρόεδρο"
 ) {
  $("#koumpi").prop("disabled", true);
 } else if (a.options[a.selectedIndex].text == "Επιλέξτε πολιτεία") {
  $("#koumpi").prop("disabled", true);
 } else if (b.options[b.selectedIndex].text == "Επιλέξτε πρόεδρο") {
  $("#koumpi").prop("disabled", true);
 } else {
  $("#koumpi").prop("disabled", false);
 }
},
selecPres: function (a) {
 var b = document.getElementById("voterState");
 if (
  a.options[a.selectedIndex].text == "Επιλέξτε πρόεδρο" &&
  b.options[b.selectedIndex].text == "Επιλέξτε πολιτεία"
 ) {
  $("#koumpi").prop("disabled", true);
 } else if (b.options[b.selectedIndex].text == "Επιλέξτε πολιτεία") {
  $("#koumpi").prop("disabled", true);
 } else if (a.options[a.selectedIndex].text == "Επιλέξτε πρόεδρο") {
  $("#koumpi").prop("disabled", true);
 } else {
  $("#koumpi").prop("disabled", false);
 }
},
```

Συνεχίζοντας ορίζουμε δύο βοηθητικές συναρτήσεις *selecState* και *selecPres* που θα κάνουν την εφαρμογή μας πιο λειτουργική. Πιο συγκεκριμένα, όπως και θα δούμε στην επίδειξη λειτουργικότητας της εφαρμογής μας, αυτές οι συναρτήσεις αποκλείουν την περίπτωση ένας ψηφοφόρος να μην επιλέξει την πολιτεία την οποία ανήκει ή να μην επιλέξει κατάλληλο πρόεδρο. Αν δηλαδή επιλέξει τον υποψήφιο πρόεδρο που θέλει για νικητή αλλά όχι την πολιτεία του, τότε το κουμπί υποβολής ψήφου θα παραμένει ανενεργό μέχρι να πράξει αναλόγως.

```
submitVote: function () {
  var candidateId = $("#candidatesSelect").val();
  App.contracts.USAelections.deployed()
   .then(function (instance) {
    return instance.vote(candidateId, { from: App.account });
   })
   .then(function (result) {
    // Wait for votes to update
```

```
    $("button").hide();
    $("label").hide();
    $("select").hide();
    $("#loader").show();
   })
   .catch(function (err) {
    console.error(err);
   });
 },
}; //end of app
```

Η τελευταία αυτή συνάρτηση είναι αυτή η οποία αξιοποιεί την επιλογή της ψήφου από τους ψηφοφόρους και την αντιστοιχεί στα προσωπικά στοιχεία του κάθε υποψηφίου. Ουσιαστικά είναι η συνάρτηση υποβολής ψήφου. Η υποβολή της ψήφου γίνεται μέσω της αλληλεπίδρασης φυσικά με το έξυπνο συμβόλαιο μας και την συνάρτηση *vote* η οποία υπάρχει εκεί. Ανάλογα λοιπόν με την επιλογή των ψηφοφόρων ενημερώνονται και οι αντίστοιχες μεταβλητές στο έξυπνο συμβόλαιο. Μέχρι να γίνει αυτή η υποβολή στο σύστημα (ενημέρωση κόμβων δικτύου Blockchain κλπ.) επιλέγουμε να μην εμφανίζονται όλα τα στοιχεία στη σελίδα μας, αλλά ένας loader.

Εδώ τελειώνει η παρουσίαση των διαδικασιών που υλοποιούνται στην αποκεντρωμένη εφαρμογή ηλεκτρονικής ψηφοφορίας που αναπτύξαμε. Όλα αυτά φυσικά αντλούνται και εξάγονται στο οπτικό περιβάλλον του χρήστη μέσω του *index.html* αρχείου μας που είναι ο κώδικας της ιστοσελίδας μας. Τα περιεχόμενα του αρχείου αυτού (κώδικας HTML) βρίσκονται στο τέλος της διπλωματικής αυτής εργασίας, στο αντίστοιχο παράρτημα.

5.3 Παρατηρήσεις

Παρατήρηση 1: Δημιουργήσαμε ένα έξυπνο συμβόλαιο στο Ethereum που επιτρέπει να ελεγχθούν και να μετρηθούν οι ψήφοι όταν έρθει η ώρα λήξης των εκλογών. Με τον τρόπο που είναι δομημένη η αποκεντρωμένη εφαρμογή μας μπορούμε να συμπεριλάβουμε οποιονδήποτε λογαριασμό Ethereum στις εκλογές. Χρησιμοποιώντας τις τιμές κατακερματισμού των λογαριασμών Ethereum των χρηστών, η ταυτότητα των ατόμων δεν μπορεί να αποκαλυφτεί. Όμως προκύπτει άμεσα το ερώτημα: Τι εμποδίζει κάποιον να φτιάξει πολλούς λογαριασμούς Ethereum και να ψηφίσει έτσι πάνω από μια φορά; Ο προσωπικός έλεγχος ταυτότητας κάθε ψηφοφόρου στην συγκεκριμένη εφαρμογή θεωρείται διαφορετικό υποπρόβλημα και έμεινε εκτός του πεδίου εφαρμογής αυτής της μελέτης, όπως και οι νομικοί κανονισμοί που ορίζουν μια εκλογική διαδικασία. Παρόλα αυτά προτείνουμε μια θεωρητική λύση στο παραπάνω πρόβλημα που τέθηκε: Υποθέτουμε ότι έχει δοθεί σε κάθε πολίτη των ΗΠΑ που πληρεί τα κριτήρια υποβολής ψήφου μια μοναδική Ethereum address επιφορτισμένη με το κατάλληλο ποσό ETH ώστε να μπορεί ο ψηφοφόρος να ψηφίσει. Το σύνολο αυτών των διευθύνσεων θα τρέχουν σε ένα αναγνωρισμένο δίκτυο αποκλειστικά για την εκλογική διαδικασία (όπως η Ganache τρέχει συνήθως στον localhost 7545). Ο οργανισμός που θα είναι υπεύθυνος για τη δημιουργία και την διανομή τέτοιων μοναδικών διευθύνσεων Ethereum θα είναι ο οργανισμός U.S Vote Foundation (αυτός ο οργανισμός υπάρχει σήμερα και τον χρησιμοποιεί ο κάθε πολίτης των ΗΠΑ ώστε να κάνει register και να αποκτήσει το δικαίωμα της ψήφου). Αφού λοιπόν ο χρήστης κάνει register στην πλατφόρμα αυτή (π.χ με τα στοιχεία Taxis που έχουμε αντίστοιχα στην Ελλάδα) θα του αποδοθεί μια

μοναδική διεύθυνση Ethereum ώστε να τη χρησιμοποιήσει για να εισέλθει μέσω του MetaMask στην ιστοσελίδα της ψηφοφορίας μας. Από εκεί και έπειτα είναι ζήτημα προγραμματιστικό από πλευράς μας να έρθουμε σε επαφή με τον οργανισμό U.S Vote Foundation και να μας παραχωρήσει όλες τις διευθύνσεις Ethereum που αποδόθηκαν, ώστε να αναγνωρίζονται αυτές και μόνο αυτές ως έγκυρες στο περιβάλλον της εφαρμογής μας. Διαφορετικά, άλλες διευθύνσεις Ethereum δεν θα μπορούν να υποβάλλουν επιτυχώς ψήφο. Αυτές τις έγκυρες διευθύνσεις θα μπορούσαμε να τις αποθηκεύσουμε εμείς σε μια βάση δεδομένων (MongoDB) ώστε να αλληλεπιδρούν με την εφαρμογή μας ή αν θέλαμε να την κάνουμε ακόμα πιο αποκεντρωμένη την εφαρμογή μας να τις αποθηκεύσουμε σε ένα αποκεντρωμένο σύστημα IPFS (InterPlanetary File System).

