UNIVERSITY OF WEST ATTICA
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

Τμ. Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής

Διπλωματική Εργασία

Ιούλιος 2022

Σταματάκης Δημήτριος 45544

Ιδιοκτησία αρχείων μέσω της χρήσης NFT σε blockchain

Επιβλέπουσα: Ελένη Αικατερίνη Λελίγκου Αναπληρωτρια Καθηγήτρια Τομέας Ηλεκτρονικού Αυτοματισμού και Τηλεματικής

Ιδιοκτησία αρχείων μέσω της χρήσης

NFT σε blockchain

Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

Α/α	ΟΝΟΜΑ ΕΠΩΝΥΜΟ	ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ	ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ
	ΛΕΛΙΓΚΟΥ ΑΙΚΑΤΕΡΙΝΗ ΕΛΕΝΗ	ΑΝΑΠΛΗΡΩΤΡΙΑ ΚΑΘΗΓΗΤΡΙΑ	Eleni Aikaterini Leligkou Digitally signed by Eleni Aikaterini Leligkou Date: 2022.07.25 10:15:43 +03'00'
	ΠΑΛΛΗΣ ΕΥΑΓΓΕΛΟΣ	ΕΞΕΤΑΣΤΗΣ ΚΑΘΗΓΗΤΗΣ	Evangelos Pallis Digitally signed by Evangelos Pallis Date: 2022.07.25 10:36:13 +03'00'
	ΠΑΠΟΥΤΣΙΔΑΚΗΣ ΜΙΧΑΗΛ	ΕΞΕΤΑΣΤΗΣ ΚΑΘΗΓΗΤΗΣ	Michail Papoutsidakis Digitally signed by Michail Papoutsidakis Date: 2022.07.25 12:05:25 +03'00'

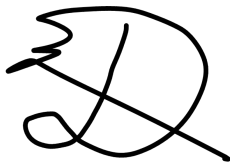
Δήλωση Συγγραφέα Διπλωματικής Εργασίας

Ο κάτωθι υπογεγραμμένος *Σταματάκης Δημήτριος* του *Νεκτάριου*, με αριθμό μητρώου *71445544* φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής *Μηχανικών* του Τμήματος *Μηχανικών Βιομηχανικής Σχεδίασης και Παραγωγής*, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Σταματάκης Δημήτριος



Ευχαριστίες

Το πρώτο άτομο που θα ήθελα να ευχαριστήσω θερμά είναι ο Ιωάννης Χριστίδης, διότι ήταν συνεχώς διπλά μου καθ’ όλη την διάρκεια εκπόνησης της διπλωματικής μου εργασίας. Παρακολουθώντας την πρόοδο μου και παρέχοντάς μου όλη την καθοδήγηση που χρειαζόμουν.

Επίσης, θέλω να ευχαριστώ την κυριά Ελένη Αικατερίνη Λελίγκου που δέχθηκε να συνεργαστεί μαζί μου και μου έδωσε την ευκαιρία να ασχοληθώ και να ανακαλύψω ένα νέο συναρπαστικό κλάδο στο τομέα του προγραμματισμού.

Συνεχίζοντας, θα ήθελα να θερμά να ευχαριστήσω τον Δρ. Δημήτριο Κόγια που μου επέτρεψε να ενταχθώ σε μια κοινότητα ανθρώπων που μοιράζονται το ενδιαφέρον μου για την συγκεκριμένη τεχνολογία, καθώς όλοι μας επιθυμούμε να τη μελετήσουμε και να την εφαρμόσουμε.

Τέλος, θέλω να ευχαριστήσω με όλη μου την καρδιά την οικογένεια και τους φίλους μου, που πραγματικά ήταν διπλά μου σε κάθε δύσκολη στιγμή και χωρίς την υποστήριξη τους δεν θα είχα φτάσει ποτέ στο σημείο που βρίσκομαι σήμερα.

Περίληψη

Η τεχνολογία blockchain χρησιμοποιήθηκε σε μεγάλη κλίμακα για πρώτη φορά από τον Satoshi Nakamoto, δημιουργό του Bitcoin το 2009. Καθώς γινόταν όλο ένα και πιο δημοφιλής άρχισε να κινεί το ενδιαφέρον αρκετών πρωτοπόρων που πίστευαν ότι αυτή η τεχνολογία μπορεί να προσφέρει πολλά περισσότερα από απλός έναν νέο τρόπο οικονομικών συναλλαγών. Γι' αυτόν το σκοπό το 2013 ο Vitalik Buterik μαζί με άλλους τέσσερεις συνεργάτες ξεκίνησαν το project Ethereum που θα έφερνε την επόμενη επανάσταση στο χώρο του blockchain.

Το Ethereum blockchain ξεκίνησε να λειτουργεί το 2015 και πρόσφερε την δυνατότητα στους χρήστες να δημιουργήσουν δίκες τους περίπλοκες εφαρμογές. Αυτό έδωσε ζωή σε μια πληθώρα νέων κλάδων στο κόσμο των κρυπτονομισμάτων. Μερικοί απ' αυτούς είναι η αποκεντρωμένη οικονομία (DeFi) και οι αποκεντρωμένοι αυτόνομοι οργανισμοί (DAOs).

Σήμερα, υπάρχουν σχεδόν 20.000 διαφορετικά κρυπτονομίσματα καθώς και πολλά δημοφιλή blockchains τα οποία είναι παρόμοια με το Ethereum αλλά το καθένα προσπαθεί να λύσει διαφορετικά προβλήματα που εμφανίζονται στο χώρο του blockchain.

Σε αυτήν την εργασία παρουσιάζονται τα βασικά στοιχεία που χαρακτηρίζουν κάθε blockchain δίκτυο καθώς και πως αυτά λειτουργούν. Στην συνέχεια, θα μάθουμε για την κρυπτογραφία μέσω μιας σύντομης αναφοράς στους αλγορίθμους που αφήσαν ιστορία στο συγκεκριμένο κλάδο καθώς και στους αλγορίθμους που χρησιμοποιούνται σήμερα. Στο τέλος του θεωρητικού σκέλους, θα δούμε τα 3 σημαντικότερα πρότυπα κρυπτονομισμάτων που χρησιμοποιούνται ευρέως από όλα τα σύγχρονα δίκτυα blockchains.

Στο πρακτικό σκέλος, γίνεται αναφορά σε διαφορά μοντέρνα εργαλεία, τεχνολογίες και μεθόδους που αφορούν τη δημιουργία και παράταση έξυπνων συμβολαίων. Αυτά στη συνέχεια αναλύονται και χρησιμοποιούνται για τη δημιουργία μιας απλής αποκεντρωμένης εφαρμογής που βασίζεται κατά κύριο λόγο στην γλώσσα Solidity, στο framework Hardhat και στην βιβλιοθήκη ethers.js. Στο τέλος, με την δύναμη της JavaScript και του Node.js, θα δούμε πως μπορούμε να αυτοματοποιήσουμε μερικές χρονοβόρες διαδικασίες για να αυξήσουμε την παραγωγικότητα μας και την ανθεκτικότητα του κώδικα μας.

Summary

The blockchain technology was initially utilized in 2009 by Satoshi Nakamoto, inventor of Bitcoin. Due to its increasing popularity it started to attract the attention of a great number of technology pioneers who believed in the new technology being able to offer much more than simply a new way of making financial transactions. To this end, Vitalik Buterik along with four partners started the Ethereum project in 2013 which was destined to start a revolution in the field of blockchain technology.

The Ethereum blockchain started operating in 2015 and offered its users the opportunity to create their own sophisticated applications. This actualized a plethora of new branches in the realm of the cryptocurrency world. To name some of those, Decentralized Finances (DeFi) and Decentralized Autonomous Organizations (DAOs).

Today, around 20.000 cryptocurrencies are in use as well as a great number of blockchains which are similar to Ethereum but each one of them aims at solving one of the many different problems that emerge in the field of blockchain technology.

In the present thesis there is a presentation of the fundamental elements that characterize every blockchain network as well as how they operate. The second part is dedicated to cryptography, followed by a short but thorough reference to some historic algorithms as well as to those that are used today. The theoretical part culminates with a reference to the three most significant cryptocurrency prototypes that are widely used by all modern blockchain networks.

The practical section concentrates on modern tools, technologies and methods that are used for the creation and deployment of smart contracts. These are further analyzed and used for the creation of a simple decentralized application based mainly on the Solidity language, the Hardhat framework and the ethers.js library. Finally, there is a presentation of how we can automate some time-consuming procedures by using the power of JavaScript and Node.js in order to increase productivity and the resilience of our code.

Περιεχόμενα

Α. Θεωρητικό Μέρος

1. Τεχνολογία Blockchain

1.1 Εισαγωγή

Το Blockchain, είναι μια βάση δεδομένων που αποθηκεύει συναλλαγές, είναι ένας αποκεντρωμένος τρόπος διαχείρισης συναλλαγών καθώς και της επικύρωσης τους μεταξύ ενός μεγάλου αριθμού συμμετεχόντων, γνωστοί ως κόμβοι [3], [4].

Σύμφωνα με μια μελέτη που διεξήχθη από τον [4], το blockchain μπορεί να ταξινομηθεί ως ένα είδος τεχνολογίας κατανεμημένου καθολικού που παρέχει εμπιστοσύνη στον χρήστη ότι οι πληροφορίες που αρχειοθετούνται, όπως για παράδειγμα πιστοποιητικά, δεν πρόκειται να παραβιαστούν ή να αλλοιωθούν.

Διάφορες μελέτες έχουν δείξει ότι το blockchain έχει την ικανότητα να μειώσει:

1 Τη συναλλακτική διαφάνεια

2 Τις ανασφαλείς καταστάσεις

3 Τις αμφιβολίες

Παρέχοντας πλήρη γνωστοποίηση όλων συναλλαγών στους συμμετέχοντες του δίκτυού [5]. Επιπλέον, η τεχνολογία blockchain αναμένεται να ανακαινίσει τις οικονομίες και την κοινωνική τάξη μέσω της μείωσης του κόστους, επειδή παρέχει πιο οικονομικές συναλλαγές και δεν χρειάζεται καλά αναγνωρισμένα και αξιόπιστα τρίτα μέλη για να λειτουργήσει [6], [7].

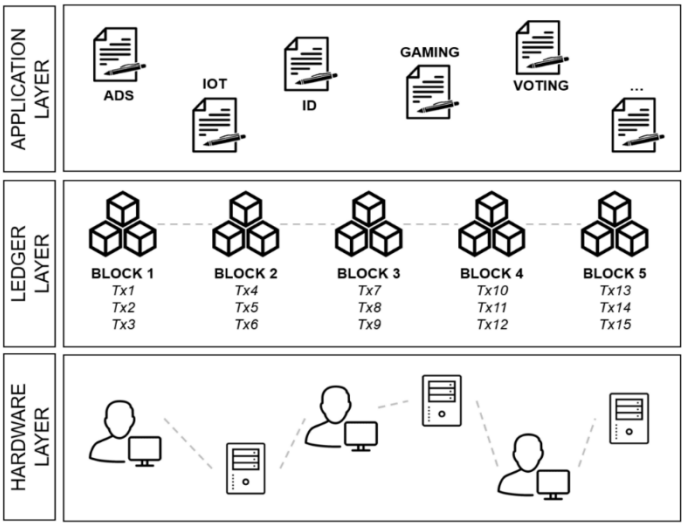
Επιπλέον, μια έρευνα από τον [8] δήλωσε ότι αυτή η τεχνολογία μπορεί να χρησιμοποιηθεί για την καταγραφή συναλλαγών, την αποθήκευση ιατρικών αρχείων, δεσμευτικές συμφωνίες, παρακολούθηση της κυκλοφορίας των εμπορευμάτων, αποθήκευση μεμονωμένων αρχείων πίστωσης, παρακολούθηση της εξέλιξης σε έργα τέχνης και επαλήθευση πληρωμών με τη βοήθεια της προμηθευτικής αλυσίδας και πολλές άλλες διαδικασίες.

Η κατανόηση της **αρχιτεκτονικής του blockchain** είναι το πρώτο βήμα για να μπορέσει κάνεις να αποκτήσει μια σφαιρική εικόνα της συγκεκριμένης τεχνολογίας, με σκοπό να κατανοήσει τα κεφάλαια που ακολουθούν. Το Blockchain μπορεί να θεωρηθεί ως ένα είδος πληροφορικής αρχιτεκτονικής, που αποτελείται από **τρία επίπεδα(layers)** [26]:

- **Επίπεδο 1ο** (Hardware Layer): Είναι το άθροισμα όλων των κόμβων που χρησιμοποιούνται για την λειτουργία ενός δικτύου. Χρησιμοποιώντας την υπολογιστική τους ισχύ, περνούν μέρος στη διαδικασία της συναίνεσης με σκοπό την επικύρωση ή απόρριψη νέων συναλλαγών. Ακόμα διατηρούν αποθηκευμένο ολόκληρο το ιστορικό των συναλλαγών που έχουν πραγματοποιηθεί έως εκείνη την στιγμή στο εν λόγω δίκτυο blockchain.

- **Επίπεδο 2ο** (Ledger Layer): Η τεχνολογία blockchain βασίζεται σε ένα διαμοιρασμένο καθολικό. Οι συναλλαγές καθώς και τα μπλόκς που τις περιέχουν ανήκουν σε αυτό το επίπεδο.

- **Επίπεδο 3ο** (Application Layer): Οι εφαρμογές, ιστοσελίδες ή οτιδήποτε αλλά προϊόντα που έχουν σκοπό την παραγωγή αξίας, ανήκουν εδώ.



Εικόνα 1.1.1 - *Τα τρία επίπεδα της αρχιτεκτονικής ενός Blockchain.*

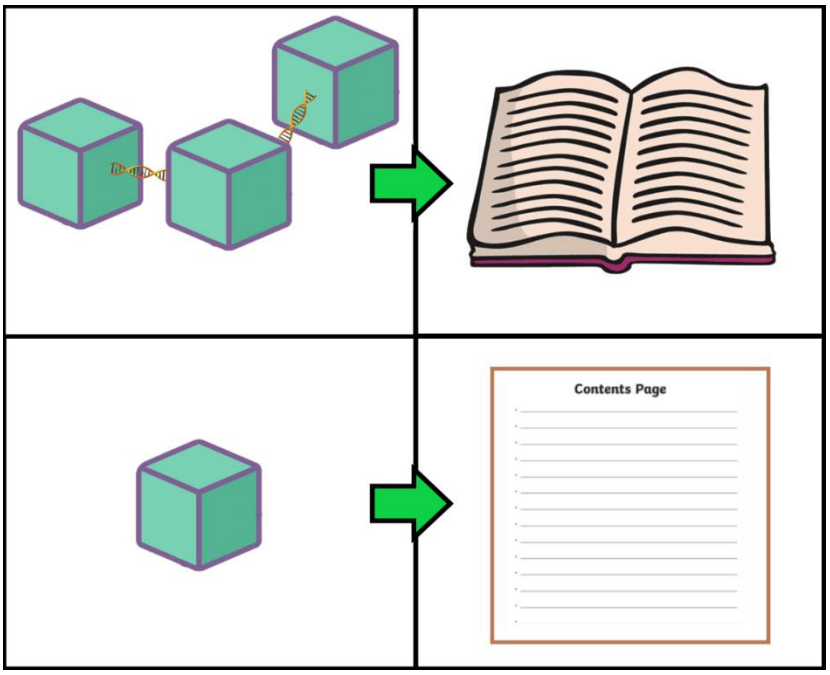
Διαθέσιμο στο [26] σελ. 3 (Figure 1)

1.2 Παρομοίωση Blockchain με Βιβλίο

Στην πιο απλή μορφή, ένα blockchain είναι ένα σύστημα υπολογιστών (κόμβων) και ένα μέρος αποθήκευσης συναλλαγών[1]. Ένα blockchain αποτελείται από δεδομένα που έχουν οργανωθεί σε μονάδες ή πιο συγκεκριμένα σε μπλόκς.