Παρατήρηση 2: Το εκλογικό σύστημα των ΗΠΑ διαφέρει πολύ με αυτό που παρουσιάζεται στη δική μας εφαρμογή. Οι πολίτες δεν ψηφίζουν άμεσα τον πρόεδρο που θέλουν, αλλά εκλέκτορες ανά πολιτεία οι οποίοι εκπροσωπούν τα αντίστοιχα κόμματα. Αυτό ενδεχομένως να μπορούσε να επιλυθεί σε μια διαφορετική υλοποίηση της όλης προσέγγισης που κάναμε για το σύστημα ηλεκτρονικής ψηφοφορίας.

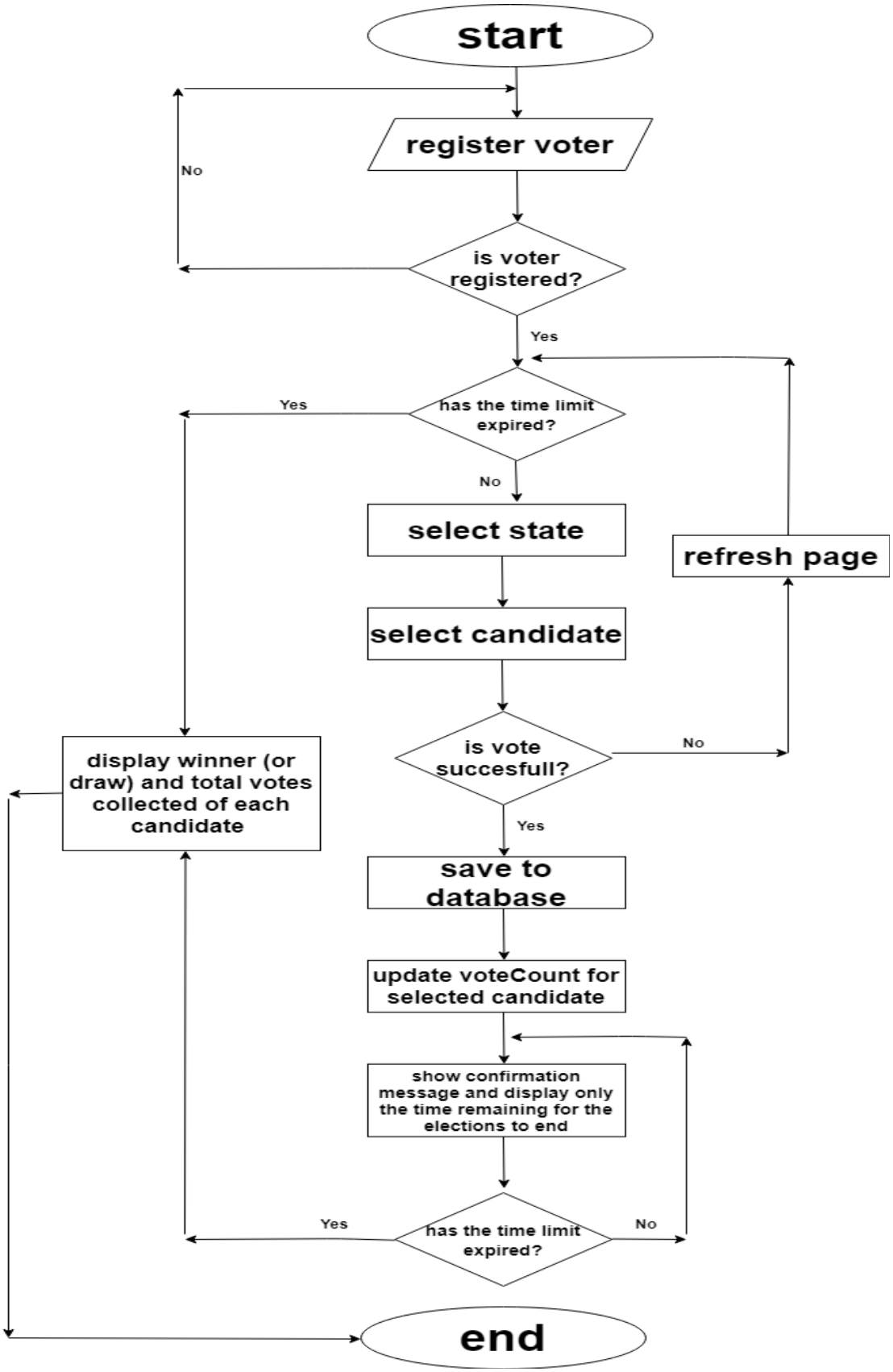
Παρατήρηση 3: Σε πιο πολύπλοκα έξυπνα συμβόλαια με πολλές συναρτήσεις είναι καλό να δηλώνονται, όπου κρίνεται απαραίτητο, οι συναρτήσεις ως προς την ορατότητα τους (visibility) αντί για public ως external. Η διαφορά αυτών των δύο είναι ότι οι public συναρτήσεις αποθηκεύονται στη μνήμη ενώ οι external όχι. Αυτό έχει ως αποτέλεσμα οι external να έχουν χαμηλότερο κόστος (gas) εκτέλεσης σε σχέση με τις public. Στην περίπτωση μας ελέγχθηκαν τα αντίστοιχα κόστη στη συνάρτηση vote και επειδή ήταν αμελητέα η διαφορά στο κόστος επιλέχθηκε να παραμείνει η συνάρτηση ως public.

6.Επίδειξη λειτουργικότητας εφαρμογής

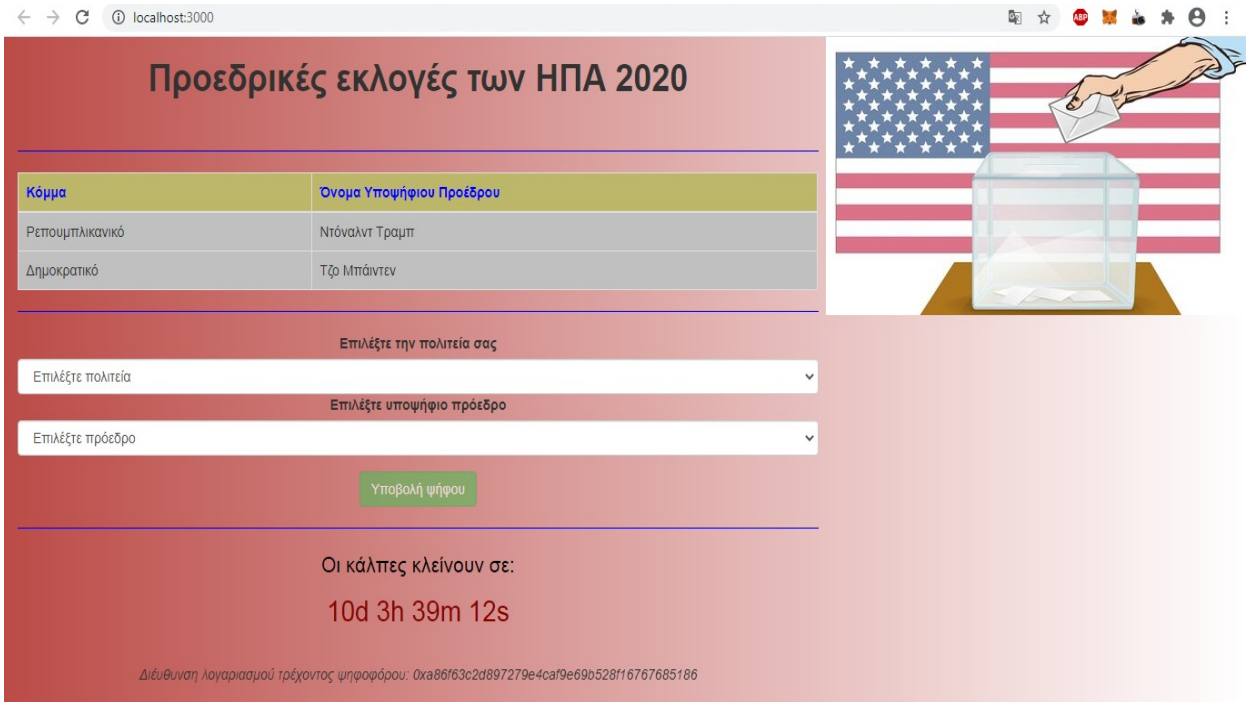
Στο κεφάλαιο αυτό θα γίνει παρουσίαση της εφαρμογής και του τρόπου λειτουργίας της. Θα παρουσιάσουμε δηλαδή την εφαρμογή από την σκοπιά του χρήστη της (ψηφοφόρου), ενώ πριν ασχοληθήκαμε με τον τρόπο υλοποίησης της.