Κάθε **μπλόκ** μοιάζει με μια σελίδα από ένα λογιστικό βιβλίο. Κάθε "σελίδα" είναι αριθμημένη και είναι πολύ εύκολο να εντοπιστεί εάν κάποια σελίδα (ή μπλοκ) έχει διαγραφεί κοιτάζοντας τους αριθμούς των σελίδων, διότι δεν θα υπάρχει συνέχεια. Κάθε «σελίδα» έχει υλικό (συναλλαγές) οργανωμένο με λογική και χρονολογική σειρά. Οι πληροφορίες κρυπτογραφούνται με χρήση κρυπτογραφικών αλγορίθμων για να διασφαλιστεί ότι το απόρρητο του χρήστη δεν διακινδυνεύει και τα δεδομένα δεν μπορούν να αλλοιωθούν.

Οι σελίδες είναι όλες ραμμένες μεταξύ τους για να κάνουν ένα μόνο βιβλίο. Και, όπως με κάθε βιβλιοθήκη, μπορεί να υπάρχει μεγάλος αριθμός διαφορετικών βιβλίων, δηλαδή μπορούν να υπάρχουν πολλά διαφορετικά είδη blockchain.



Εικόνα 1.2.1 - *Παρομοίωση Blockchain με Βιβλίο. Δημιουργημένο από τον συγγραφέα*

1.3 Αρχιτεκτονική των Μπλόκς

1.3.1 Χαρακτηριστικά ενός Μπλοκ

Κάθε **μπλόκ** συνδέεται με το προηγούμενο με λογική σειρά. Με αυτόν τον τρόπο δημιουργείται μια αλυσίδα από μπλόκς. Οι χρήστες δεν μπορούν να αφαιρέσουν ένα μεμονωμένο μπλόκ χωρίς να αλλοιώσουν ολόκληρη την αλυσίδα από την αρχή έως το συγκεκριμένο μπλόκ[1].

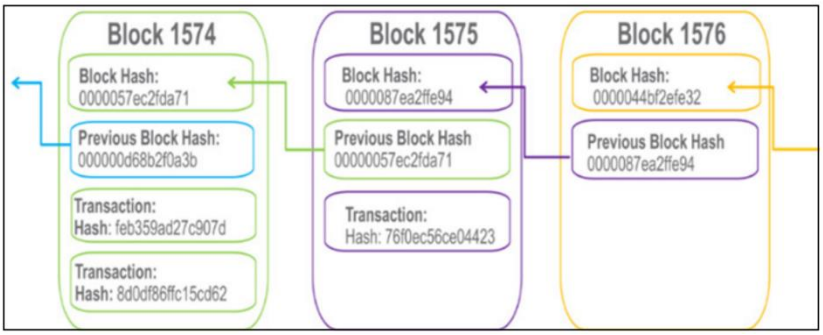
Ακόμα κι αν αυτό μπορεί να μην φαίνεται σημαντικό πρόβλημα, δεν είναι καθόλου βολικό, καθώς απαιτεί να γίνεται επαλήθευση σε κάθε μπλόκ που προσθέτετε στην αλυσίδα.

Είναι αδύνατη η εισαγωγή ενός μπλοκ εκτός εάν η πλειοψηφία του δικτύου συμφωνεί ότι το νέο μπλοκ είναι γνήσιο και έχει επιβεβαιωθεί ως έγκυρο στοιχείο της αλυσίδας[1]. Η τυχόν αλλαγή ήδη επικυρωμένων δεδομένων είναι εξαιρετικά δύσκολη.

Οπότε μπορούμε να συμπεράνουμε ότι η **ασφάλεια του blockchain** είναι άμεσα συνδεδεμένη με τον συνολικό **αριθμό των κόμβων** καθώς και το **μέγεθος της αλυσίδας**[1].

Ας μελετήσουμε λίγο περισσότερο την **αρχιτεκτονική ενός μπλοκ**. Κάθε μπλοκ θα πρέπει να αποτελείται τουλάχιστον από τα ακόλουθα τρία χαρακτηριστικά[2]:

1. Ένα hash (μια ψηφιακή υπογραφή ή ένα μοναδικό αναγνωριστικό στοιχείο)
2. Παρτίδες από πρόσφατα επικυρωμένες συναλλαγές που έχουν χρονογραφηθεί.
3. Το hash του προηγουμένου μπλοκ.



Εικόνα 1.3.1 – *Σύνδεση μεταξύ των μπλόκς. Διαθέσιμο στο βιβλίο [2], σελ. 14, Figure 2-1*

Το hash του προηγούμενου μπλοκ δημιουργεί μια σύνδεση μεταξύ των μπλοκ και απαγορεύει την επεξεργασία ή την εισαγωγή οποιουδήποτε μπλοκ μεταξύ δύο ήδη υπαρχόντων μπλοκ. Έτσι, κάθε διαδοχικό μπλοκ ενισχύει την επαλήθευση του προηγούμενου μπλοκ και επομένως ολόκληρης την αλυσίδας. Αυτή η στρατηγική καθιστά το blockchain αδιαπέραστο σε παραβιάσεις, ενισχύοντας την κρίσιμη ιδιότητα του να παραμένει αναλλοίωτο & αμετάβλητο.

1.3.2 Πρώτο Μπλοκ (Genesis Block)

Το **πρώτο μπλοκ** ενός blockchain ονομάζεται Genesis-Block. Το συγκεκριμένο μπλοκ είναι ιδιαίτερο σε σχέση με αυτά που ακολουθούν διότι μέσα του δεν περιέχονται καθόλου συναλλαγές αλλά περικλείονται κωδικοποιημένες πληροφορίες αφορούν τον τρόπο λειτουργείας του blockchain[10]. Παρακάτω ακολουθεί ο κώδικας ενός genesis block:

```
// Genesis block
const char* pszTimestamp = "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks";
CTransaction txNew;
txNew.vin.resize(1);
txNew.vout.resize(1);
txNew.vin[0].scriptSig = CScript() << 486604799 << CBigNum(4) << vector<unsigned char>((const unsigned char*)
txNew.vout[0].nValue = 50 * COIN;
CBigNum bnPubKey;
bnPubKey.SetHex("0x5F1DF16B2B704C8A578D0BBAF74D385CDE12C11EE50455F3C438EF4C3FBCF649B6DE611FEAE06279A60939E028
txNew.vout[0].scriptPubKey = CScript() << bnPubKey << OP_CHECKSIG;
CBlock block;
block.vtx.push_back(txNew);
block.hashPrevBlock = 0;
block.hashMerkleRoot = block.BuildMerkleTree();
block.nVersion = 1;
block.nTime    = 1231006505;
block.nBits    = 0x1d00ffff;
block.nNonce   = 2083236893;
```

Εικόνα 1.2.2 *- Ο κώδικας του Bitcoin's Genesis Block. Διαθέσιμο στο Διαδίκτυο: https://thenextweb.com/news/bitcoin-easter-egg*

1.4 Χαρακτηριστικά ενός Blockchain

1.4.1 Δυναμική Επεξεργαστική Ισχύ

Ένα από τα πιο αξιοσημείωτα χαρακτηριστικά της τεχνολογίας Blockchain είναι ότι η συνολική επεξεργαστική ισχύς ολόκληρου του δικτύου μεγαλώνει όσο πιο δημοφιλής γίνεται το δίκτυο blockchain. Αυτό συμβαίνει επειδή υπάρχουν πολλοί υπολογιστές που συνεργάζονται, με αποτέλεσμα να αυξάνεται η συνολικά προσφερόμενή ισχύ. Αντίθετα, τα συγκεντρωτικά δίκτυα διαθέτουν στατική επεξεργαστική ισχύ, που είναι ανάλογη με τον διαθέσιμο εξοπλισμό τους[9].

Ένα τέλειο παράδειγμα αυτής της αυξημένης χωρητικότητας είναι ένα έργο που ξεκίνησε από το Πανεπιστήμιο του Στάνφορντ το οποίο δημιούργησε έναν υπερυπολογιστή με αυτόν τον τρόπο οπού προσομοιώνει την αναδίπλωση πρωτεϊνών για ιατρική έρευνα.

1.4.2 Υψηλή Ασφάλεια

• **Η βάση δεδομένων είναι απλωμένη σε πολλούς υπολογιστές**: Κάθε άτομο που χρησιμοποιεί το blockchain έχει πρόσβαση σε ολόκληρη τη βάση δεδομένων, καθώς και στο ιστορικό της[1]. Έτσι, ο έλεγχος δεν συγκεντρώνεται πλέον στα χέρια ενός μόνο οργανισμού. Ως αποτέλεσμα, εάν κάποιος επιθυμεί να πραγματοποιήσει μια συναλλαγή με κάποιον άλλον δεν απαιτείται μεσάζων για τον έλεγχο αυθεντικότητας των δεδομένων, κάτι που μπορεί να οδηγήσει σε φθηνότερες και ταχύτερες συναλλαγές.

• **Peer-to-peer επικοινωνία**: Η μετάδοση δεδομένων χρησιμοποιεί πρωτόκολλα απευθείας peer-to-peer (από χρήστη σε χρήστη)[1]. Αυτό είναι προτιμότερο καθώς η διέλευση των πληροφοριών δεν περιορίζεται σε ένα καθορισμένο κόμβο. Στην περίπτωση που συνέβαινε αυτό οι χάκερ θα επιχειρούσαν να παραβιάσουν τα δεδομένα καθώς αυτά περνούν μέσα από τον κόμβο. Με την τεχνολογία blockchain, οι πληροφορίες διανέμονται σε όλους τους κόμβους του δικτύου. Καθιστώντας την διαδικασία του hacking αρκετά πιο επώδυνη, καθώς τα δεδομένα δεν δρομολογούνται πλέον μέσω ενός κεντρικού διακομιστή.

• **Αμεταβλητότητα των εγγραφών**: Κάθε συναλλαγή συνδέεται με την προηγούμενη και μετά με αυτή που ακολουθεί[1]. Αυτό σημαίνει ότι οι συναλλαγές αποθηκεύονται με τη σειρά με την οποία πραγματοποιήθηκαν. Κάθε συναλλαγή φέρει επίσης χρονική σήμανση. Δεν γίνεται να τροποποιηθεί μια συναλλαγή, καθώς κάτι τέτοιο θα σήμαινε την διαφοροποίηση όλων των υπολοίπων στην αλυσίδα. Ακόμα, θα έπρεπε να πείσει το 51% του δίκτυού να αποδεχτεί την συγκεκριμένη αλλαγή.

1.4.3 Διαφάνεια και Ανωνυμία

Οποιοσδήποτε έχει πρόσβαση σε κάθε μπλοκ δεδομένων στην αλυσίδα, από τη δημιουργία του έως την επιβεβαίωσή του μέσω των κόμβων του δικτύου. Κάθε κόμβος αναγνωρίζεται από μια μοναδική διεύθυνση[1]. Εναπόκειται στους υποψηφίους να αποφασίσουν αν θέλουν να διατηρήσουν ή να μοιραστούν την πραγματική τους ταυτότητα. Το κρίσιμο σημείο εδώ είναι ότι αυτό μπορεί να επιτευχθεί ανώνυμα.

1.4.4 Υπολογιστική Λογική

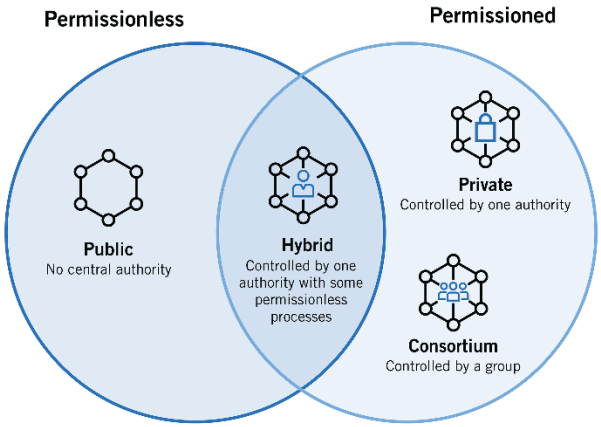
Οι συναλλαγές μπορούν να συνδεθούν με υπολογιστική λογική[1]. Αυτό σημαίνει ότι μπορούν να χρησιμοποιηθούν στον προγραμματισμό. Για παράδειγμα, μπορείτε να αναπτύξετε μια εφαρμογή που θα πραγματοποιεί αυτόματα μια συναλλαγή σε μια προκαθορισμένη χρονική στιγμή ή όταν συμβαίνει ένα συγκεκριμένο συμβάν. Χρησιμοποιώντας αυτό το χαρακτηριστικό μπορεί κανείς να δημιουργήσει αυτόνομες εφαρμογές που επικοινωνούν με το blockchain χωρίς να απαιτείται ανθρώπινη αλληλεπίδραση.

1.5 Κατηγορίες Blockchain

Τα δίκτυα blockchain μπορούν να κατηγοριοποιηθούν με βάση το μοντέλο άδειας (permission model) που χρησιμοποιούν, το οποίο διέπει ποιος μπορεί να διατηρεί το δίκτυο (π.χ. έκδοσή μπλόκς). Στην περίπτωση οπού ο καθένας έχει την ικανότητα να εκδώσει ένα μπλοκ, το δίκτυο ανήκει στην κατηγορία των **ελευθέρων δικτυών** που δεν απαιτείται ειδική άδεια (**permissionless**). Στην περίπτωση που μονάχα ειδικά επιλεγμένοι κόμβοι μπορούν να εκδώσουν νέα μπλόκς, το δίκτυο πέφτει στην κατηγορία των **εξουσιοδοτημένων δικτυών** (**permissioned**) [25].

Υπάρχουν άλλες δυο κατηγορίες στις οποίες μπορούν να χωριστούν τα blockchain. Αυτές είναι, τα **δημοσιά** (public) και **ιδιωτικά** (private) **blockchains**. Με λίγα λογία, η ουσιαστική **διαφορά μεταξύ των δυο** έχει να κάνει με την **προσβασιμότητα** (accessibility). Στα *δημοσιά blockchain*, οποιοσδήποτε μπορεί να συνδεθεί χωρίς να υπάρχει κάποιος έλεγχος. Από την άλλη μεριά στα *ιδιωτικά blockchain* για μπορέσει κάποιος να συνδεθεί θα πρέπει πρώτα να δεχτεί κάποιου είδους πρόσκληση ή να πληροί κάποιες προϋποθέσεις.

Τέλος, πρέπει να αναφερθεί πως ένα δίκτυο blockchain μπορεί να έχει **ένα χαρακτηριστικό από κάθε κατηγορία (*διατήρηση δικτύου* και *προσβασιμότητα*)** που μόλις αναφέρθηκαν. Δηλαδή, ένα blockchain μπορεί να είναι ταυτόχρονα *ελεύθερο* και *δημόσιο*, ενώ ένα άλλο blockchain να είναι *ελεύθερο* και *ιδιωτικό*. Η πρώτη κατηγορία επηρεάζει τις συνθήκες για την έκδοσή των μπλόκς, ενώ η δεύτερη την δυνατότητα πρόσβασης/σύνδεσης στο δίκτυο.



Εικόνα 1.5.1 – *Οι 4 κατηγορίες των Blockchain. Διαθέσιμο στο Διαδίκτυο: https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between*

1.5.1 Ελευθέρα Δίκτυα (Permissionless)

Τα ελευθέρα δίκτυα blockchain χωρίς άδεια είναι αποκεντρωμένα συστήματα καθολικού που επιτρέπουν σε οποιονδήποτε να δημοσιεύει μπλόκς χωρίς την απαίτηση άδειας. Τα συστήματα blockchain χωρίς άδεια είναι συχνά λογισμικά ανοιχτού κώδικα που οποιοσδήποτε μπορεί να κατεβάσει δωρεάν. Επειδή όλοι έχουν τη δυνατότητα να δημοσιεύουν μπλόκς, αυτό σημαίνει ότι οποιοσδήποτε μπορεί να έχει πρόσβαση στο blockchain καθώς και να εκδώσει συναλλαγές σε αυτο (συμπεριλαμβάνοντας και τις συναλλαγές σε ήδη δημοσιευμένα μπλόκς)[25].

Μέσα σε ένα δίκτυο blockchain χωρίς άδεια, κάθε χρήστης μπορεί να διαβάζει και να γράφει στο καθολικό. Επειδή τα δίκτυα blockchain χωρίς άδεια είναι διαθέσιμα σε όλους, τα κακόβουλα άτομα μπορεί να προσπαθήσουν να εκμεταλλευτούν το σύστημα δημοσιεύοντας μπλόκς με μη εξουσιοδοτημένο τρόπο. Για να αποφευχθεί αυτό, τα δίκτυα blockchain χωρίς άδεια χρησιμοποιούν συχνά μια πολυμερή συμφωνία ή προσέγγιση «συναίνεσης» (*βλ. Ενότητα 1.6*) που υποχρεώνει τους χρήστες να επενδύσουν (PoW) ή να διατηρήσουν πόρους (PoS) κατά τη δημοσίευση των μπλοκ. Αυτό καθιστά δύσκολο για τους κακόβουλους χρήστες να υπονομεύσουν το σύστημα. Η απόδειξη εργασίας (PoW, *βλ. Ενότητα 1.6.2*) και η απόδειξη συμμετοχής (PoS*, βλ. Ενότητα 1.6.3*) είναι δύο παραδείγματα τέτοιων συναινετικών προσεγγίσεων. Οι μέθοδοι συναίνεσης των δικτύων blockchain χωρίς άδεια συνήθως ενθαρρύνουν τη μη κακόβουλη δραστηριότητα, ανταμείβοντας τους εκδότες μπλοκ που συμμορφώνονται με το πρωτόκολλο με εγγενή νομίσματα[25].

1.5.2 Εξουσιοδοτημένα Δίκτυα (Permissioned)

Τα εξουσιοδοτημένα δίκτυα blockchain είναι εκείνα στα οποία οι χρήστες που δημοσιεύουν τα μπλόκς πρέπει να έχουν εγκρίνομε από κάποιο εξουσιοδοτημένο μέλος (είτε είναι κεντρικό είτε αποκεντρωμένο). Είναι εφικτός ο έλεγχος προσβασιμότητας της ανάγνωσης του συναλλαγών και του ποιος μπορεί να εκδίδει συναλλαγές, καθώς το blockchain διατηρείται μόνο από εξουσιοδοτημένους χρήστες[25].

Τα εξουσιοδοτημένα δίκτυα blockchain μπορούν να επιτρέπουν σε οποιονδήποτε να δει το blockchain ή να περιορίζονται μόνο σε εξουσιοδοτημένα άτομα. Το λογισμικό τους μπορεί να είναι ανοιχτού ή κλειστού κώδικα[25].

Τα εξουσιοδοτημένα δίκτυα blockchain μπορεί να έχουν την ίδια ιχνηλασιμότητα των ψηφιακών τους στοιχείων με τα ελευθέρα δίκτυα blockchain, καθώς και τον ίδιο κατανεμημένο και ισχυρό μηχανισμό αποθήκευσης δεδομένων. Χρησιμοποιούν και αυτά μοντέλα συναίνεσης για τη δημοσίευση μπλοκ, ωστόσο αυτές οι προσεγγίσεις δεν χρειάζονται πάντα τη δαπάνη ή τη συντήρηση πόρων (όπως κάνουν τα τρέχοντα ελευθέρα δίκτυα blockchain). Αυτό συμβαίνει διότι η εξακρίβωση της ταυτότητας κάποιου είναι απαραίτητη για την ένταξη στο εξουσιοδοτημένο δίκτυο blockchain[25].

Στα άτομα που διατηρούν το blockchain υπάρχει ένας βαθμός εμπιστοσύνης μεταξύ τους, διότι τους δόθηκε η άδεια να δημοσιεύουν μπλοκ και επίσης αυτή η άδειά μπορεί ευκολά να αφαιρεθεί εάν συμπεριφερθούν κακόβουλα. Στα **εξουσιοδοτημένα δίκτυα blockchain**, οι **μηχανισμοί συναίνεσης** είναι γενικά **ταχύτεροι και λιγότερο δαπανηροί** όσον αφορά την υπολογιστική ισχύ[25].

Οι οργανισμοί που επιθυμούν **αυστηρότερο έλεγχο και προστασία** ενδέχεται να χρησιμοποιούν **εξουσιοδοτημένα δίκτυα blockchain**. Οι χρήστες του blockchain, από την άλλη πλευρά, θα πρέπει να έχουν πίστη σε ένα ενιαίο σώμα που θα κυβερνά ποιος μπορεί να δημοσιεύει μπλόκς. Αυτού του είδους τα δίκτυα μπορούν επίσης να χρησιμοποιηθούν από εταιρείες που θέλουν να συνεργαστούν με άλλες αλλά ταυτόχρονα δεν εμπιστεύονται πλήρως η μία την άλλη. Σε αυτήν την περίπτωση, μπορούν να δημιουργήσουν ένα εξουσιοδοτημένο δίκτυο blockchain και να επιτρέψουν στους επιχειρηματικούς εταίρους να μοιράζονται ένα κατανεμημένο καθολικό για την καταγραφή των συναλλαγών τους[25].

Με βάση το πόσο εμπιστεύονται ο ένας τον άλλον, αυτοί οι οργανισμοί μπορούν να επιλέξουν το μοντέλο συναίνεσης που θα χρησιμοποιήσουν. Τα εξουσιοδοτημένα δίκτυα blockchain επιτρέπουν τη διαφάνεια και τη γνώση, κάτι που μπορεί να βοηθήσει τις επιχειρήσεις να κάνουν καλύτερες επιλογές και να εντοπίσουν ευκολότερα κακούς παράγοντες[25].

Ορισμένα εξουσιοδοτημένα δίκτυα **blockchain επιτρέπουν την επιλεκτική αποκάλυψη πληροφοριών** ανάλογα με την ταυτότητα ή τα διαπιστευτήρια ενός χρήστη. Αυτή η δυνατότητα επιτρέπει ένα ορισμένο **επίπεδο μυστικότητας συναλλαγών**. Για παράδειγμα, το blockchain μπορεί να καταγράφει μια συναλλαγή μεταξύ δύο μελών του δικτύου, αλλά οι πραγματικές λεπτομέρειες της συναλλαγής είναι ορατές μόνο στα εμπλεκόμενα άτομα[25].

Για τη μετάδοση και τη λήψη συναλλαγών σε ορισμένα εξουσιοδοτημένα δίκτυα blockchain, όλοι οι χρήστες πρέπει να είναι εγκεκριμένοι (δεν είναι ανώνυμοι ούτε και ψευδό-ανώνυμοι). Τα μέλη σε τέτοια συστήματα συνεργάζονται για να δημιουργήσουν μια κοινή επιχειρηματική διαδικασία με ενσωματωμένα αντικίνητρα για τη διάπραξη απάτης ή κακού παράγοντα (καθώς μπορούν να εντοπιστούν) [25].

1.5.3 Δημοσιά Δίκτυα (Public)

Όπως σύντομα αναφέραμε και στην αρχή του κεφαλαίου, τα δημοσιά δίκτυα blockchain είναι αυτά που επιτρέπουν σε οποιοδήποτε χρήστη να εισέλθει στο δίκτυο δίχως να υπάρχουν περιορισμοί ή έλεγχοι. Για παράδειγμα, μπορείτε να φανταστείτε ότι τα δημοσιά blockchain είναι σαν το Internet. Οποιοσδήποτε έχει τα μέσα για να συνδεθεί (δηλ. μια υπολογιστή μονάδα και σύνδεση σε ένα παροχή που προσφέρει πρόσβαση στη Internet) μπορεί να το χρησιμοποιήσει. Στο κεφάλαιο 1.9 θα δούμε τέσσερα δημοφιλή blockchain τα οποία είναι όλα δημοσιά.

Συνεχίζοντας, θα δούμε τα *πλεονεκτήματα* καθώς και *μειονεκτήματα* της χρήσης ενός δημοσίου blockchain.

ΠΛΕΟΝΕΚΤΗΜΑΤΑ

- ✓ **Δεν υπάρχει ανάγκη για εμπιστοσύνη (Trustless)** : Ο στόχος είναι να εξαλειφθούν οι μεσάζοντες οποιαδήποτε μορφής, καθώς και η απαίτηση για εμπιστοσύνη που πρέπει να τους έχουν οι επιχειρηματικοί παράγοντες προκειμένου να εξελιχθούν ομαλά οι δραστηριότητές τους. Σε αυτόν τον τομέα, το δημόσιο blockchain είναι πολλά υποσχόμενο.

- ✓ **Ασφαλής**: Όσο μεγαλύτερο είναι το ποσοστό της αποκέντρωσης καθώς και της ενεργού συμμετοχής σε ένα δίκτυο blockchain, τόσο πιο ασφαλής γίνεται. Για να το θέσουμε διαφορετικά, όσο περισσότεροι κόμβοι υπάρχουν σε ένα δίκτυο, τόσο πιο δύσκολο θα είναι για τους χάκερ να του επιτεθούν. Σε ένα δημόσιο δίκτυο, οποιοσδήποτε μπορεί να βοηθήσει στη διατήρηση του, αναλάμψαντας τον ρολού ενός κόμβου. Είναι σχεδόν αδύνατον για τους χάκερ να κατακτήσουν το δίκτυο συναίνεσης ακόμα και εάν προσπαθήσουν να συνεργαστούν μεταξύ τους.

- ✓ **Ανοιχτό και διαφανές**: Όλα τα δεδομένα συναλλαγών είναι προσβάσιμα για δημόσια επιθεώρηση και επαλήθευση. Ένα θεμελιώδες χαρακτηριστικό ενός Public Blockchain είναι η διαφάνειά του, η οποία προσφέρει μια ευρεία ποικιλία εφαρμογών, από την ψηφοφορία έως τις οικονομικές συναλλαγές. Επιπλέον, οποιοσδήποτε μπορεί να επαληθεύσει την ακρίβεια των συναλλαγών και των δεδομένων.

ΜΕΙΟΝΕΚΤΗΜΑΤΑ

- ☹ **Αργό**: Δεν πρέπει να προκαλεί έκπληξη το γεγονός ότι τα δημόσια Blockchain είναι εξαιρετικά αργά. Το Bitcoin, για παράδειγμα, μπορεί να χειριστεί 7 συναλλαγές ανά δευτερόλεπτο, ενώ το Ethereum μπορεί να επεξεργαστεί 15 συναλλαγές ανά δευτερόλεπτο.

Ένας κεντρικός επεξεργαστής πληρωμών, όπως η Visa, από την άλλη πλευρά, μπορεί να χειριστεί γύρω στις 20.000 συναλλαγές ανά δευτερόλεπτο. Ένα δημόσιο blockchain είναι αρκετά νωχελικό επειδή χρειάζεται χρόνος για να συμφωνήσει ολόκληρο το δίκτυο του σχετικά με την κατάσταση των συναλλαγών, η οποία επιτυγχάνεται μέσω διαδικασιών συναίνεσης

όπως για παράδειγμα το Proof of Work που χρησιμοποιείται στο Bitcoin. Το ποσό των συναλλαγών που μπορεί να εισέλθει σε ένα μπλοκ και ο χρόνος που απαιτείται για την εξόρυξη ή δημιουργία του μπλοκ και ένταξη του στην αλυσίδα είναι εξίσου σημαντικοί περιορισμένοι.

- ☹ **Επεκτασιμότητα (Scalability)**: Οι δημόσιες αλυσίδες μπλοκ δεν είναι προς το παρόν σε θέση να ανταγωνιστούν παλαιότερα συστήματα που μπορούν να χειριστούν μεγάλους όγκους συναλλαγών. Πράγματι, καθώς ο αριθμός των συμμετεχόντων αυξάνεται, ο αριθμός των συναλλαγών που πρέπει να διεκπεραιωθούν στο δίκτυο αυξάνεται και το Public Blockchain γίνεται πιο αργό.

- ☹ **Κατανάλωση ενέργειας**: Η συναινετική μέθοδος του Bitcoin, Proof of Work, καταναλώνει σημαντική ποσότητα ηλεκτρικής ενέργειας για να λειτουργήσει, εγείροντας περιβαλλοντικές ανησυχίες. Πράγματι, το blockchain του Bitcoin χρησιμοποιεί την ιδιά ισχύ με την Ιρλανδία ή το 5% της παγκόσμιας ενέργειας που καταναλώνεται για την παραγωγή αλουμινίου. Υπάρχουν και άλλες λιγότερο ενεργοβόρες τεχνικές συναίνεσης, όπως η Απόδειξη Πονταρίσματος (PoS) και η Απόδειξη Εξουσιοδότησης (PoA) που θα αναφερθούν παραπάνω, έχουν προκύψει για να ξεπεραστεί αυτό το ζήτημα.

1.5.4 Ιδιωτικά Δίκτυα (Private)

Τα ιδιωτικά δίκτυα blockchain είναι το αντίθετο των δημοσίων, αφού για να μπορέσει κάποιος να συνδεθεί σε ένα τέτοιο δίκτυο θα πρέπει να πληροί κάποιες προϋποθέσεις ή να λάβει κάποιου είδους πρόσκληση από ένα ήδη υπάρχον μέλος που του έχει δοθεί αυτό το δικαίωμα.

Τα ιδιωτικά δίκτυα blockchain είναι περισσότερο αρεστά σε ιδιωτικές επιχειρήσεις ή οργανισμούς οι οποίοι επιθυμούν να έχουν τα πλεονεκτήματα της χρήσης ενός blockchain ενώ ταυτόχρονα δεν αποκαλύπτουν σε άλλες εταιρίες ευαίσθητα δεδομένα.

Ας δούμε τα πλεονεκτήματα και τα μειονεκτήματα των ιδιωτικών blockchain:

ΠΛΕΟΝΕΚΤΗΜΑΤΑ

- ✓ **Ταχύτητα**: Σε σύγκριση με τις δημόσιες αλυσίδες μπλοκ, οι ιδιωτικές μπορούν να εκτελούν πολλές περισσότερες συναλλαγές ανά δευτερόλεπτο. Πράγματι, δεδομένου ότι η πρόσβαση περιορίζεται σε ένα σταθερό αριθμό χρηστών, είναι εφικτό να επιτευχθεί συναίνεση στο δίκτυο πολύ γρηγορότερα λόγω μικρότερης ανάγκης για συμφόρηση. Σε αντίθεση με ένα αποκεντρωμένο σύστημα, όπου η οικοδόμηση συναίνεσης απαιτεί αρκετό χρόνο, η λήψη αποφάσεων σε ένα ιδιωτικό δίκτυο είναι πιο συγκεντρωτική (επίσης, λόγω του γεγονότος ότι ο αριθμός των κόμβων είναι πολύ λιγότερος από ό,τι σε ένα δημόσιο blockchain) και ως εκ τούτου πολύ πιο γρήγορη.

ΜΕΙΟΝΕΚΤΗΜΑΤΑ

- ☹ **Ανάγκη εμπιστοσύνης**: Σε αντίθεση με το δημόσιο Blockchain, στο οποίο δεν χρειάζεται να υπάρχει εμπιστοσύνη (καθώς το δίκτυο είναι προσβάσιμο στο κοινό), η ακεραιότητα του ιδιωτικού δικτύου του Blockchain εξαρτάται αρκετά από την αξιοπιστία των εξουσιοδοτημένων κατόχων των κόμβων, οι οποίοι είναι υπεύθυνοι για την επαλήθευση και την επικύρωση των συναλλαγών. Κατά συνέπεια, είναι αδύνατο να επαληθευτεί ανεξάρτητα η ακρίβεια των εγγράφων (logs. Οι παράγοντες εκτός του blockchain πρέπει να το εμπιστεύονται, με αποτέλεσμα την έλλειψη ελέγχου στην επαλήθευση των δεδομένων που μετακινούνται εκεί.