Προτού ξεκινήσει η παρουσίαση αυτή, ας δούμε στο *σχήμα 6.1* ένα διάγραμμα ροής που αντιστοιχεί στην διαδικασία ηλεκτρονικής ψηφοφορίας που σχεδιάσαμε. Να σημειωθεί ότι στο διάγραμμα αυτό περιλαμβάνεται και το κομμάτι της αυθεντικοποίησης του ψηφοφόρου, το οποίο δεν υλοποιείται πρακτικά στην εφαρμογή μας αλλά δίνεται μια θεωρητική περιγραφή επίλυσης του στην Παρατήρηση 1 του κεφαλαίου 5.3.

Η ιστοσελίδα (*Παράρτημα ΙΙ*) η οποία υλοποιεί την εφαρμογή μας είναι μία και έχει επιλεχτεί να είναι όσο πιο απλή γίνεται για να επιτευχθεί έτσι μεγαλύτερη αποκεντροποίηση. Αν ένας χρήστης εισέλθει επιτυχώς στην ιστοσελίδα ηλεκτρονικής ψηφοφορίας θα αντικρίσει την εικόνα στο *σχήμα 6.2*:

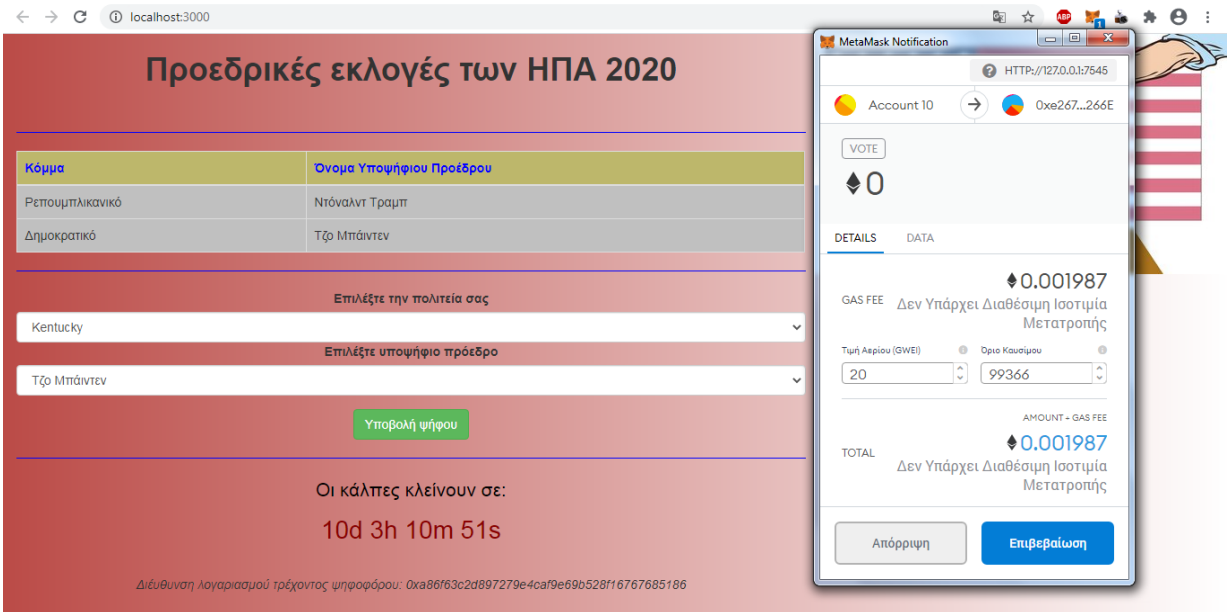


Σχήμα 6.1–Διάγραμμα ροής της εφαρμογής μας

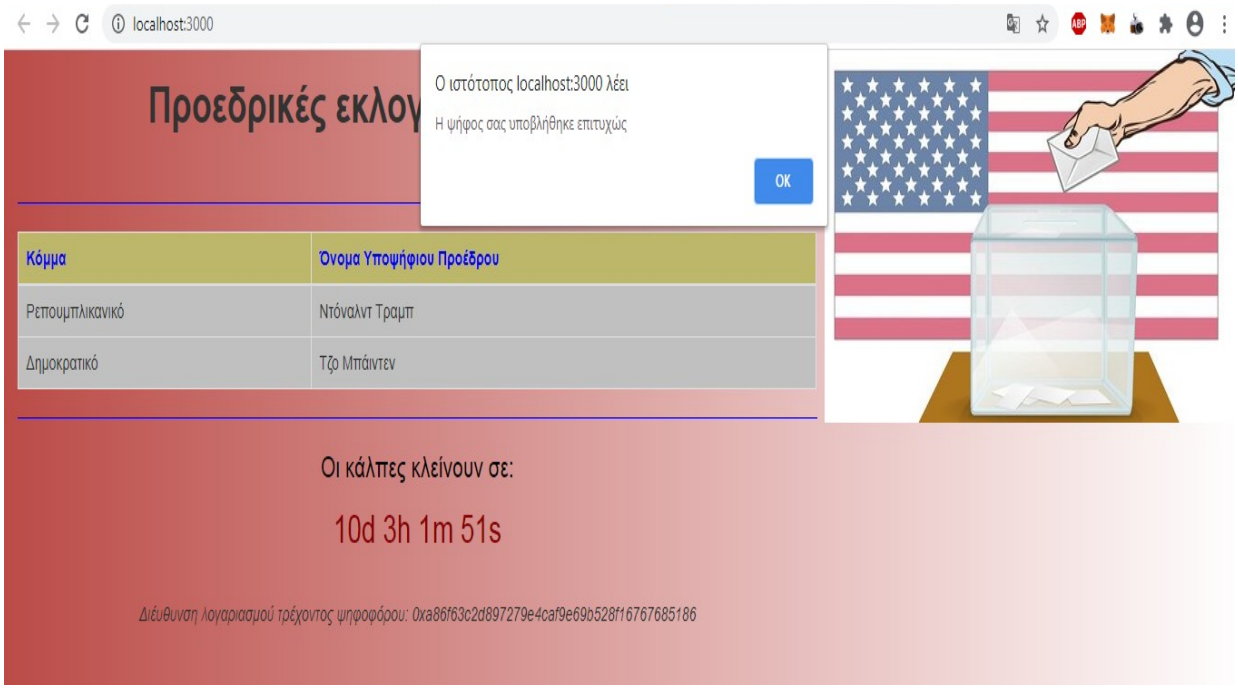


Σχήμα 6.2–Αλληλεπίδραση με την εφαρμογή (1)

Αφού ο χρήστης λοιπόν έχει συνδέσει μέσω MetaMask την Ethereum address του με την ιστοσελίδα μας είναι έτοιμος να υποβάλλει ψήφο. Στο τέλος της ιστοσελίδας αναγράφεται κάθε φορά η διεύθυνση του ψηφοφόρου που εισήλθε στην εφαρμογή, ώστε να επιβεβαιωθεί και οπτικά η σύνδεση με το συμβόλαιο μας. Αν τυχόν η διεύθυνση δεν συμπίπτει με του ψηφοφόρου τότε δε θα μπορέσει να συνεχιστεί η διαδικασία της ψηφοφορίας. Συνεχίζοντας, παρατηρούμε ότι ο ψηφοφόρος εξετάζει μέσω ενός πίνακα τις επιλογές των υποψήφιων προέδρων που έχει και το κόμμα το οποίο εκπροσωπούν. Επίσης λίγο πιο κάτω έχει την δυνατότητα να επιλέξει την πολιτεία στην οποία ανήκουν τα εκλογικά του δικαιώματα και τον πρόεδρο τον οποίο επιθυμεί να στηρίξει. Αν δεν επιλέξει και στα δύο πεδία την αντίστοιχη επιλογή του, τότε το κουμπί υποβολής ψήφου θα παραμένει ανενεργό έως ότου πράξει αναλόγως. Να σημειώσουμε επίσης ότι η αντίστροφη μέτρηση για το κλείσιμο των καλπών που φαίνεται είναι εντελώς τυχαία καθώς