- ☹ **Ασφάλεια**: Με λιγότερους κόμβους, ένα δίκτυο γίνεται πιο συγκεντρωτικό με αποτέλεσμα ένας χάκερ να μπορεί εύκολα να αποκτήσει τον έλεγχο του δικτύου και να τροποποιήσει τα δεδομένα σε αυτό.

1.6 Αλγόριθμοι κοινής συναίνεσης

1.6.1 Εισαγωγή

Όπως ήδη μάθαμε ως τώρα το blockchain είναι ένα κατανεμημένο δίκτυο το οποίο προσφέρει την δυνατότητα εμπιστοσύνης μεταξύ των συμμετεχόντων δίχως την ανάγκη τρίτων μελών για την επίτευξης και τον εξασφαλισμό της ασφάλειας.

Για να το καταφέρει αυτό το blockchain χρησιμοποιήσει αλγορίθμους συναίνεσης. Η διαδικασία της συναίνεσης βασίζεται στην ιδέα της διενέργειας συχνών ασφαλών αλλαγών στο κατανεμημένο καθολικό. Ο συγχρονισμός των κόμβων είναι μια κρίσιμη προσέγγιση που διασφαλίζει την ύπαρξη και την εκτέλεση κοινών καταστάσεων σύμφωνα με προκαθορισμένους κανόνες οι οποίοι χρησιμοποιούνται για την αλλαγή της κατάστασης του blockchain.

Λόγω του γεγονότος ότι η εκάστοτε κατάσταση μοιράζεται σε πολλά αντίγραφα μέσα στο δίκτυο, η εκτέλεση της εκάστοτε κατάστασης θα παράγει στο τέλος ισοδύναμα αποτελέσματα. Επομένως, τα αντίγραφα πρέπει να επικοινωνούν και να επιτυγχάνουν συναίνεση σχετικά με ενδεχόμενες ενημερώσεις κατάστασης χρησιμοποιώντας μια τεχνική συναίνεσης.

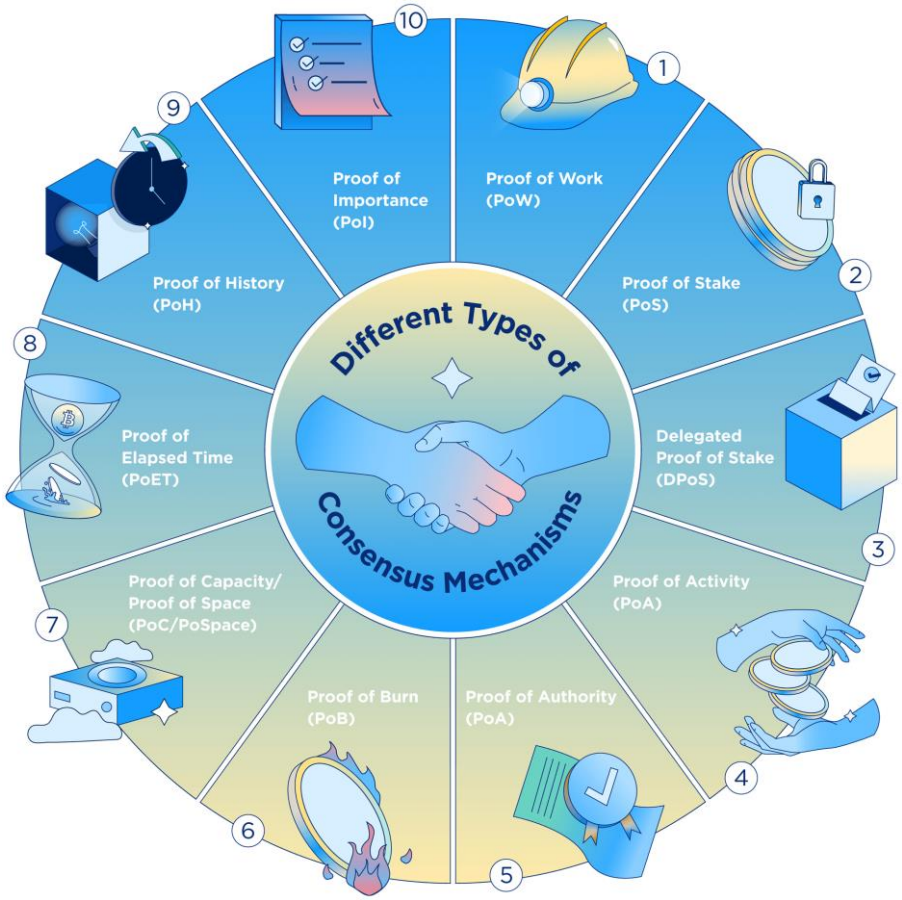
Ωστόσο, η ενσωμάτωση της συναίνεσης σε ένα κατανεμημένο σύστημα είναι αρκετά προκλητική διαδικασία λόγω της ανάγκης για ένα μοντέλο συναίνεσης που να είναι σε θέση να διατηρήσει ανοχή στις αντιξοότητες, ανθεκτικότητα στην αστοχία, τμηματοποίηση του δικτύου, διαχείριση των καθυστερήσεων και πολλές άλλες κρίσιμες ιδιότητες.

Επιπλέον, ένας αλγόριθμος συναίνεσης ενσωματώνει μέτρα ασφαλείας, όπως η επίβλεψη κακόβολων κόμβων μέσω της εφαρμογής νόμων, όπως ο συγχρονισμός ή η μετάδοση μηνυμάτων. Η συνάφεια της συναίνεσης γενικότερα στα κατανεμημένα συστήματα (DLTs), συμπεριλαμβανομένης της τεχνολογίας blockchain, είναι η **διατήρηση τριών θεμελιωδών χαρακτηριστικών** που διασφαλίζουν την αποτελεσματικότητα του υποκείμενου δικτύου.

Η συναίνεση δεν είναι μονάχα η διατήρηση μιας συνεπούς παγκόσμιας κατάστασης σε συμφωνία σε ένα κατανεμημένο καθολικό, αλλά επίσης διασφαλίζει την **ασφάλεια**, τη

λειτουργικότητα και την **ανοχή σφαλμάτων** του δικτύου. Κατά συνέπεια, οι αλγόριθμοι συναίνεσης μπορούν να αξιολογηθούν χρησιμοποιώντας αυτά τα χαρακτηριστικά.

Το μοντέλο συναίνεσης μπορεί να διατηρήσει την ασφάλεια εάν εγγυάται ότι όλοι οι κόμβοι θα παρέχουν πανομοιότυπη, συνεχή και νόμιμη έξοδο. Το Liveness είναι η ικανότητα ενός μοντέλου συναίνεσης να καθοδηγεί τη συμβολή των κόμβων χωρίς σφάλματα κατά τη δημιουργία αξίας. Προκειμένου να διατηρηθεί ένας συγκεκριμένος βαθμός ανοχής σε σφάλματα, απαιτείται ο συναινετικός αλγόριθμος να είναι σε θέση να ανακάμψει από τις πιθανές αστοχίες των συνεισφερόντων κόμβων.



Εικόνα 1.6.1 *– Διαφορά δημοφιλή είδη Αλγορίθμων Συναίνεσης. Διαθέσιμο στο Διαδίκτυο: https://crypto.com/university/consensus-mechanisms-in-blockchain*

1.6.2 PAXOS

Ως **ο πρώτος αλγόριθμος συναίνεσης**, ο PAXOS διευκολύνει την επιλογή μιας μεμονωμένης τιμής όταν υπάρχουν προβληματικές συνθήκες στο δίκτυο. Ο PAXOS ταξινομεί τους κόμβους σε αυτούς που προτείνουν (**proposers**), στους αποδέκτες (**acceptors**) και στους μαθητές (**learners**). Οι proposers παρέχουν ένα μήνυμα που υποδεικνύει έναν αριθμό πρότασης και το προωθούν στον acceptor. Ο αριθμός της πρότασης θεωρείται ως χρονοδιάγραμμα σε όλη τη διαδικασία, στην οποία η πρόταση με υψηλότερο αριθμό είναι και η πιο πρόσφατη. Ο acceptor συγκρίνει τον αριθμό πρότασης με την τρέχουσα γνωστή τιμή, και αποδέχεται την πρόταση μόνο εάν είναι η πιο πρόσφατη. Μετά, ο αποδέκτης προωθεί ένα μήνυμα απάντησης που υποδεικνύει εάν η πρόταση έγινε αποδεκτή ή απορρίφθηκε, καθώς και τον αριθμό πρότασης και όλες τις αποδεκτές τιμές. Οι proposers απαιτείται να διερευνήσουν εάν η πλειοψηφία των acceptors απέρριψε την πρόταση ή όχι. Σε περίπτωση απόρριψης ο proposer ενημερώνει τον αριθμό πρότασης με την πιο πρόσφατη τιμή. Διαφορετικά, ο acceptor μεταδίδει την αποδεκτή τιμή στο όλους τους learners στο δίκτυο. Για να επιτευχθεί συναίνεση στο PAXOS, ο proposer θα πρέπει να λάβει τουλάχιστον N/2 − 1 αποδοχές (N είναι ο αριθμός των προτάσεων) από τους acceptors [11].

1.6.3 Proof of Work (PoW)

Η διαδικασία απόδειξης εργασίας (Proof Of Work) ξεκινά με τον υπολογισμό της τιμής κατακερματισμού(hash) της κεφαλίδας μπλοκ. Στην κεφαλίδα του μπλοκ περιλαμβάνεται ένα nonce (only-once), το οποίο αλλάζεται τακτικά από τους εξορύκτες για την επίτευξη εναλλακτικών τιμών κατακερματισμού. Ως αποτέλεσμα, ο μηχανισμός συναίνεσης απαιτεί η κερδισμένη τιμή να παραμείνει εντός ενός συγκεκριμένου εύρους.

Το PoW επιβάλλει μια δύσκολη μαθηματική πρόκληση που πρέπει να επιλυθεί από τους συμμετέχοντες κόμβους προκειμένου να διατηρηθεί η συναίνεση του δικτύου σχετικά με τη διάδοση νέων μπλοκ.

Οι ανθρακωρύχοι που θα λύσουν το αίνιγμα θα λάβουν την άδεια να προσθέσουν ένα νέο μπλοκ στη αλυσίδα. Το πρόβλημα διατηρεί την προσαρμοσμένη πολυπλοκότητά του και λύνεται με τον υπολογισμό της τιμής του nonce. Αυτή η τιμή χρησιμοποιείται με τις πληροφορίες κεφαλίδας στο μπλοκ για την τροφοδοσία του αλγόριθμου κατακερματισμού SHA-256 (*βλ. κεφ. 2.2.3.2*). Μετά από αυτό, η συνάρτηση κατακερματισμού(hash function) παίρνει όλες τις εισόδους για να παράγει το αποτέλεσμα κατακερματισμού.

Εάν η έξοδος της συνάρτησης κατακερματισμού περιέχει ένα ορισμένο αριθμό μηδενικών ψηφίων γίνεται αποδεκτή και ο εξορύκτης επιτρέπεται να προσθέσει ένα μπλοκ στην αλυσίδα. Ως αποτέλεσμα, κάθε φορά που ένας εξορύκτης αποκτά την αντικειμενική τιμή, θα μεταδίδει το μπλοκ σε ολόκληρο το δίκτυο και ο αλγόριθμος συναίνεσης θα απαιτήσει από κάθε κόμβο να πιστοποιήσει τη νομιμότητα της τιμής κατακερματισμού και να προσθέσει το σχετικό μπλοκ στο αποθηκευμένο αντίγραφο της αλυσίδα του. Ο κόμβος που κατάφερε να λύσει το αίνιγμα ανταμείβεται με το τοπικό νόμισμα του blockchain. Τα πιο δημοφιλή blockchains που χρησιμοποιούν PoW είναι το Bitcoin και το Ethereum [12].

1.6.4 Proof of Stake (PoS)

Το Proof of Stake (PoS), το οποίο προτάθηκε για πρώτη φορά για το Peercoin, έχει χρησιμοποιηθεί ως υποκατάστατο του proof of work (PoW) για τη μείωση της χρήσης ηλεκτρικής ισχύος του κάθε κόμβου. Οι προτεινόμενες λύσεις περιλαμβάνουν το ποσό πονταρίσματος, καθώς η επιλογή του προτεινόμενου αποκλεισμού με βάση το υπόλοιπο του λογαριασμού φαίνεται άδικη. Αν και το PoS είναι πιο ενεργειακά αποδοτικό από το PoW, είναι ευάλωτο σε επιθέσεις. Ως αποτέλεσμα, ορισμένα συστήματα blockchain ξεκινούν με PoW και στη συνέχεια μεταβαίνουν σε PoS. Ένα από αυτά είναι το Ethereum που αναμένεται στα τέλη του 2022 ή αρχές του 2023 να αλλάξει σε PoS [13], [14].

1.6.5 Proof of Stake Time (PoST)

Το Proof of Stake Time (PoST) είναι ένας μη γραμμικός αλγόριθμος συναινετικής σχετιζόμενος με τον χρόνο και έχει προταθεί ως αντικαταστάτης του Proof of Stake (PoS). Το PoST στοχεύει στη βελτίωση της ασφάλειας και της διανομής του δικτύου συμπεριλαμβάνοντας μια λειτουργία περιοδικής χρονικής αποδοχής που αντιστοιχεί με τα διατηρημένα νομίσματα. Οι συνεισφορές εθελοντών υπολογίζονται χρησιμοποιώντας ένα επιτόκιο που είναι αντιστρόφως ανάλογο με την ισχύ του δικτύου.

Το PoST δημιουργεί ένα ποσοτικοποιημένο χαρακτηριστικό χρόνου αδράνειας για να αντικατοπτρίζει ένα ποσοστό ηλικίας που δεν ενισχύει πλέον στη συναινετική κατανομή. Αυτή η παράμετρος επιδρά στους κερδισμένους τόκους και αφαιρεί την πιθανότητα να ικανοποιηθεί η απόδειξη. Ως αποτέλεσμα, για να αυξηθεί το επιτόκιο, ο συμμετέχων κόμβος πρέπει να ποντάρει συνεχώς προκειμένου να περάσει όλους τους συσχετιζόμενους κόμβους μέσω του παραθύρου χρόνου συμμετοχής [15], [16].

1.6.6 Proof of Burn (PoB)

Το Proof of Burn έχει προταθεί ως ενεργειακά αποδοτική, μακροπρόθεσμη εναλλακτική λύση σε σχέση με το Proof of Work, στην οποία οι εξορύκτες(miners) στέλνουν χρήματα σε μια μη αναστρέψιμη διεύθυνση με αποτέλεσμα να τα «καίνε». Για να αποτραπεί η ανάκτηση νομισμάτων, η μη αναστρέψιμη διεύθυνση είναι γνωστή ως διεύθυνση καταναλωτή(eater address) και περιλαμβάνει ένα δημόσιο κλειδί που δεν σχετίζεται με κανένα ιδιωτικό κλειδί.

Μόλις ένα νόμισμα μεταδοθεί στη διεύθυνση του καταναλωτή, αφαιρείται από το δίκτυο για πάντα. Οι εξορύκτες στο PoB δεν επενδύουν σε πραγματικά μετρητά, καθώς τα κρυπτονομίσματα καίγονται για να προσδιορίσουν την επένδυση της αλυσίδας μπλοκ. Η καύση του bitcoin δημιουργεί εικονική δύναμη εξόρυξης. Ως αποτέλεσμα, όσο περισσότερα νομίσματα καίει ένας χρήστης για να υποστηρίξει το σύστημα, τόσο περισσότερη δύναμη εξόρυξης αποκτά το σύστημα.

Επιπλέον, όπως αναφέρθηκε προηγουμένως, ο εξορύκτης ο οποίος καίει περισσότερα νομίσματα είναι ο πιο πιθανός να οριστεί ως ο επικυρωτής του μπλοκ. Η ιδέα του PoB είναι συγκρίσιμη με εκείνη του PoW καθώς η απόκτηση του τίτλου του block miner στο PoB ισοδυναμεί με την απόκτηση πόρων υπολογιστή σε PoW. Όλες οι συναλλαγές που υποδεικνύουν τη μεταφορά νομισμάτων σε διευθύνσεις καταναλωτή καταγράφονται και ο κατακερματισμός εγγραφής για κάθε συναλλαγή στο δίκτυο υπολογίζεται χρησιμοποιώντας τον αλγόριθμο SHA-256. Ο εξορύκτης με τη χαμηλότερη αξιολόγηση κατακερματισμού εγγραφής κερδίζει τελικά το δικαίωμα εξόρυξης [17], [18].

1.6.7 Proof of History (PoH)

Για την αντιμετώπιση των προκλήσεων που σχετίζονται με την έντονη ανάγκη για επεξεργαστική ισχύ, έχει προταθεί η απόδειξη της ιστορίας(Proof of History). Αυτή η μέθοδος χρησιμοποιεί τον αλγόριθμο κατακερματισμού SHA-256 σε ένα σύστημα με γύρους που το αποτέλεσμα κάθε γύρου αποτελεί την κατάλληλη είσοδο για τον επόμενο γύρο. Η επιβεβαίωση και η ενσωμάτωση κάθε συναλλαγής με τον τρέχοντα κατακερματισμό είναι ευθύνη των ηγετών.

Σε σύγκριση με το συμβατικό PoW, το PoH αναγνωρίζεται ότι είναι ενεργειακά αποδοτικό αφού δεν εκτελεί δαπανηρές δραστηριότητες εξόρυξης. Ωστόσο, ευνοεί τους πλούσιους ηγέτες, με αποτέλεσμα μια πιο συγκεντρωτική και προβλέψιμη διαδικασία που χρειάζεται μεγάλη χωρητικότητα λόγω της επανειλημμένης εκτέλεσης της συνάρτησης κατακερματισμού. Ένα πρόσφατο blockchain, το Solana, κάνει χρήση αυτού το μηχανισμού συναίνεσης [19], [20], [21].

1.6.8 Proof of Proof (PoP)

Τα αποδεικτικά στοιχεία συναίνεσης (Proof of Proof) επιτρέπουν σε ένα νεοσύστατο blockchain, γνωστό ως blockchain κληρονομιάς ασφάλειας (Security Inheriting, SI), να κληρονομήσει μέτρα ασφαλείας από blockchain Παροχής Ασφαλείας (Security Providing, SP), επιτρέποντας επεκτασιμότητα. Ο μηχανισμός κληρονομικότητας PoP είναι αποκεντρωμένος και

δεν χρειάζεται άδεια από το blockchain SP ή εξουσιοδότηση από κεντρικό δίκτυο ή ομοσπονδιακά ιδρύματα.

Εκτός από την αποτροπή της επαφής χρηστών SI που δεν κάνουν εξόρυξη στα δίκτυα SP, δεδομένου ότι είναι υποχρεωμένοι να συμβάλλουν στη διατήρηση των εγγενών διακριτικών του blockchain, η κληρονομικότητα δεν επιβάλλει κανένα μη τετριμμένο ή τεχνικό όριο στις αλυσίδες μπλοκ SI που εφαρμόζουν αυτό το πρωτόκολλο. Η κατάσταση που έχει μεταδοθεί στο δίκτυο blockchain SP χρησιμοποιείται για την παροχή κινήτρων στο blockchain SI [22], [23], [24].

1.6.9 Proof of Authority (PoA)

Για εξουσιοδοτημένα (permissioned) δίκτυα blockchain, η Απόδειξη Εξουσιοδότησης (**P**roof **o**f **A**uthority, PoA) έχει προταθεί ως υποκείμενος μηχανισμός συναίνεσης. Αυτός ο αλγόριθμος χρησιμοποιεί μια πιο ελαφριά τεχνική μετάδοσης μηνυμάτων, η οποία είχε ως αποτέλεσμα την **ανώτερη απόδοση** αυτής της προσέγγισης. Το Aura και το Clique [71], τα οποία εφαρμόστηκαν αρχικά στο Ethereum σαν ιδιωτικά δίκτυα, είναι δύο υλοποιήσεις του αλγορίθμου PoA.

Τόσο το Aura όσο και το Clique χρησιμοποιούν μια μέθοδο πρότασης μπλοκ όπου μια αξιόπιστη αρχή, όπως ο σημερινός ηγέτης εξόρυξης, προτείνει ένα νέο μπλοκ. Στη συνέχεια, η διαδικασία αποδοχής μπλοκ εκτελείται από το Aura, το οποίο δεν είναι απαραίτητο στην υλοποίηση του Clique.

Η PoA πραγματοποιείται σε πολλές χρονικές διαιρέσεις και σε κάθε διάστημα, οι αρχές προτείνουν μπλοκ με κυκλικό τρόπο. Μόλις η πλειοψηφία των εξουσιοδοτημένων κόμβων υπογράψει ένα προτεινόμενο μπλοκ, γίνεται αποδεκτό. Πρέπει όμως να αναφερθεί πως, η διαδικασία των οξυδερκών αρχών οδηγεί σε μια κεντρική οργάνωση (central entity) στον PoA, καθιστώντας αυτή τη λύση **κατάλληλη για ιδιωτικές κοινοπραξίες**. [72], [73].

1.7 Κόμβοι (Nodes)

1.7.1 Εισαγωγή

Οι κόμβοι, γνωστοί και ως peers, είναι συσκευές που παρακολουθούν τις συναλλαγές και γενικότερα τις πληροφορίες που «ζουν» στο δίκτυο του blockchain. Γενικά, κάθε ιθαγενές κρυπτονόμισμα έχει το δικό του blockchain και τους δικούς του κόμβους[31].

Ένα δίκτυο blockchain P2P αποτελείται από ομότιμους κόμβους οπού ο καθένας διατηρεί ένα πλήρες αντίγραφο όλων των μπλόκς του δικτύου, το οποίο χρησιμεύει ως κοινόχρηστο καθολικό. Κάθε blockchain επαληθεύει μπλοκ χρησιμοποιώντας μια μοναδική μέθοδο συναίνεσης (*βλ. Κεφ. 1.6*) και μπορεί να απορρίψει ένα μπλοκ που δεν ακολουθεί το συμφωνημένο σύνολο κανόνων του δικτύου[31].

1.7.2 Full Nodes

Ένας πλήρης κόμβος επαληθεύει κάθε μπλοκ και συναλλαγή που διέρχεται από αυτόν. Αυτό το καταφέρνει συγκρίνοντάς τα δεδομένα που λαμβάνει με τους κανόνες συναίνεσης του δικτύου[30]. Εάν δεν τηρούνται οι προκαθορισμένοι κανόνες συναίνεσης, ο κόμβος θα απορρίψει ολόκληρο το μπλοκ. Για να σας δώσω ένα παράδειγμα, οι κανόνες συναίνεσης του Bitcoin σε απλοποιημένη μορφή είναι οι εξής:

- Το αποτέλεσμα μιας συναλλαγής δεν μπορεί να δαπανηθεί δύο φορές.
- Η μορφή των συναλλαγών και των μπλοκ πρέπει να είναι ακριβής.
- Ως ανταμοιβή προσθήκης ενός μπλοκ, παρέχεται στο εξορυκτή που κατάφερε να λύση το παζλ του PoW (*βλ. Κεφ. 1.6*) μια συγκεκριμένη ποσότητα bitcoin.

Ένας πλήρης κόμβος θα κατεβάσει ολόκληρο το ιστορικό blockchain, το οποίο για να συντηρηθεί θα χρειαστεί πολλή υπολογιστική ισχύ και μνήμη. Περνώντας το ρολό ενός πλήρους κόμβου, ένας χρήστης βοηθάει το δίκτυο πραγματοποιώντας συνδέσεις με άλλους

πλήρεις κόμβους καθώς όλοι μαζί αποδέχονται, επικυρώνουν και αποθηκεύουν μπλόκς και συναλλαγές[30].

Οι πλήρεις κόμβοι πρέπει να έχουν ένα πλήρες αντίγραφο του blockchain[30] για να μπορούν να εγγυώνται για την λήψη κάθε απόφασης που έχει συμβεί ποτέ στο blockchain, διασφαλίζοντας με αυτόν τον τρόπο ότι το δίκτυο παραμένει αποκεντρωμένο. Αυτό παρέχει επίσης επιπλέον επίπεδα προστασίας και διασφαλίζει ότι κανένα σημείο ευπάθειας δεν εκτίθεται σε επίθεση.

Οι πλήρεις κόμβοι είναι επίσης ανεξάρτητοι (trustless)[30], πράγμα που σημαίνει ότι κάθε μπλοκ ή συναλλαγή που παραβιάζει τους κανόνες συναίνεσης θα απορριφθεί, ακόμη και αν συμφωνεί κάθε άλλος κόμβος στο δίκτυο.

1.7.3 Light Nodes

Οι ελαφριοί κόμβοι διαφέρουν από τους πλήρης σε μερικούς τομείς. Η Απλοποιημένη Επαλήθευση Πληρωμών (SPV) είναι ένας μηχανισμός που χρησιμοποιείται από τους ελαφριούς κόμβους (γνωστούς και ως lightweight clients) για την επικύρωση συναλλαγών[30]. Αυτή η λύση εξαλείφει την ανάγκη για έναν χρήστη να κατεβάσει ολόκληρο το ιστορικό του blockchain προκειμένου να ελέγξει εάν μια συναλλαγή έχει συμπεριληφθεί σε ένα μπλοκ ή όχι.

Αυτοί οι κόμβοι απλώς κατεβάζουν τις επικεφαλίδες (Block Headers) όλων των μπλόκς στο blockchain, γεγονός που τους καθιστά ιδανικούς για νεοεισερχόμενα μέλη και συναλλαγές μικρής κλίμακας[30].

Το SPV επιτρέπει στους πλήρεις κόμβους να εξυπηρετούν τους ελαφριούς κόμβους, επιτρέποντάς τους να ενταχθούν στο δίκτυο και να στείλουν τις συναλλαγές τους, καθώς και ειδοποιώντας τους χρήστες τους όταν πραγματοποιείται μια συναλλαγή που τους επηρεάζει[30].

Τέλος, το SPV βασίζεται σε πλήρεις κόμβους για την επαλήθευση της νομιμότητας των μπλοκ και των συναλλαγών. Δίχως του πλήρης κόμβους οι ελαφριοί δεν θα μπορούσαν να λειτουργήσουν[30].

1.8 Διακλάδωση (Forks)

1.8.1 Εισαγωγή

Fork (Διακλάδωση) είναι μια λέξη που αναφέρεται σε μια παραλλαγή ή απόκλιση ή διαχωρισμό από την κύρια ή καθιερωμένη τεχνολογία, δομή, πολιτική ή περιβάλλον. Τα μπλόκς του δικτύου Bitcoin δημιουργούνται μέσω της διαδικασίας της εξόρυξης(mining), η οποία περιλαμβάνει τη συμπερίληψη επαληθευμένων συναλλαγών σε ένα μπλοκ και την επίλυση ενός δύσκολου μαθηματικού-αλγοριθμικού παζλ (Proof-of-Work) για αυτό το μπλοκ[28].

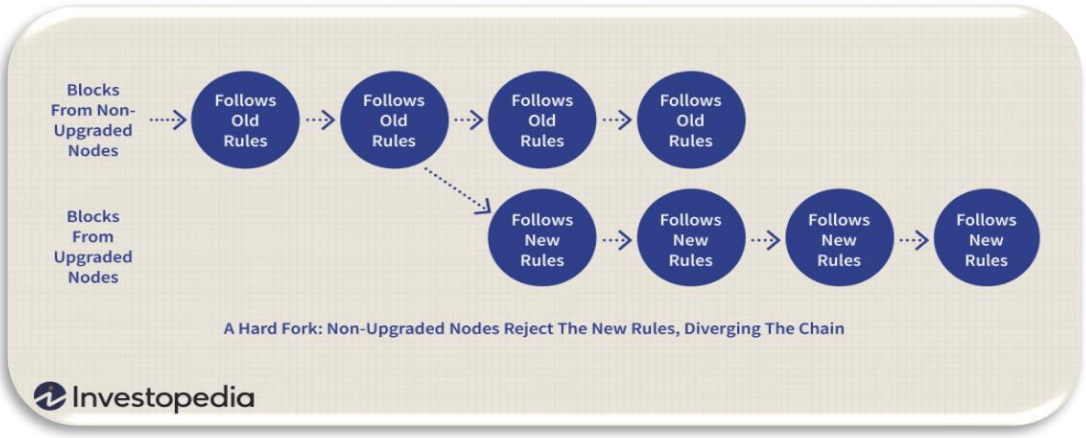
Ένας διαχωρισμός (διακλάδωση) στο blockchain συμβαίνει όταν δύο εξορύκτες(miners) εντοπίζουν και δημοσιεύουν ανεξάρτητα και ταυτόχρονα ένα νέο μπλοκ. Η αντίφαση είναι μονάχα προσωρινή και δεν θα επηρεάσει τα ακόλουθα μπλόκς, διότι θα επέμβει ο μηχανισμός συναίνεσης και θα συντονίσει το δίκτυο . Οι διακλαδώσεις είναι ένα συχνό φαινόμενο της τεχνολογίας blockchain, επειδή μεταφέρουν τους παρόντες κανόνες σε ένα νέο σύνολο προγραμματισμένων κανόνων. Οι καθυστερήσεις, οι διαφορετικές τακτικές εξόρυξης και οι διαφορετικές προοπτικές πολιτικής συμβάλλουν στη δημιουργία πιρουνιών (forks) στο blockchain (*βλ. Εικόνα 1.8.1*) , από το τελευταίο, γνωστό μπλοκ. Αυτό επηρεάζει την ασφάλεια των εφαρμογών του blockchain καθώς και τις υποθέσεις τοπολογίας δικτύου σε περιβάλλον χωρίς άδεια, γνωστό ως ελεύθερο δίκτυο (*βλ. Κεφ. 1.5.1*).

Στις περισσότερες περιπτώσεις, **μια διακλάδωση (fork) εμφανίζεται** όταν δύο ξεχωριστοί ανθρακωρύχοι(miners) λύνουν το πρόβλημα απόδειξης εργασίας την ίδια στιγμή, αλλά μπορεί επίσης να συμβεί όταν οι ανθρακωρύχοι λύνουν και διατηρούν νέα μπλοκ αντί να τα μεταδίδουν αμέσως για να αποκτήσουν πλεονέκτημα στην ανακάλυψη του επόμενου μπλοκ. Όπως περιγράφεται λεπτομερώς στο, όταν συμβαίνει αυτό τα νέα μπλοκ διαδίδονται πλημμυρίζοντας το δίκτυο[27].

1.8.2 Σκληρή Διακλάδωση (Hard Fork)

Ένα hard fork προκύπτει όταν υπάρξει μια ενημέρωση στον τρόπο λειτουργίας του blockchain που δεν είναι συμβατή με την τρέχουσα εφαρμογή blockchain. Αυτό μπορεί να συμβεί ως συνέπεια μιας αλλαγής σε κάποιο κανόνα στο κύριο λογισμικό (client software) ή στις εικονικές μηχανές (virtual machines) που επαληθεύουν τις συναλλαγές και τα μπλόκς. Όταν συμβεί αυτό οι κόμβοι που υποστηρίζουν τον παλιό κανόνα πλέον θα βλέπουν τα καινούργια μπλόκς ως μη έγκυρα ή περιττά, κάτι που αναγκάζει όλους τους εμπλεκόμενους κόμβους να ενημερώσουν το λογισμικό τους ώστε να υπάρχει συγχρονισμός και κοινή συναίνεση στο δίκτυο.