εμείς καθορίζουμε τη λήξη της ψηφοφορίας ως προγραμματιστές. Έστω ότι ο ψηφοφόρος μας ανήκει στην πολιτεία του Kentucky και επιλέξει τον Τζο Μπάιντεν ως υποψήφιο πρόεδρο. Πατώντας το κουμπί υποβολή ψήφου η εφαρμογή θα αλληλεπιδράσει με το έξυπνο συμβόλαιο και θα ζητηθεί μέσω του MetaMask η μεταφορά του αντίστοιχου ποσού που είναι αναγκαίο από τον λογαριασμό του ψηφοφόρου στο λογαριασμό του έξυπνου συμβολαίου (*σχήμα 6.3*). Αν τυχόν ο λογαριασμός Ethereum του ψηφοφόρου δεν έχει το κατάλληλο ποσό σε ETH ώστε να καλέσει το έξυπνο συμβόλαιο και να υποβάλλει τη ψήφο, τότε θα ανοίξει το MetaMask όπως παρακάτω, αλλά χωρίς να του δίνει την δυνατότητα να πατήσει “Επιβεβαίωση”. Έστω ότι ο λογαριασμός του χρήστη έχει το κατάλληλο ποσό ως απόθεμα. Τότε πατώντας “Επιβεβαίωση” θα εμφανιστεί στην ιστοσελίδα ένα μήνυμα επιτυχής υποβολής ψήφου (*σχήμα 6.4*). Έκτοτε ο ψηφοφόρος αυτός δεν θα μπορέσει να ξαναψηφίσει, καθώς θα αναγνωρίζεται μέσω της εφαρμογής μας.

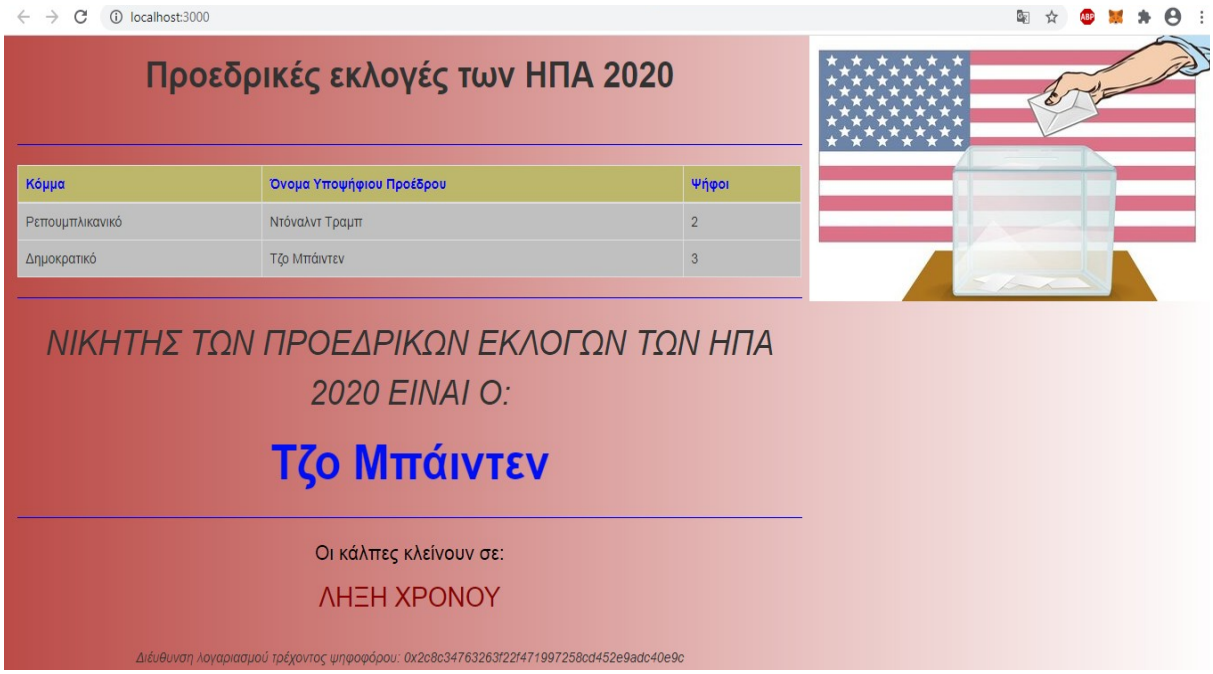


Σχήμα 6.3–Αλληλεπίδραση με την εφαρμογή (2)

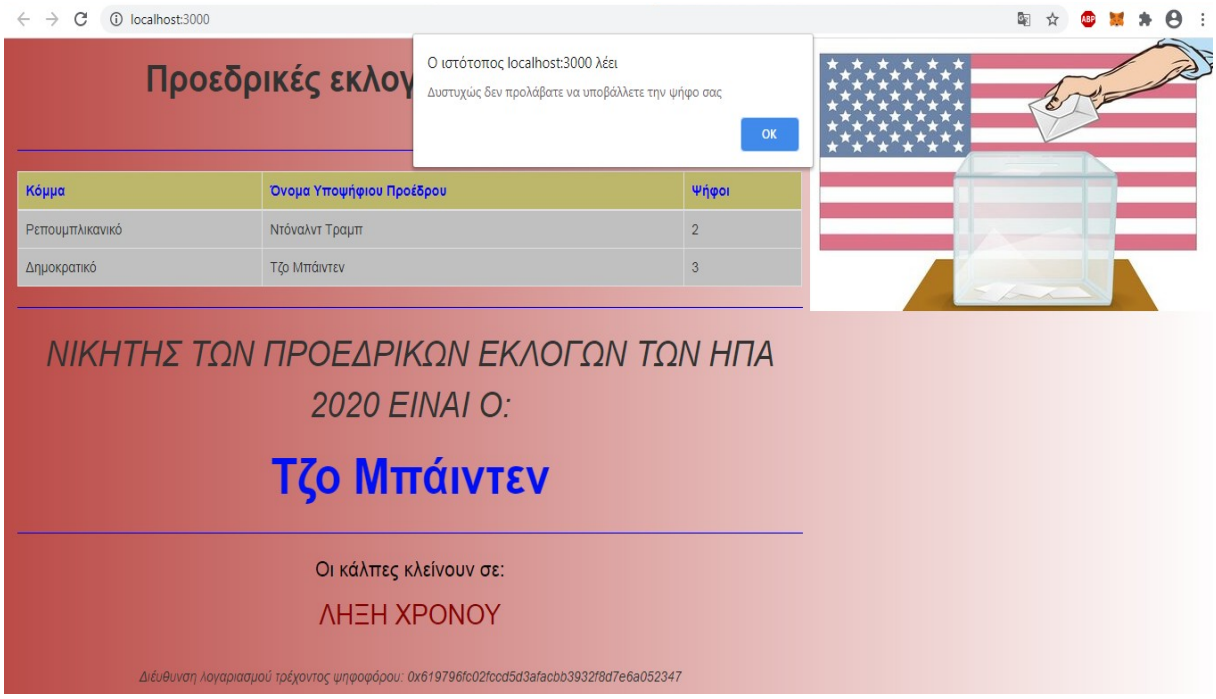


Σχήμα 6.4–Επιτυχής υποβολή ψήφου

Ας υποθέσουμε ότι η ψηφοφορία έφτασε στο τέλος της. Νικητής αναδεικνύεται ο Τζο Μπάιντεν με 3 ψήφους έναντι του Ντόναλντ Τραμπ που συγκέντρωσε 2 ψήφους. Στη περίπτωση που ένας ψηφοφόρος που έχει ήδη ψηφίσει και μπει στην ιστοσελίδα της εφαρμογής εκπρόθεσμα θα αντικρίσει τον νικητή των εκλογών και τις ψήφους που συγκέντρωσε (*σχήμα 6.5*). Να σημειωθεί εδώ ότι αν υπάρξει ισοψηφία αντί του ονόματος του νικητή θα εμφανιστεί "ΙΣΟΨΗΦΙΑ" στην οθόνη του χρήστη. Αν τυχόν επιχειρήσει κάποιος ψηφοφόρος να εισέλθει εκπρόθεσμα στην πλατφόρμα ηλεκτρονικής ψηφοφορίας τότε η εφαρμογή δε θα του επιτρέψει να υποβάλλει ψήφο. Αντί αυτού θα του εμφανιστεί το αντίστοιχο μήνυμα λόγω μη έγκαιρης προσέλευσής του (*σχήμα 6.6*).