Η σκληρή διακλάδωση (hard fork) του Bitcoin, το Bitcoin Cash, είναι ένα αξιοσημείωτο παράδειγμα τέτοιου συμβάντος. Φανταστείτε σε κάποιο X μπλοκ του Bitcoin blockchain, ορισμένοι κανόνες τροποποιήθηκαν ή προτάθηκε μια ενημέρωση πρωτοκόλλου. Η αποδοχή και η συμπερίληψη αυτών των τροποποιήσεων πρέπει να ψηφιστούν από όλους τους κόμβους εξόρυξης (ομότιμοι σε ένα ελεύθερο δίκτυο peer-to-peer). Εάν επιλεγεί η υπάρχουσα πρόταση, δεν θα υπάρξουν αλλαγές. Ωστόσο, εάν η αναβάθμιση που έχει επιλεχθεί, εφαρμοστεί δίχως την συναίνεση όλων των κόμβων, η αλυσίδα μπορεί να αντιμετωπίσει μια "σκληρή διακλάδωση". Οποιοδήποτε μπλοκ εξορύχθηκε χρησιμοποιώντας προηγούμενα πρωτόκολλα, για παράδειγμα, θα καθιστούνταν άκυρο και δεν θα μπορούσε να συνδεθεί με ένα μπλοκ που θα εξορυχθεί μελλοντικά με το νέο πρωτόκολλο, όπως φαίνεται στο [29] και στην παρακάτω εικόνα.

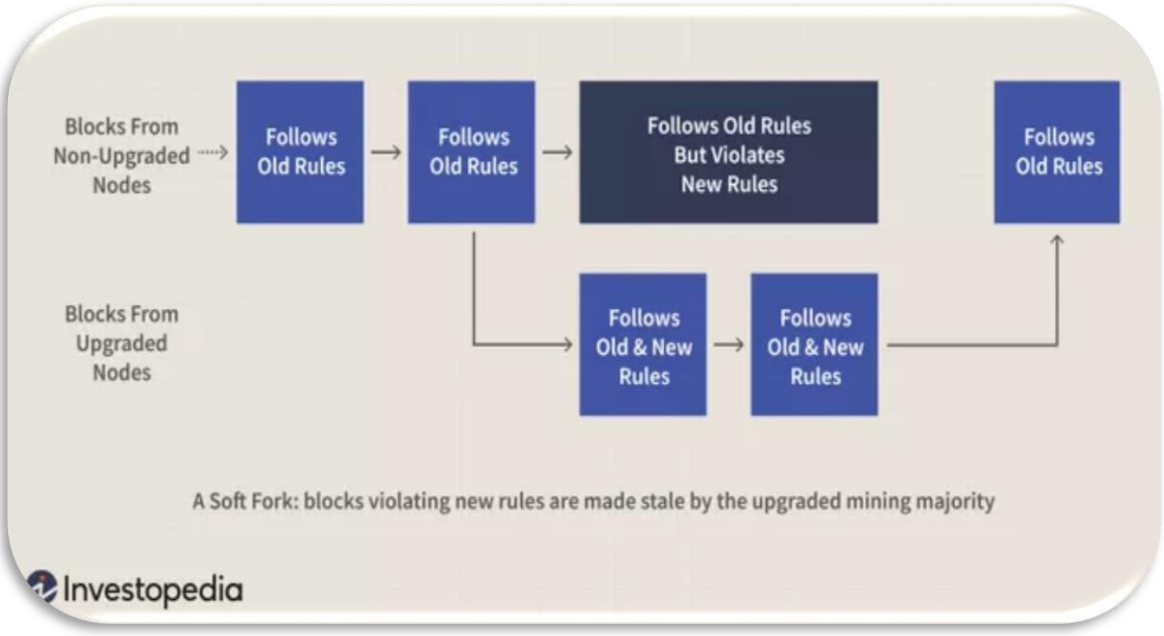


Εικόνα 1.8.1 – *Σκληρή Διακλάδωση. Διαθέσιμο στο Διαδίκτυο: https://www.investopedia.com/terms/h/hard-fork.asp*

1.8.3 Μαλακή Διακλάδωση (Soft Fork)

Ένα soft fork, σε αντίθεση με τη σκληρή διακλάδωση, είναι μια ενημερωμένη έκδοση συμβατή προς τα πίσω στην οποία οι αναβαθμισμένοι κόμβοι μπορούν να επικοινωνούν με μη αναβαθμισμένους κόμβους. Αυτό είναι εφικτό μόνο εάν η νέα, βελτιωμένη έκδοση δεν έρχεται σε αντίθεση με την παλιά.

Τόσο στο blockchain Bitcoin όσο και στο Ethereum υπάρχουν τέτοιου είδους παραδείγματα πιρουνιών, με ανακοινωμένες και ενσωματωμένες βελτιώσεις που είναι συμβατές με παλιότερες. Αυτή η μέθοδος ενδέχεται να προκαλέσει **αποκλίσεις** όταν ένα μπλοκ που εξορύσσεται χρησιμοποιώντας παλιούς κανόνες δεν υποβάλλεται σε επεξεργασία από ενημερωμένους κόμβους, αλλά αντ' αυτού επεξεργάζεται από μη αναβαθμισμένους κόμβους, με αποτέλεσμα τον αποσυγχρονισμό χωρίς να το καταλάβουν οι χρήστες.



Εικόνα 1.8.2 – *Μαλακή Διακλάδωση. Διαθέσιμο στο Διαδίκτυο: https://www.investopedia.com/terms/s/soft-fork.asp*

1.9 Δημοφιλή Blockchains

1.9.1 Bitcoin

1.9.1.1 Εισαγωγή

Όταν μια εργασία με τίτλο "Bitcoin: A Peer-to-Peer Electronic Cash System" δημοσιεύτηκε το 2008, συντάχθηκε από ένα άτομο που χρησιμοποιεί το ψευδώνυμο Satoshi Nakamoto. Για να αναπτύξει ένα εντελώς αποκεντρωμένο ηλεκτρονικό σύστημα μετρητών που δεν εξαρτάται από μια κεντρική αρχή για την έκδοση νομισμάτων, τον διακανονισμό και την επικύρωση συναλλαγών, η Nakamoto συγχώνευσε μια σειρά από ιδέες του παρελθόντος, όπως το b-money και το HashCash.

Η **βασική καινοτομία** ήταν η διενέργεια παγκόσμιων "εκλογών" κάθε 10 λεπτά χρησιμοποιώντας ένα κατανεμημένο υπολογιστικό σύστημα (γνωστό και ως μέθοδος "απόδειξης εργασίας", Proof of Work (*βλ. Κεφ. 1.6.3*), το οποίο επέτρεπε στο αποκεντρωμένο δίκτυο να καταλήξει σε συμφωνία σχετικά με την κατάσταση των συναλλαγών. Αυτό λύνει κομψά το ζήτημα της διπλής δαπάνης (Double-Spending) όπου μια μονάδα ενιαίου νομίσματος μπορεί να δαπανηθεί δύο φορές[74].

Εικόνα 1.9.1 – *Σύμβολο του Bitcoin. Διαθέσιμο στο Διαδίκτυο: https://bitcoinmagazine.com/markets/bitcoin-back-below-30000-after-a-record-8-weeks-in-the-red*

1.9.1.2 Τρόποι σύνδεσης

Για χρησιμοποιήσει κάνεις το Bitcoin δεν έχει πάρα να κατεβάσει μια εφαρμογή ή να συνδεθεί μέσω μιας διαδικτυακής εφαρμογής. Επειδή το bitcoin είναι πρότυπο (standard), υπάρχουν πολλές διαφορετικές εφαρμογές σχεδιασμένες για τον χρήστη (client software)[74].

Υπάρχουν τρεις βασικές κατηγορίες:

1. Full Client (Πλήρης Έκδοση)
2. Lightweight Client (Ελαφριά Έκδοση)
3. Web Client (Διαδικτυακή Έκδοση)

1.9.2.3 Χαρακτηριστικά

Το δίκτυο (P2P, Peer-to-Peer):

Το Bitcoin βασίζεται στο κύριο δίκτυο Bitcoin, το οποίο είναι προσβάσιμο μέσω της θύρας **TCP 8333** και χρησιμοποιεί ένα P2P πρωτόκολλο που η έκδοση μπορεί να ενημερωθεί[74].

Συναλλαγές:

Οι **συναλλαγές P2PKH** (Pay-to-Public-Key-Hash) αποτελούν τον συντριπτικό όγκο των συναλλαγών που πραγματοποιούνται στο δίκτυο bitcoin. Αυτές έχουν ένα κώδικα (script) κλειδώματος που περιορίζει την έξοδο με έναν κατακερματισμό(hash) ενός δημόσιου κλειδιού, περισσότερο γνωστές ως διευθύνσεις bitcoin[74].

Δομή Δεδομένων:

Προκειμένου να δημιουργηθεί ένα ολοκληρωμένο ψηφιακό αποτύπωμα της πλήρους συλλογής συναλλαγών, το bitcoin χρησιμοποιεί δέντρα συγχώνευσης (**Merkle trees**), τα οποία καθιστούν πολύ γρήγορο τον έλεγχο εάν μια συναλλαγή είναι μέρος ενός μπλοκ[74].

Αλγόριθμος Συναίνεσης:

Το Bitcoin χρησιμοποιεί το συναινετικό μοντέλο **Nakamoto Consensus**, το οποίο χρησιμοποιεί διαδοχικά μπλοκ μίας υπογραφής (sequential single-signature blocks) που σταθμίζονται από το **PoW** (*βλ. Κεφ. 1.6.3*) για τον προσδιορισμό της μεγαλύτερης αλυσίδας και ως εκ τούτου της τρέχουσας κατάστασης (Με αυτόν τον τρόπο επιλύεται το πρόβλημα «Double-Spending[74].

Λογισμικό Κόμβων:

Το κύριο λογισμικό που χρησιμοποιείται από τους πλήρης κόμβους είναι το **Bitcoin Core** client[74].

1.9.2 Ethereum

1.9.2.1 Εισαγωγή

Ο όρος "παγκόσμιος υπολογιστής" χρησιμοποιείται συχνά για να περιγράψει το Ethereum. Τι ακριβώς σημαίνει όμως; Ας ξεκινήσουμε με μια εξήγηση που εστιάζει στην επιστήμη των υπολογιστών και, στη συνέχεια, θα συγκρίνουμε το Ethereum με το Bitcoin.

Το Ethereum είναι μια ντετερμινιστική αλλά ουσιαστικά απεριόριστη μηχανή κατάστασης στην επιστήμη των υπολογιστών, που αποτελείται από μια παγκόσμια προσβάσιμη κατάσταση singleton και μια εικονική μηχανή που εφαρμόζει τροποποιήσεις σε αυτήν.

Πιο πρακτικά, το Ethereum είναι μια παγκόσμια αποκεντρωμένη πλατφόρμα υπολογιστών , ανοιχτού κώδικα που εκτελεί εφαρμογές έξυπνων συμβολαίων (Smart Contracts). Χρησιμοποιεί τη τεχνολογία blockchain για τον συγχρονισμό και τη διατήρηση των αλλαγών κατάστασης του συστήματος, καθώς και ένα κρυπτονόμισμα που ονομάζεται αιθέρας (ether) για την παρακολούθηση και τον περιορισμό του κόστους των πόρων εκτέλεσης[25].

Οι προγραμματιστές μπορούν να χρησιμοποιήσουν την πλατφόρμα Ethereum για να δημιουργήσουν εξελιγμένες αποκεντρωμένες εφαρμογές (Dapps) με ενσωματωμένες οικονομικές λειτουργίες. Ακόμα, μειώνει ή εξαλείφει τη λογοκρισία και ορισμένους κινδύνους αντισυμβαλλομένου, ενώ παρέχει υψηλή διαθεσιμότητα, δυνατότητα ελέγχου, διαφάνεια και αμεροληψία[25].

1.9.2.2 Σύγκριση με Bitcoin

Πολλοί χρήστες που επισκέπτονται το Ethereum θα έχουν ήδη δραστηριοποιηθεί με μερικά κρυπτονομίσματα, ιδιαίτερα με το Bitcoin. Το Ethereum έχει πολλά κοινά χαρακτηριστικά με το Bitcoin[25]:

- ✓ Ενα δίκτυο peer-to-peer

- ✓ Έναν βυζαντινό αλγόριθμο συναίνεσης με ανοχή σε σφάλματα (Byzantine fault–tolerant) για συγχρονισμό ενημερώσεων κατάστασης (ένα PoW blockchain)

- ✓ Τη χρήση πρωτόγονων κρυπτογραφικών στοιχείων όπως ψηφιακές υπογραφές και κατακερματισμούς (hashes) και ένα ψηφιακό νόμισμα (ether).

Ωστόσο, υπάρχουν πολλές απόψεις που υποστηρίζουν ότι, ο στόχος και ο σχεδιασμός του Ethereum είναι αρκετά διαφορετικός από εκείνους των υπολοίπων δημοσίων blockchains που προηγήθηκαν, συμπεριλαμβανομένου του Bitcoin[25].

Ο πρωταρχικός στόχος του Ethereum δεν είναι να χρησιμεύσει ως δίκτυο πληρωμών για ψηφιακά νομίσματα. Παρόλα αυτά, ένα ψηφιακό νόμισμα είναι απαραίτητο για τη λειτουργία του Ethereum, προορίζεται μόνο να χρησιμοποιηθεί ως πληρωμή για τη χρήση της πλατφόρμας Ethereum ως παγκόσμιου υπολογιστή[25].

Σε αντίθεση με το Bitcoin, που έχει μια αρκετά περιορισμένη γλώσσα δέσμης ενεργειών (scripting language), το Ethereum προορίζεται να είναι ένα προγραμματιζόμενο blockchain γενικής χρήσης που διαθέτει μια εικονική μηχανή ικανή να εκτελεί κώδικα οποιουδήποτε μεγέθους και πολυπλοκότητας. Ενώ η γλώσσα Script του Bitcoin περιορίζεται σε μια βασική αξιολόγηση true/false των συνθηκών δαπανών, η γλώσσα του Ethereum είναι πλήρης Turing (*Turing complete*), υπονοώντας ότι το Ethereum μπορεί να χρησιμοποιηθεί ως υπολογιστής γενικής χρήσης[25].

1.9.2.3 Χαρακτηριστικά του Ethereum

Το δίκτυο (P2P, Peer-to-Peer):

Το Ethereum βασίζεται στο κύριο δίκτυο Ethereum, το οποίο είναι προσβάσιμο μέσω της θύρας TCP 30303 και χρησιμοποιεί το πρωτόκολλο *ÐΞVp2p*[25].

Συναλλαγές:

Οι συναλλαγές Ethereum είναι μηνύματα δικτύου που περιλαμβάνουν, μεταξύ άλλων, έναν αποστολέα, τον παραλήπτη, την αξία και το ωφέλιμο φορτίο δεδομένων[25].

Μηχανή Κατάστασης (State Machine):

Ο εικονικός υπολογιστής Ethereum (EVM), μια εικονική μηχανή που βασίζεται σε στοίβα (stack-based) και εκτελεί bytecode, επεξεργάζεται τις αλλαγές της κατάστασης του Ethereum. Τα έξυπνα συμβόλαια είναι προγράμματα του EVM γραμμένα σε γλώσσες προγραμματισμού υψηλού επιπέδου (για παράδειγμα, Solidity) και μεταγλωττίζονται σε bytecode με την χρήση ενός compiler για εκτέλεση στο EVM[25].