Σχήμα 6.5–Αποτελέσματα εκλογών (διεπαφή χρήστη που έχει ήδη ψηφίσει)



Σχήμα 6.6–Αποτελέσματα εκλογών (διεπαφή χρήστη που δεν έχει ψηφίσει)

Όπως έχουμε αναφέρει και προηγουμένως, η εφαρμογή έχει υλοποιηθεί με δοκιμαστικό τρόπο μέσω του private Blockchain που προσφέρει η Ganache. Συνεπώς αναγνωρίζει διευθύνσεις που τρέχουν μόνο στην θύρα 7545 (RPC Server: HTTP://127.0.0.1:7545). Σε ενδεχόμενη αναβάθμιση της εφαρμογής μας ώστε να τρέξει στο Main Net του Ethereum θα πρέπει να γίνουν κάποιες αλλαγές στο κομμάτι κώδικα που αφορά την επικοινωνία μέσω της τεχνολογίας web3. Επιστρέφουμε στην εφαρμογή μας ως έχει. Για να τρέξει η εφαρμογή μας (*Παράρτημα Ι*) θα πρέπει να

έχουμε ρυθμίσει το MetaMask να αναγνωρίζει διευθύνσεις από την θύρα που παρέχει η Ganache, δηλαδή την 7545. Αν αυτό δεν είναι εφικτό, τότε η ψηφοφορία δεν μπορεί να ανοίξει και εμφανίζεται ένας loader, όπως φαίνεται στο παρακάτω *σχήμα 6.7*.

Σχήμα 6.7–Απαίτηση σύνδεσης Ethereum λογαριασμού μέσω MetaMask

Έτσι λοιπόν, αν κάποιος αλληλεπιδράσει ως χρήστης με την εφαρμογή της ηλεκτρονικής ψηφοφορίας που αναπτύχθηκε στο πλαίσιο αυτής της διπλωματικής θα αντικρίσει τα παραπάνω σενάρια. Αφού αναλύσαμε την λειτουργικότητα της, θα την σχολιάσουμε ως προς την αποδοτικότητα της, κλείνοντας αυτό το κεφάλαιο.

Σε αυτό το project λοιπόν, το πεδίο εφαρμογής είναι περιορισμένο για εκλογές μικρής κλίμακας, ακόμη και αν διακυβεύονται θεωρητικά οι εκλογές των ΗΠΑ. Σε μια ψηφοφορία με εκατομμύρια ψηφοφόρους θα εμφανιστούν πολλά και διαφορετικά προβλήματα τα οποία τώρα δεν μπορούμε να τα αντικρίσουμε. Η επεκτασιμότητα του δικτύου Ethereum στο εκλογικό επίπεδο είναι ακόμα άγνωστη και χρειάζεται περαιτέρω έρευνα, γι' αυτό και δεν μπορούμε να προτείνουμε τη χρήση αυτών των συμβολαίων για τις εκλογές μια ολόκληρης χώρας, τουλάχιστον για τώρα.

7.Επίλογος

7.1 Σύνοψη και συμπεράσματα

Σκοπός της παρούσας διπλωματικής εργασίας ήταν η εξέταση των τεχνολογιών αποκέντρωσης που προσφέρει πλέον το Blockchain, και πιο συγκεκριμένα το Ethereum. Μέσω αυτών των τεχνολογιών αναπτύχθηκε ένα έξυπνο συμβόλαιο το οποίο το χρησιμοποιήσαμε έπειτα σε μια αποκεντρωμένη εφαρμογή (Dapp – Decentralized Application) ηλεκτρονικής ψηφοφορίας. Αναμφισβήτητα η εφαρμογή αυτή σε συνδυασμό με τις μελλοντικές βελτιώσεις που θα προτείνουμε στην συνέχεια, θα μπορούσε να αποτελέσει αργότερα κομμάτι του αποκεντρωμένου ιστού και να συνεισφέρει θετικά στην ταχύτατα αναπτυσσόμενη κοινότητα του.

Με την δημιουργία αυτού του προτεινόμενου έξυπνου συμβολαίου μας, καταφέραμε να μεταφέρουμε το σενάριο της ηλεκτρονικής ψηφοφορίας (e-voting) στην Blockchain πλατφόρμα και αντιμετωπίσαμε σε κάποιο βαθμό ορισμένα από τα θεμελιώδη ζητήματα που αντιμετωπίζουν τα παλαιά συστήματα ηλεκτρονικής ψηφοφορίας, χρησιμοποιώντας τη δύναμη του Ethereum δικτύου και τη δομή του Blockchain του. Σαν αποτέλεσμα των δοκιμών μας, η έννοια του Blockchain και η ασφαλή μεθοδολογία που χρησιμοποιεί, δηλαδή αμετάβλητες αλυσίδες κατακερματισμού, προσαρμόστηκαν σε ένα πιθανό σενάριο εκλογών. Να αναφέρουμε εδώ ότι το Ethereum και τα έξυπνα συμβόλαια του, αποτέλεσαν ένα από τα πιο επαναστατικά επιτεύγματα έπειτα από την έλευση του ίδιου του Blockchain (Bitcoin), βοήθησαν στο να ανατραπεί η αντίληψη περιορισμού του Blockchain ως υπόβαθρο απλά ενός κρυπτονομίσματος και το μετέτρεψαν σε μια ευρύτερη βάση λύσεων για πολλά θέματα που σχετίζονται με το Διαδίκτυο στον σύγχρονο κόσμο. Έτσι ήρθαν ακόμα πιο πολλοί χρήστες του Διαδικτύου πιο κοντά με την τεχνολογία του Blockchain.

Η ηλεκτρονική ψηφοφορία εξακολουθεί να είναι αμφιλεγόμενο θέμα και κάνει συνεχώς πολιτικούς και επιστημονικούς κύκλους. Παρά την ύπαρξη μερικών πολύ καλών παραδειγμάτων e-voting, οι περισσότερες προσπάθειες είτε απέτυχαν να παρέχουν την ασφάλεια και τα χαρακτηριστικά απορρήτου των παραδοσιακών εκλογών είτε είχαν σοβαρά ζητήματα χρήσης και επεκτασιμότητας. Αντιθέτως, λύσεις ηλεκτρονικής ψηφοφορίας που βασίζονται στο Blockchain, συμπεριλαμβανομένης αυτής που εμείς έχουμε εφαρμόσει χρησιμοποιώντας τα έξυπνα συμβόλαια και το δίκτυο του Ethereum, διευθύνουν κάποιες από τις ανησυχίες ασφαλείας που προκύπτουν, όπως μυστικότητα των ψηφοφόρων, ακεραιότητα, επαλήθευση και διαφάνεια καταμέτρησης. Ωστόσο, υπάρχουν και ορισμένα ζητήματα που δεν μπορούν να αντιμετωπιστούν αποκλειστικά χρησιμοποιώντας το Blockchain, για παράδειγμα έλεγχος ταυτότητας ψηφοφόρων (στο επίπεδο του ψηφοφόρου ως πολίτη, όχι σε επίπεδο λογαριασμού) και απαιτούνται επιπλέον μηχανισμοί που πρέπει να ενσωματωθούν σε ένα τέτοιο έργο.

7.2 Μελλοντικές επεκτάσεις

Η αποκεντρωμένη εφαρμογή που αναπτύχθηκε, αν και είναι λειτουργική όπως είδαμε στο κεφάλαιο 6, δεν μπορεί να θεωρηθεί προϊόν έτοιμο για να χρησιμοποιηθεί σε πραγματικά δεδομένα ψηφοφορίας μεγάλης κλίμακας. Μερικές επεκτάσεις που θα

την καταστήσουν πιο λειτουργική και πιο εύχρηστη είναι οι εξής:

- **Έλεγχος ταυτότητας ψηφοφόρων σε επίπεδο πολίτη -** Όπως έχουμε αναφέρει πολλές φορές προηγουμένως στην παρούσα διπλωματική εργασία, η εφαρμογή μας δεν επιλύει το κομμάτι της αυθεντικοποίησης του κάθε πολίτη ως νόμιμου ψηφοφόρου. Συνεπώς μπορεί κάποιος "κακόβουλος" χρήστης να χρησιμοποιήσει παραπάνω από ένα λογαριασμό Ethereum ώστε να ψηφίσει. Η θεωρητική λύση που προτείνουμε για την επίλυση αυτού του ζητήματος βρίσκεται στην Παρατήρηση 1 του κεφαλαίου 5.3.
- **Προσαρμογή του εκλογικού μοντέλου στο πραγματικό εκλογικό μοντέλο των ΗΠΑ -** Όπως έχουμε αναφέρει και σε προηγούμενο κεφάλαιο (Παρατήρηση 2 του κεφαλαίου 5.3.) η διαδικασία εκλογής προέδρου στις ΗΠΑ διαφέρει πολύ με αυτήν που παρουσιάζεται στη δική μας εφαρμογή. Σε μια άλλη υλοποίηση της εφαρμογής θα μπορούσε να δοθεί μεγαλύτερη βαρύτητα σε αυτό το κομμάτι. Στα πλαίσια της διπλωματικής εργασίας αυτής επιλέχθηκε να δοθεί μεγαλύτερη προσοχή στο χαρακτηριστικό της αποκέντρωσης που προσφέρει το Ethereum και όχι σε λεπτομέρειες της εκλογικής διαδικασίας.
- **Πλήρης απεξάρτηση από την ανάγκη ύπαρξης ιστοσελίδας για την υλοποίηση της εφαρμογής -** Ανάλογα και με την κατεύθυνση που θα ακολουθήσει στο μέλλον το Ethereum Blockchain, ίσως απαλλαγούμε από την αναγκαστική ύπαρξη μιας ιστοσελίδας ως διεπαφή του χρήστη με την αποκεντρωμένη εφαρμογή. Με αυτόν τον τρόπο κάθε Dapp δε θα έχει καμία απολύτως επαφή με την αρχιτεκτονική πελάτη-εξυπηρετητή (client-server), αλλά θα βασίζεται πλήρως στην αρχιτεκτονική ομότιμων (P2P).
- **Νέες δυνατότητες με τις αλλαγές που είναι προγραμματισμένες για το Ethereum Blockchain -** Η πλατφόρμα του Ethereum είναι σε διαδικασία αλλαγής. Πλησιάζει ο ερχομός του Ethereum 2, το οποίο θα προσδίδει πολλές περισσότερες δυνατότητες στα έξυπνα συμβόλαια, ενώ θα αλλάξει και τελείως ο αλγόριθμος απόδειξης εργασίας, ο οποίος θα μετατραπεί από Proof of Work σε Proof of Stake. Οπότε, όλες οι αποκεντρωμένες εφαρμογές, που είναι χτισμένες στην πλατφόρμα του Ethereum, θα έχουν νέες δυνατότητες, άρα και η συγκεκριμένη εφαρμογή, θα μπορεί να βελτιωθεί με τρόπους άγνωστους τη χρονική περίοδο συγγραφής της διπλωματικής αυτής εργασίας.

Βιβλιογραφία

Διεθνής Βιβλιογραφία

Ameer Rosic, «What is Blockchain Technology? A Step-by-Step Guide For Beginners». 25 June 2019. [Online]. Available: https://blockgeeks.com/guides/what-is-blockchain-technology/

Andreas M.Antonopoulos,G.W,«Mastering Ethereum»,O'Reilly Media,November 2018.

Binance Academy, «Byzantine Fault Tolerance Explained». January 2020. [Online]. Available: https://academy.binance.com/en/articles/byzantine-fault-tolerance-explained

Binance Academy, «History of Blockchain». 21 October 2020. [Online]. Available: https://academy.binance.com/en/articles/history-of-blockchain

Blockchainhub Berlin, «Tokenized Networks: Web3, the Stateful Web». [Online]. Available: https://blockchainhub.net/web3-decentralized-web/

Blockgeeks, «What Are Dapps? The New Decentralized Future». (n.d). [Online]. Available: https://blockgeeks.com/guides/dapps/

Cachin Christian & Vukolić Marko, «Blockchain consensus protocols in the wild». 7 July 2017. [Online]. Available: https://arxiv.org/pdf/1707.01873v2.pdf

Daniel, «What's A Merkle Tree? A Simple Guide To Merkle Trees». 19 July 2018. [Online]. Available: https://komodoplatform.com/whats-merkle-tree/

Ethereum Homestead Documentation, «What is Ethereum». (n.d). [Online]. Available: http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html.

Kiayias Aggelos, «General Challenges of e-Voting», 29 November 2017. [Online]. Available: https://www.ed.ac.uk/files/atoms/files/05_-_kiayias.pdf

L. Lamport, R. Shostak & M. Pease, «The Byzantine general problem», ACM Transactions on Programming languages and Systems, pp. 382-401, 1982.

Leo Arias, «Proof of work». November 2019. [Online]. Available: https://forum.openzeppelin.com/t/proof-of-work/1759

Marius Popa,M.D,B.I, «VOTING DAPP IN EMBEDDED DEVICES USING BLOCKCHAIN TECHNOLOGY AND SECURITY CHALLENGES». May 2019. [Online]. Available: https://www.researchgate.net/publication/335379534_VOTING_DAPP_IN_EMBEDDED_DEVICES_USING_BLOCKCHAIN_TECHNOLOGY_AND_SECURITY_CHALLENGES

Marr Bernard, «A Very Brief History Of Blockchain Technology Everyone Should Read» 16 February 2018. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/02/16/a-very-brief-history-of-blockchain-technology-everyone-should-read/#46370e257bc4

MetaMask, «MetaMask». (n.d). [Online]. Available: https://metamask.io/

npm Docs, «About npm». (n.d). [Online]. Available: https://docs.npmjs.com/about-

npm

OpenJS Foundation, «Introduction to Node.js». (n.d). [Online]. Available: https://nodejs.dev/learn

Network Encyclopedia, «Peer-to-Peer Network (P2P)». (n.d). [Online]. Available: https://networkencyclopedia.com/peer-to-peer-network-p2p/

OpenZeppelin Team,«Proof of stake».15 June 2020. [Online]. Available: https://forum.openzeppelin.com/t/proof-of-stake/3106

Paar Christof,J.P, «SHA-3 and The Hash Function Keccak». 18 June 2013. [Online]. Available: http://professor.unisinos.br/linds/teoinfo/Keccak.pdf

Reddy Shyam Shankar, «Understanding Merkle Tree & Its Importance In Blockchain». 18 May 2020. [Online]. Available: https://www.forex.academy/understanding-merkle-tree-its-importance-in-blockchain/

Richards Sam, «MINING». 22 September 2020. [Online]. Available: https://ethereum.org/en/developers/docs/mining/

Truffle Suite, «Ganache». (n.d). [Online]. Available: https://www.trufflesuite.com/docs/ganache/overview

Truffle Suite, «Truflle». (n.d). [Online]. Available: https://www.trufflesuite.com/docs/truffle/overview

Web3js, «web3.js - Ethereum JavaScript API» (n.d). [Online]. Available: https://web3js.readthedocs.io/en/v1.3.0/

Wikipedia, «CSS». (n.d). [Online]. Available: https://en.wikipedia.org/wiki/CSS

Wikipedia, «Ethereum». (n.d). [Online]. Available: https://en.wikipedia.org/wiki/Ethereum

Wikipedia, «HTML». (n.d). [Online]. Available: https://en.wikipedia.org/wiki/HTML

Wood Gavin, «ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER». 5 September 2020. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf

Ελληνική Βιβλιογραφία

Burmester Mike,Σ.Γ,Σ.Κ,Β.Χ,«Σύγχρονη Κρυπτογραφία: Θεωρία και Εφαρμογές», Εκόδσεις Παπασωτηρίου,2011,Αθήνα.

Κουκουβίνος Χρήστος,Α.Π,«Κρυπτογραφία»,Έκδοση Ε.Μ.Π,2007,Αθήνα.

Cyprus Company, «Τι είναι το Blockchain και πως λειτουργεί». (n.d). [Online]. Available: https://www.cypruscompanies.gr/article/en/89/

Παράρτημα Ι: Εγχειρίδιο χρήσης

Στο παράρτημα αυτό θα γίνει παρουσίαση των οδηγιών-βημάτων που χρειάζεται να ακολουθήσει ένας μέσος χρήστης ώστε να χρησιμοποιήσει την αποκεντρωμένη εφαρμογή της παρούσας διπλωματικής σε λειτουργικό σύστημα Windows. Ο κώδικας του έξυπνου συμβολαίου, των ιστοσελίδων και όλων απαραίτητων αρχείων που χρειάζεται να έχει στη διάθεση του ο χρήστης βρίσκονται αναρτημένα στην σελίδα: https://github.com/stefanosmav/Diplomathesis_dapp. Το παράρτημα αυτό χωρίζεται στο κομμάτι της εγκατάστασης μερικών απαιτούμενων εφαρμογών-εργαλειών και στο κομμάτι της χρήσης της εφαρμογής.

Εγκατάσταση

Εδώ θα παραθέσουμε τις οδηγίες που είναι απαραίτητες να γίνουν με την παρακάτω σειρά ώστε να εγκατασταθούν κάποια απαραίτητα εργαλεία (Κεφάλαιο 4) για τη χρήση της αποκεντρωμένης εφαρμογής.

Βήμα 1

Κατεβάζουμε και εγκαθιστούμε το περιβάλλον ανάπτυξης Node.js από την ιστοσελίδα: https://nodejs.org/en/download/. Το npm διανέμεται μαζί με το Node.js που σημαίνει ότι όταν γίνεται λήψη του Node.js, εγκαθίσταται αυτόματα το npm στον υπολογιστή. Ανάλογα με το λειτουργικό σύστημα του υπολογιστή μας, επιλέγουμε και το αντίστοιχο αρχείο εγκατάστασης από τη λίστα. Είναι σημαντικό να επιλέξουμε το αρχείο που παρέχεται στην καρτέλα LTS, διότι τα υπόλοιπα δεν έχουν ελεγχθεί μαζί με το npm. Επίσης να σημειωθεί ότι αν δεν χρησιμοποιείται την τελευταία έκδοση των Windows (π.χ Windows 7) ενδέχεται να χρειαστεί να κατεβάσετε κάποια προηγούμενη έκδοση του Node.js.

Βήμα 2

Κατεβάζουμε και εγκαθιστούμε με τις προτεινόμενες ρυθμίσεις το σύστημα ελέγχου git από την ιστοσελίδα: https://git-scm.com/. Τη στιγμή συγγραφής της παρούσας διπλωματικής η τελευταία έκδοση του git ήταν η 2.29.2.

Βήμα 3

Κατεβάζουμε και εγκαθιστούμε το framework της σουίτας Truffle ανοίγοντας ένα terminal και γράφοντας την εξής εντολή:

```
>npm install truffle -g
```

Βήμα 4

Κατεβάζουμε και εγκαθιστούμε το Ganache από την ιστοσελίδα: https://www.trufflesuite.com/ganache. Όποτε χρησιμοποιούμε το Ganache επιλέγουμε το έτοιμο Workspace Quickstart που παρέχεται μαζί με 10 λογαριασμούς Ethereum ή δημιουργούμε ένα δικό μας Workspace .

Βήμα 5

Εγκαθιστούμε το MetaMask plugin στον browser που χρησιμοποιούμε. Για το Google Chrome είναι διαθέσιμο στο παρακάτω link: https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn. Έπειτα απαιτείται η δημιουργία ενός λογαριασμού στο Metamask το οποίο γίνεται πολύ εύκολα. Είναι σημαντικό (αν θέλετε να χρησιμοποιείται τακτικά το

Metamask) να αποθηκεύσετε κάπου που μόνο εσείς έχετε πρόσβαση τις λέξεις φύτρου που έχουν αποδοθεί στον λογαριασμό σας. Όπως έχουμε ξαναπεί το MetaMask είναι στην ουσία μία "γέφυρα" η οποία δίνει στον browser πρόσβαση στις αποκεντρωμένες εφαρμογές (Dapps). Επειδή η αποκεντρωμένη εφαρμογή μας τρέχει δοκιμαστικά στο δίκτυο Blockchain που παρέχεται μέσω του Ganache στην θύρα 7545 και όχι στο Main Net του Ethereum πρέπει να δημιουργήσουμε την "γέφυρα" που αναφέραμε προηγουμένως. Ανοίγουμε το MetaMask και στο πάνω μέρος που αναγράφεται το τρέχον δίκτυο επικοινωνίας Blockchain (συνήθως είναι το Main Net) επιλέγουμε την επιλογή "Προσαρμοσμένο RPC". Αντιγράφουμε από το Ganache τα πεδία RPC SERVER (συνήθως HTTP://127.0.0.1:7545) και NETWORK ID (συνήθως 5777 ή 0x539) και τα κάνουμε επικόλληση στο MetaMask στα πεδία "Νέο RPC URL" και "Αναγνωριστικό αλυσίδας" αντίστοιχα. Αν θέλουμε εισάγουμε και ένα οποιοδήποτε όνομα στο δίκτυο και πατώντας Αποθήκευση η "γέφυρα επικοινωνίας" ανάμεσα στο Ganache και το MetaMask έχει πλέον αποκατασταθεί. Να σημειώσουμε εδώ οτι το Ganache πρέπει να είναι μονίμως ενεργό. Πλέον είμαστε σε θέση να εισάγουμε έναν τυχαίο λογαριασμό στο MetaMask από το Ganache χρησιμοποιώντας μόνο το ιδιωτικό κλειδί του. Πατώντας το εικονίδιο του λογαριασμού μας και έπειτα "Εισαγωγή Λογαριασμού", εισάγουμε το ιδιωτικό κλειδί και έτσι φορτώσαμε ένα λογαριασμό Ethereum στο MetaMask.

Χρήση εφαρμογής

Βήμα 1

Ανοίγω το Ganache, επιλέγω να μου παρέχει αυτόματα τις ρυθμίσεις του δικτύου Blockchain, τους λογαριασμούς Ethereum κλπ. μέσω του Quickstart Workspace ή ενός δικού μας Workspace και το έχω μονίμως ανοιχτό καθ όλη τη διάρκεια χρήσης της εφαρμογής.

Βήμα 2

Ανοίγω ένα terminal. Εκτελώ την εντολή:

```
>git clone https://github.com/stefanosmav/Diplomathesis_dapp
```

Αυτό δημιουργεί ένα φάκελο (στο directory που έχει καθοριστεί στο terminal) με όνομα Diplomathesis_dapp που περιέχει όλα τα αρχεία κώδικα της αποκεντρωμένης εφαρμογής μας.

Βήμα 3

Έπειτα στο terminal εκτελώ τις παρακάτω εντολές:

```
>cd Diplomathesis_dapp
>truffle migrate
```

Επιλέγουμε τώρα ως directory του project μας τον φάκελο Diplomathesis_dapp και εκτελούμε compile και deploy στο έξυπνο συμβόλαιο μας ώστε να δημιουργηθεί το αντίστοιχο *.json αρχείο που θα αλληλεπιδρά με την web app μας. Ό,τι αλλαγή θέλουμε να κάνουμε στο έξυπνο συμβόλαιο μας πρέπει να γίνει πριν από το deploy του, καθώς μετά είναι αδύνατο λόγω της ιδιότητας των smart contracts να μην αλλάζουν αφού υλοποιηθούν. Έτσι έχουμε δημιουργήσει ουσιαστικά την ψηφοφορία μας. Από εδώ και πέρα όποιος ψηφίζει θα κατοχυρώνεται η ψήφος του και θα αποθηκεύεται στη μνήμη του έξυπνου συμβολαίου.

Βήμα 4

Ανοίγω το UI (User Interface) της εφαρμογής μου εκτελώντας στο terminal την

εντολή:

```
>npm run dev
```

Αυτή η εντολή θα μου ανοίξει αυτόματα την ιστοσελίδα της εφαρμογής (*Παράρτημα II*) συνήθως στην τοποθεσία: http://localhost:3000/. Αυτόματα η ιστοσελίδα αλληλεπιδρά με το MetaMask και το ανοίγει στον browser. Απαιτείται να κάνουμε σύνδεση στον λογαριασμό MetaMask που δημιουργήσαμε και να επιλέξουμε το δίκτυο που δημιουργήσαμε πριν στο Βήμα 5 των οδηγιών εγκατάστασης. Εισάγουμε έναν λογαριασμό μέσω του ιδιωτικού του κλειδιού και τον συνδέουμε με την ιστοσελίδα της εφαρμογής μας. Πλέον ο χρήστης είναι σε θέση να ψηφίσει και γενικά να αλληλεπιδράσει με την εφαρμογή.

Παρατήρηση: Στο βήμα 3 ουσιαστικά δίνουμε στο χρήστη να ξεκινήσει μόνος του την διαδικασία της ψηφοφορίας, αφού κάνει ο ίδιος compile και deploy το έξυπνο συμβόλαιο. Σε μια ενδεχόμενη μελλοντική χρήση της εφαρμογής ο χρήστης με το βήμα 2 θα έχει στη διάθεση του και έτοιμο το *.json αρχείο (φάκελος build) που δημιουργείται σε εμάς εδώ στο βήμα 3. Έτσι η δεύτερη εντολή στο βήμα 3 θα παραλείπεται εντελώς και δεν θα επιτρέπεται σε κανέναν χρήστη πέραν του δημιουργού της ψηφοφορίας (προγραμματιστή) να κάνει compile και deploy τα έξυπνα συμβόλαια και γενικά να επεμβαίνει στα αρχεία (only readable) που του έχουν δοθεί μέσω του github.

Παράρτημα ΙΙ: Ιστοσελίδα

Σε αυτό το παράρτημα βρίσκεται ο κώδικας της ιστοσελίδας γραμμένος στην γλώσσα HTML.

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <!-- The above 3 meta tags *must* come first in the head; any other head content
must come *after* these tags -->
  <title>United States Presidential Election Results 2020</title>
  <!-- Bootstrap and other css -->
  <link href="css/bootstrap.min.css" rel="stylesheet" />
  <link href="css/hr.css" rel="stylesheet" />
  <link href="css/load.css" rel="stylesheet" />
  <style>
   body {
    background: url(images/rsz_2election.jpg) no-repeat right top,
     linear-gradient(to right, #bb4b47, #ffffff);}
  </style>
 </head>
 <body>
  <center>
   <div class="container-fluid">
    <div class="row">
```

```
<div class="col-lg-8">
 <h1 style="text-align: center; font-weight: bold">
  Προεδρικές εκλογές των ΗΠΑ 2020
 </h1>
 <br />
 <hr />
 <div id="loader">
  <br /><br /><br /><br /><br /><br /><br /><br /><br />
  <p
   class="text-center"
   style="text-align: center; font-size: large">
   <b>Φόρτωση...</b>(απαιτείται σύνδεση λογαριασμού μέσω Metamask)
  </p>
  <div class="load"></div>
 </div>
 <div id="content" style="display: none">
  <table id="pinakas1" class="table table-bordered">
   <thead style="color: blue; background-color: darkkhaki">
    <tr>
     <th scope="col">Κόμμα</th>
     <th scope="col">Όνομα Υποψήφιου Προέδρου</th>
     <th scope="col">Ψήφοι</th>
    </tr>
   </thead>
   <tbody
    style="background-color: silver"
    id="candidatesResults"
   ></tbody>
  </table>
  <table id="pinakas2" class="table table-bordered">
   <thead style="color: blue; background-color: darkkhaki">
    <tr>
     <th scope="col">Κόμμα</th>
     <th scope="col">Όνομα Υποψήφιου Προέδρου</th>
    </tr>
   </thead>
   <tbody
    style="background-color: silver"
    id="candidatesResults2"
   ></tbody>
  </table>
 <hr />
  <form onSubmit="App.submitVote(); return false;">
   <div class="form-group">
    <div id="winner">
     <p style="font-size: 250%; font-style: italic">
      ΝΙΚΗΤΗΣ ΤΩΝ ΠΡΟΕΔΡΙΚΩΝ ΕΚΛΟΓΩΝ ΤΩΝ ΗΠΑ 2020 ΕΙΝΑΙ Ο:
     </p>
     <p
```

```
          id="candidatesWinner"
          style="
           font-size: 350%;
           font-weight: bold;
           color: rgb(0, 16, 243); "></p>
        </div>
        <label for="voterState">Επιλέξτε την πολιτεία σας</label>

        <select
         class="form-control"
         id="voterState"
         onchange="App.selecState(this)"
        >
         <option selected="true" disabled="disabled">
          Επιλέξτε πολιτεία
         </option>
        </select>

        <label for="candidatesSelect"
         >Επιλέξτε υποψήφιο πρόεδρο</label
        >
        <select
         class="form-control"
         id="candidatesSelect"
         onchange="App.selecPres(this)"
        ></select>
       </div>
       <div class="text-center" id="no-">
        <p id="_time">
         <button
          id="koumpi"
          type="submit"
          class="btn btn-success"
          disabled="disabled"
         >
          Υποβολή ψήφου
         </button>
        </p>
       </div>
       <hr />
      </form>
      <p style="font-size: 150%; color: black">
       Οι κάλπες κλείνουν σε:
      </p>
      <p style="color: darkred; font-size: 200%" id="time"></p>
      <br />
      <p
       id="accountAddress"
       class="text-center"
```

```
       style="font-style: italic"
      ></p>
     </div>
    </div>
   </div>
  </div>
  <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
jquery.min.js"></script>
  <!-- Include all compiled plugins (below), or include individual files as needed
-->
  <script src="js/bootstrap.min.js"></script>
  <script src="js/web3.min.js"></script>
  <script src="js/truffle-contract.js"></script>
  <script src="js/time.js"></script>
  <script src="js/app.js"></script>
 </center>
</body>
</html>
```