Δομή Δεδομένων:

Η κατάσταση του Ethereum διατηρείται τοπικά σε κάθε κόμβο με τη μορφή μιας βάσης δεδομένων (συνήθως το LevelDB της Google), η οποία αποθηκεύει τις συναλλαγές και τις πληροφορίες του συστήματος σε μια σειριακή κατακερματισμένη δομή δεδομένων γνωστή ως Merkle Patricia Tree[25].

Αλγόριθμος Συναίνεσης:

Το Ethereum χρησιμοποιεί το συναινετικό μοντέλο Nakamoto Consensus, το οποίο χρησιμοποιεί διαδοχικά μπλοκ μίας υπογραφής (sequential single-signature blocks) που σταθμίζονται από το PoW (*βλ. Κεφ. 1.6.3*) για τον προσδιορισμό της μεγαλύτερης αλυσίδας και ως εκ τούτου της τρέχουσας κατάστασης (Με αυτόν τον τρόπο επιλύεται το πρόβλημα «Double-Spending"). Ωστόσο, στο εγγύς μέλλον, υπάρχουν σχέδια για μετάβαση σε ένα σταθμισμένο σύστημα ψηφοφορίας PoS που ονομάζεται Casper[25].

Ασφάλεια Οικονομίας:

Το Ethereum χρησιμοποιεί τώρα μια μέθοδο PoW γνωστή ως Ethash, ωστόσο αυτή θα καταργηθεί σταδιακά όταν το δίκτυο μεταβεί στο PoS[25].

Λογισμικό Κόμβων:

Το Go-Ethereum (Geth, γραμμένο με την γλώσσα προγραμματισμού *GO)* και το Parity (γλώσσα προγραμματισμού *Rust*) είναι δύο από τις πιο δημοφιλείς διαλειτουργικές εφαρμογές λογισμικού κόμβων για το Ethereum[25].

1.9.3 Solana

1.9.3.1 Σύντομη Περιγραφή

Το Solana project δημοσιεύτηκε το 2019 από τον Anatoly Yakovenko, ειδικό στις τηλεπικοινωνίες και μεγάλο οπαδό των τεχνολογιών crypto, οποίος ονόμασε το project από μια παράλια στην Καλιφόρνια[75].

Το πρόβλημα που προσπαθεί να λύσει το Solana είναι η επεκτασιμότητα, διότι όπως έχει παρατηρηθεί από τα προηγούμενα δυο δημοφιλή blockchains, μετά από ένα σημείο το δίκτυο αρχίζει να γίνεται αρκετά αργό και ακριβό[75].

Ο τρόπος με τον οποίο το Solana αντιμετωπίζει αυτό το πρόβλημα είναι με μια παραλλαγή του αλγορίθμου PoS (*βλ. Κεφ. 1.6.4*) που ονομάζεται Proof of History (*βλ. Κεφ. 1.6.7*). Με αυτόν τον αλγόριθμο το Solana μπορεί να διαχειριστεί μεγάλους όγκους συναλλαγών σε αρκετά μικρό χρόνο και ταυτόχρονα να μην απαιτεί πολύ επεξεργαστική ισχύ. Κάνοντας το πολύ πιο ελκυστικό σε σχέση με το Ethereum[75].

Ωστόσο, πρέπει να αναφέρουμε ότι, όπως όλες οι τεχνολογίες έχουν τα μειονεκτήματα τους έτσι και το Solana δεν είναι τέλειο. Όπως έχει αναφέρει ο ιδρυτής του Ethereum Vitalik Buterin, υπάρχει ένα τρίλημμα (trilemma) όσον αφορά τα κυριά χαρακτηριστικά οποιουδήποτε blockchain. Οι τρεις πτυχές αυτού του τριλήμματος είναι:

1. Ασφάλεια
2. Επεκτασιμότητα
3. Αποκεντροποίηση

Συνοπτικά, αυτό σημαίνει ότι κανένα blockchain δεν μπορεί να υπερισχύει και στις τρεις πτυχές. Επιστρέφοντας πίσω στο Solana, το Solana αποφάσισε να θυσιάσει την υψηλή ασφάλεια ώστε να αποκτήσει υψηλότερη επεκτασιμότητα με αποτελέσματα να είναι πιο ευάλωτο σε επιθέσεις (κυρίως DDoS). Πράγματι, κατά την συγγραφή της εργασίας, έχουν ήδη πραγματοποιηθεί τουλάχιστον επτά πολύωρες διακοπές του δικτύου λόγω κάποιου προβλήματος στο κώδικα ή κάποιας επίθεσης.

Εν κατακλείδι, το Solana είναι ένα πολύ ενδιαφέρον project που ήδη έχει κερδίσει το ενδιαφέρον από ένα μεγάλο μέρος του crypto-κόσμου αλλά χρειάζεται σημαντική βελτίωση όσον αφορά την ασφάλεια και την σταθερότητα λειτουργίας του.

Εικόνα 1.9.3 – *Σύμβολο του Solana. Διαθέσιμο στο Διαδίκτυο: https://news.coincu.com/61251-solana-looks-like-it-will-integrate-a-fees-market-similar-to-ethereum/*

2. Κρυπτογραφία

2.1 Εισαγωγή

Η κρυπτογραφία, είναι ένα πεδίο που έχει ως **βάση τα μαθηματικά** και χρησιμοποιείται εκτενώς στην **ασφάλεια των υπολογιστών**, είναι μία από τις **βασικότερες τεχνολογίες** του όλων των **blockchain δικτυών**[25]. Η κρυπτογραφία(cryptography) προέρχεται από την ελληνική γλώσσα και σημαίνει «μυστική γραφή». Ωστόσο, η κρυπτογραφία περιλαμβάνει πολλά περισσότερα από μια απλή κρυφή-μυστική γραφή, η οποία αναφέρεται ως κρυπτογράφηση. Η κρυπτογραφία μπορεί επίσης να χρησιμοποιηθεί για την **επαλήθευση της γνώσης ενός μυστικού χωρίς την αποκάλυψή του** (για παράδειγμα, με ψηφιακή υπογραφή) ή για τη **διαπίστωση της αυθεντικότητας των δεδομένων** (για παράδειγμα, με ψηφιακά δακτυλικά αποτυπώματα, γνωστά και ως "hashes"). Αυτές οι κρυπτογραφικές αποδείξεις είναι μαθηματικές τεχνικές που είναι απαραίτητες για τη λειτουργικότητα της πλατφόρμας Ethereum και όλων γενικότερα των blockchain δικτυών[25].

Υπάρχουν και προηγμένες κρυπτογραφικές μέθοδοι, όπως αποδείξεις μηδενικής γνώσης (**Z**ero **K**nowledge **P**roof, **ZKP**) και ομομορφική κρυπτογράφηση (Homomorphic Encryption), οι οποίες επιτρέπουν την καταγραφή ορισμένων κρυπτογραφημένων υπολογισμών στο blockchain, διατηρώντας παράλληλα τη κοινή συναίνεση. Αν και έχουν προγραμματιστεί, υπάρχουν λίγες εφαρμογές που τις αξιοποιούν (ZCash[53], Monero[54]).

2.2 Κλασσική Κρυπτογραφία

2.2.1 Συμμετρική Κρυπτογράφηση (Symmetric Encryption)

Η συμμετρική κρυπτογράφηση επιτυγχάνεται χρησιμοποιώντας ένα κλειδί που μοιράζεται τόσο ο αποστολέας όσο και ο παραλήπτης[55]. Ένα από τα κύρια ζητήματα με την ασύμμετρη κρυπτογράφηση είναι ότι οι λειτουργίες (ιδιαίτερα η αποκρυπτογράφηση) είναι ιδιαίτερα δύσκολο να διεξαχθούν λόγω της μεγάλης υπολογιστικής ισχύος που απαιτείται για την εφαρμογή τέτοιων αλγορίθμων στα προτεινόμενα επίπεδα ασφάλειας.

Ως αποτέλεσμα αυτού του προβλήματος, η ασύμμετρη κρυπτογράφηση δεν είναι κατάλληλη για την αποστολή μεγάλων μηνυμάτων και είναι προτιμότερο να αλλάζετε το κλειδί[55]. Ως αποτέλεσμα, καθιερώνουμε ένα πιο ομαλό σύστημα ανταλλαγής κρυπτογραφημένων επικοινωνιών χρησιμοποιώντας συμμετρική κρυπτογράφηση/αποκρυπτογράφηση που διεξάγεται με το ίδιο κοινόχρηστο κλειδί[55].

2.2.1.1 Αλγόριθμος DES

ΙΣΤΟΡΙΚΗ ΑΝΑΦΟΡΑ

Το Πρότυπο Κρυπτογράφησης Δεδομένων (**D**ata **E**ncryption **S**tandard ,DES) είναι ο πρώτος αλγόριθμος που θα αναφερθεί σε αυτό το κεφάλαιο. Το Εθνικό Γραφείο Προτύπων (NBS), στη συνέχεια γνωστό ως Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST), χρειαζόταν έναν αλγόριθμο για να υιοθετηθεί ως εθνικό πρότυπο το 1973. Η IBM πρόσφερε τον Lucifer, μια συμμετρική μέθοδο, στο NIST το 1974, και διαβιβάστηκε στην Υπηρεσία Εθνικής Ασφάλειας (NSA). Μετονομάστηκε σε DES μετά από κάποιες έρευνες και αλλαγές. Το DES καθιερώθηκε ως εθνικό πρότυπο το 1977 και χρησιμοποιήθηκε ευρέως για κρυπτογράφηση δεδομένων σε περιβάλλοντα ηλεκτρονικού εμπορίου, όπως ο τραπεζικός τομέας[55].

Μέσα στον ακαδημαϊκό και επαγγελματικό κόσμο των κρυπτολόγων, υπήρξαν έντονες συζητήσεις για τη δύναμη του DES. Η κριτική προήλθε από το μικρό μήκος κλειδιού του

αλγόριθμου και την αβεβαιότητα ότι, μετά από εξέταση της NSA, μπορεί να είναι ευάλωτος σε μια καταπακτή (trapdoor), την οποία η NSA έβαλε ειδικά στο DES για να παρακολουθεί κρυπτογραφημένες επικοινωνίες[55].

Τώρα που έχουμε μια καλύτερη κατανόηση της ιστορίας αυτού του προδρόμου των σημερινών συμμετρικών αλγορίθμων, μπορούμε να προχωρήσουμε στην εξήγηση του λογικού και μαθηματικού σχεδιασμού του[55].

ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ

Θα γίνει μια σύντομη περιγραφή στο *απλό DES* (Simple DES) οποίος δεν κάτι παραπάνω από μια απλοποιημένη εκδοχή του κανονικού DES. Αυτή η απλοποιημένη έκδοση είναι ένα μπλοκ cypher, το οποίο υπονοεί ότι το απλό κείμενο διαχωρίζεται αρχικά σε μπλόκς. Η ιδέα είναι ότι μπορούμε να κοιτάμε μόνο ένα μπλοκ τη φορά, καθώς το καθένα είναι κρυπτογραφημένο ανεξάρτητα[55].

Το κλειδί [K] έχει μήκος 9 bit, ενώ το μήνυμα [M] έχει μήκος 12 bit. Το S-Box, το οποίο είναι παρόμοιο με το DES S-Box, είναι η πιο σημαντική πτυχή του αλγορίθμου. Η πραγματική πολυπλοκότητα και η μη γραμμική συνάρτηση των συμμετρικών αλγορίθμων μπορεί να βρεθεί εδώ. Το υπόλοιπο της διαδικασίας είναι απλώς μεταθέσεις και μετατοπίσεις bit, τις οποίες ένας τυπικός υπολογιστής μπορεί να πραγματοποιήσει αυτόματα, επομένως δεν χρειάζεται να ασχοληθούμε με αυτό[55].

Σε αυτήν την περίπτωση, ένα S-Box είναι ένας πίνακας 4X16 με 6 bit ως είσοδο και 4 bit ως έξοδο. Όλες οι πρόσφατες μέθοδοι συμμετρικής κρυπτογράφησης, όπως οι DES, Triple DES, Bluefish και AES, περιλαμβάνουν το S-Box[55].

ΑΔΥΝΑΜΙΕΣ ΑΛΓΟΡΙΘΜΟΥ

- **Επίθεση ωμής-βίας** (Brute-force assault): Αυτός είναι ο πιο βασικός τύπος επίθεσης για οποιοδήποτε γνωστό cypher και συνεπάγεται στην εξάντληση όλων των δυνατών επιλογών για να αποκτήσετε το κλειδί. Ο DES χρησιμοποιεί ένα κλειδί 56-bit, που σημαίνει ότι ένας εισβολέας θα πρέπει να δοκιμάσει και τους 72.057.594.037.927.936 πιθανούς συνδυασμούς (2^{56}). Ο DES θεωρείται ένας «σπασμένος» αλγόριθμος από τις αρχές της δεκαετίας του 1990[55].

• **Γραμμική κρυπτανάλυση** (Linear cryptanalysis): Πρόκειται για έναν στατιστικό τύπο επίθεσης που βασίζεται σε απλό κείμενο που είναι γνωστό. Δεν λειτουργεί πάντα, αλλά τις περισσότερες φορές λειτουργεί. Ο στόχος είναι να εργαστείτε προς τα πίσω από τη γνωστή είσοδο (απλό κείμενο) για να προσδιορίσετε το κλειδί κρυπτογράφησης και, ως αποτέλεσμα, όλες τις άλλες εξόδους που δημιουργούνται από αυτό το κλειδί[55].

• **Διαφορική κρυπτανάλυση** (Differential cryptanalysis): Αυτή η προσέγγιση είναι πιο περίπλοκη και απαιτεί εντοπισμό ορισμένων ελαττωμάτων του DES (παρόμοια με άλλους συμμετρικούς αλγόριθμους). Ξεκινώντας με ένα επιλεγμένο απλό κείμενο, αυτή η τεχνική επίθεσης προσπαθεί να βρει το απλό κείμενο ή το κλειδί. Σε αντίθεση με τη γραμμική κρυπτανάλυση, η οποία ξεκινά με απίθανο γνωστό απλό κείμενο, ο εισβολέας εργάζεται με απλό κείμενο που έχει επιλεγεί[55].

2.2.1.2 Αλγόριθμος AES

ΙΣΤΟΡΙΚΗ ΑΝΑΦΟΡΑ

Μετά από μια τριετή φάση δοκιμών μεταξύ των κρυπτογράφων, το NIST (η κυβέρνηση των ΗΠΑ) ενέκρινε τον AES, γνωστό και ως Rijndael, ως έναν αρκετά ισχυρό αλγόριθμο το 2001.

Πέντε φιναλίστ επιλέχθηκαν από τους 15 υποψήφιους που διαγωνίστηκαν για τον καλύτερο αλγόριθμο: MARS (IBM), RC6 (RSA Laboratories), Rijndael (Joan Daemen και Vincent Rijmen), Serpent (Ross Anderson και άλλοι) και Twofish (Ross Anderson και άλλοι). Όλοι οι διαγωνιζόμενοι ήταν δυνατοί, αλλά στο τέλος, ο Rijndael βγήκε νικητής[55].

Επειδή ο AES είναι ένας block cypher, μπορεί να χρησιμοποιηθεί με διάφορους τρόπους, συμπεριλαμβανομένων των ECB (όπως και ο DES), Cipher Block Chaining (CBC), Cypher Feedback Block (CFB), Output Feedback Block (OFB) και Counter (CTR)[55].

Υπάρχουν διαφορά μεγέθη κλειδιών που μπορεί να χρησιμοποιήσει ο AES: 128-bit, 192-bit και 256-bit. Ο στόχος του διαγωνισμού του NIST ήταν να βρει έναν αλγόριθμο που είχε μερικά πολύ ισχυρά χαρακτηριστικά, όπως το να μπορεί να λειτουργεί σε μπλοκ των 128 bit εισόδου

και να μπορεί να τρέχει σε μια ποικιλία υλικού, που κυμαίνονται από επεξεργαστές 8 bit (που είναι χρησιμοποιούνται επίσης στις έξυπνες κάρτες) έως αρχιτεκτονικές 32-bit, οι οποίες χρησιμοποιούνται συνήθως σε προσωπικούς υπολογιστές. Τέλος, θα πρέπει να είναι γρήγορος και αξιόπιστος[55].

ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ

Η παρούσα ανάλυση του AES θα βασιστεί στην υποδιαίρεση της λειτουργικότητας του αλγόριθμου σε 2 βασικά τμήματα (Key Expansion και First Add Round Key) και υστέρα το κάθε τμήμα θα διαιρεθεί σε 4 υπό-τμήματα (SubBytes transformation, ShiftRows transformation, MixColumn και τελος AddRoundKey).

Το **Key Expansion** (KE) δουλεύει ως εξής[55]:

1 Η είσοδος σταθερού κλειδιού των 128 bit επεκτείνεται σε μήκος κλειδιού ανάλογα με το μέγεθος του AES: 128, 192 ή 256.
2 Στη συνέχεια, δημιουργούνται τα δευτερεύοντα κλειδιά [K1], [K2],…[Kr] για την κρυπτογράφηση κάθε γύρου (γενικά προσθέτοντας XOR στον γύρο).
3 Ο AES χρησιμοποιεί μια συγκεκριμένη μέθοδο που ονομάζεται χρονοδιάγραμμα κλειδιών του Rijndael για να επεκτείνει ένα μικρό κύριο κλειδί (master key) σε έναν ορισμένο αριθμό κυκλικών κλειδιών.
4 Το **First Add Round Key** (F-ARK) δουλεύει ως εξής[55]:

Είναι η πρώτη λειτουργία του αλγορίθμου. Η διαδικασία χρησιμοποιεί ένα XOR κατά bit του τρέχοντος μπλοκ με ένα κομμάτι του μεγεθυσμένου κλειδιού για να προσθέσει το πρώτο κλειδί, [K1], στο AddRoundKey.

Οι γύροι **R_1 έως R_{n-1}** δουλεύουν ως εξής[55]:

Κάθε γύρος (εκτός από τον τελευταίο) χωρίζεται σε τέσσερα βήματα που ονομάζονται στρώματα και αποτελούνται από τα ακόλουθα:

1 Μετατροπή **SubBytes** (SB): Αυτό το βήμα είναι ένα θεμελιώδες μη γραμμικό βήμα, εκτελείται μέσω ενός συγκεκριμένου S-Box.
2 Μετατροπή **ShiftRows** (SR): Αυτό είναι ένα ανακάτεμα ενός bit που προκαλεί διάχυση σε πολλαπλούς γύρους.
3 Μετατροπή **MixColumns** (MC): Αυτό το βήμα έχει παρόμοιο εύρος με το SR αλλά είναι

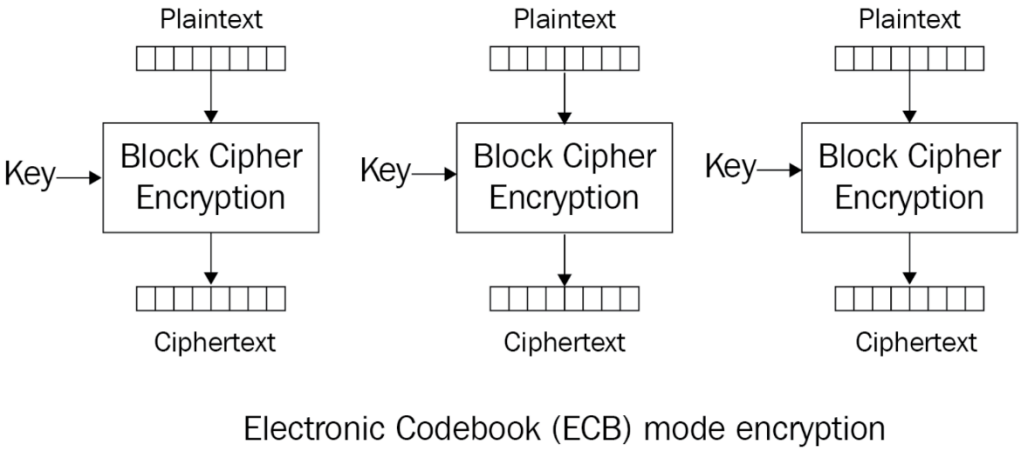
εφαρμόζεται στις στήλες.

4 **AddRoundKey** (ARK): Το κλειδί του κάθε γύρου γίνεται XORed με το αποτέλεσμα του προηγούμενο στρώμα.

ΑΔΥΝΑΜΙΕΣ ΑΛΓΟΡΙΘΜΟΥ

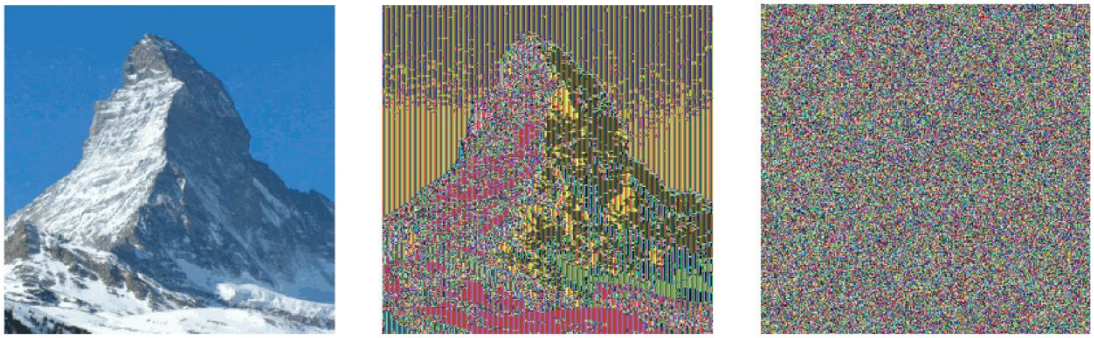
Ο AES κηρύχθηκε άτρωτος σε οποιαδήποτε γνωστή επίθεση στα έγγραφα της NSA και της NIST. Όμως η αλήθεια είναι πως τα ελαττώματα του. Στην πραγματικότητα, κάθε σύστημα που μπορεί να εφαρμοστεί έχει ελαττώματα.

Παρά το γεγονός ότι το AES είναι θεωρητικά άθραυστο (-x%), οποιοσδήποτε αλγόριθμος μπορεί να σπάσει μέσω ωμής βίας (brute force). Είναι γνωστό πως κατά το στάδιο της εφαρμογής μιας τεχνολογίας λάθη θα εμφανιστούν, στην περίπτωση μας αφού μιλάμε για κρυπτογραφικούς αλγορίθμους, παραβιάσεις(breaches). Δώστε προσοχή στο τι συμβαίνει όταν χρησιμοποιηθεί ο AES με λειτουργία ECB. Αυτή η απλή προσέγγιση περιλαμβάνει το διαχωρισμό του απλού κειμένου σε μπλοκ και τον υπολογισμό του κρυπτογραφημένου κειμένου για κάθε μπλοκ απλού κειμένου[55].



Εικόνα 2.2.1.2.1 *Η λειτουργία κρυπτογράφησης ECB. Διαθέσιμο στο βιβλίο [55], σελ. 86, Figure 2.30*

Ενδέχεται να υπάρξουν μεγάλες επιπλοκές εάν ο AES χρησιμοποιηθεί σε λειτουργία ECB. Στην εικόνα στη μέση έχει εφαρμοστεί ο αλγόριθμος AES σε λειτουργία ECB (μεταξύ της αρχικής εικόνας και αυτής που έχει κρυπτογραφηθεί με άλλη λειτουργία block). Όπως παρατηρείται, ακόμα κι αν οι πληροφορίες του όρους Cervino κρυπτογραφηθούν με την λειτουργία ECB, είναι εφικτό να τις αναγνωρίσουμε.



Εικόνα 3.2.1.2.2 *Η γνησιά εικόνα του ορούς Cervino, κρυπτογραφημένη με την λειτουργεία ECB, και με οποιαδήποτε άλλη λειτουργία. Διαθέσιμο στο βιβλίο [55], σελ. 87, Figure 2.31*

2.2.2 Ασύμμετρη Κρυπτογράφηση (Asymmetric Encryption)

Η ασύμμετρη κρυπτογράφηση κρυπτογραφεί και αποκρυπτογραφεί ένα μήνυμα χρησιμοποιώντας **διαφορετικά ζεύγη κλειδιών**. Πιο συχνά, ειδικά στο οικοσύστημα του blockchain, χρησιμοποιούμε τον ορό κρυπτογράφηση με χρήση δημόσιου/ιδιωτικού κλειδιού (public/private key encryption) τον οποίο θεωρούμε ότι είναι συνώνυμος με την ασύμμετρη κρυπτογράφηση, αλλά υπάρχουν σημαντικές διακρίσεις μεταξύ των δύο, τις οποίες θα διερευνήσουμε σε αυτό το κεφάλαιο.

Θα εξετάσουμε δυο είδη ασύμμετρων αλγορίθμων και πώς βοηθούν στην προστασία των πιστωτικών καρτών, των ταυτοτήτων και γενικότερα των δεδομένων μας.

Σε αυτό το κεφάλαιο, θα αναφερθούν σύντομα τα ακόλουθα θέματα:

- Αλγόριθμος Diffie-Hellman και το σχετικό πρόβλημα man-in-the-middle
- Αλγόριθμος RSA
- **P**retty **G**ood **P**rivacy (PGP)

2.2.2.1 Αλγόριθμος D-H

Η ανταλλαγή ενός κλειδιού μεταξύ δύο μελών, καθώς και η διευκόλυνση ασφαλών ανταλλαγών πληροφοριών, είναι το πιο σημαντικό χαρακτηριστικό της κρυπτογράφησης ιδιωτικού/δημόσιου κλειδιού.

Αυτό το είδος κρυπτογραφίας είναι αρκετά σημαντικό στην καθημερινή μας ζωή. Αυτός είναι ο κλάδος της κρυπτογραφίας που είναι υπεύθυνος για τη διαφύλαξη των οικονομικών μας πληροφοριών, όπως πιστωτικές κάρτες και διαδικτυακούς λογαριασμούς. Τη δημιουργία των κωδικών πρόσβασης που χρησιμοποιούμε σε καθημερινή βάση και, γενικά, χρησιμοποιείται για την ασφαλής κοινή χρήση ευαίσθητων δεδομένων με τρίτους και τη προστασία του απορρήτου μας.

ΙΣΤΟΡΙΚΗ ΑΝΑΦΟΡΑ

Ο Diffie είναι ένας από τους πατέρες της σύγχρονης κρυπτογραφίας και το όνομά του θα παραμείνει για πάντα στην ιστορία της κρυπτογράφησης δημόσιου/ιδιωτικού κλειδιού. Ο Diffie

γεννήθηκε το 1944 και έλαβε το πτυχίο του από το MIT στη Βοστώνη το 1965. Εργάστηκε στον τομέα της κυβερνοασφάλειας μετά την αποφοίτησή του και έγινε ένας από τους πιο αξιότιμους ανεξάρτητους κρυπτογράφους της δεκαετίας του 1970.

Το θεμελιώδες πρόβλημα που έπρεπε να αντιμετωπιστεί ήταν ο τρόπος κρυπτογράφησης / αποκρυπτογράφησης ενός κοινόχρηστου εγγράφου χωρίς να μεταφερθούν άλλες πληροφορίες εκτός από το ίδιο το έγγραφο. Με λίγα λόγια, αυτό είναι ένα πρόβλημα ανταλλαγής κλειδιών.

Ο Diffie πήγε στο εργαστήριο Thomas Watson της IBM μια μέρα το 1974, όταν του ζητήθηκε να δώσει μια διάλεξη. Ανάφερε πολλούς τρόπους για την αντιμετώπιση του βασικού ζητήματος της διανομής κλειδιών, αλλά το κοινό ήταν καχύποπτο για την πρότασή του. Μόνο ένα άλλο άτομο συμμερίστηκε την άποψή του: ένας ανώτατος κρυπτογράφος της IBM που του ανέφερε ότι ένας καθηγητής από το Στάνφορντ είχε πρόσφατα επισκεφτεί το ίδιο εργαστήριο και είχε παρόμοια άποψη σχετικά με την πρόκληση. Ο Μάρτιν Χέλμαν ήταν ο εν λόγω καθηγητής.

Ο Diffie ήταν τόσο ενθουσιασμένος που ταξίδεψε σε από την μια άκρη της χώρας στην άλλην, στο Palo Alto της Καλιφόρνια, ώστε το ίδιο βράδυ για να δει τον καθηγητή Hellman. Η συνεργασία του Diffie και της Hellman θα μείνει στη μνήμη στην κρυπτογραφία για την ανάπτυξη ενός από τους πιο κομψούς αλγόριθμους στον τομέα: την **ανταλλαγή κλειδιών Diffie-Hellman**.

ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ

Για να κατανοήσουμε τον αλγόριθμο Diffie-Hellman (D-H), μπορούμε να βασιστούμε στα λεγόμενα *πειράματα σκέψης ή νοερή αναπαράσταση* μιας θεωρίας που χρησιμοποιούσε συχνά ο Αϊνστάιν.

Ας υποκριθούμε ότι έχουμε δύο ηθοποιούς, την Άλις και τον Μπομπ, που θέλουν να στείλουν ο ένας στον άλλο ένα μήνυμα (σε χαρτί), αλλά το κεντρικό ταχυδρομείο της πόλης επιθεωρεί όλες τις επιστολές. Ως αποτέλεσμα, η Άλις και ο Μπομπ πειραματίστηκαν με διάφορες τεχνικές για να παραδώσουν ένα γράμμα κρυφά αποφεύγοντας οποιαδήποτε εισβολή, όπως το κλείσιμο ενός κλειδιού σε ένα σιδερένιο κλουβί και η αποστολή του στον Μπομπ. Ωστόσο, ο Μπομπ δεν θα μπορούσε να ανοίξει το κουτί χωρίς το κλειδί, η Αλίκη και ο Μπομπ θα έπρεπε να συναντηθούν κάπου εκ των προτέρων ώστε η Αλίκη να παραδώσει το κλειδί στον Μπομπ. Επιστρέψαμε για άλλη μια φορά στο θέμα της ανταλλαγής κλειδιών.

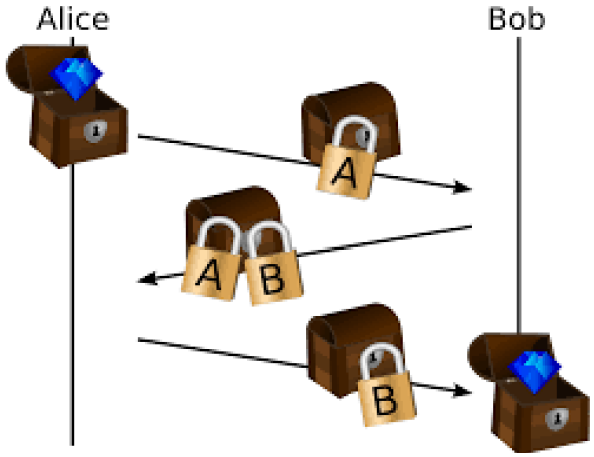
Για να δούμε τι πρέπει να κάνουν η Άλις και ο Μπομπ για μπορέσουν να ανταλλάξουν το κλειδί:

- **Βήμα 1ο**: Η Αλίκη βάζει το μυστικό της μήνυμα μέσα σε ένα μεταλλικό κουτί κλειστό με ένα σιδερένιο λουκέτο και το στέλνει στον Μπομπ, αλλά κρατά η ίδια το κλειδί. Τώρα, θυμηθείτε ότι η Αλίκη κλειδώνει το κουτί χρησιμοποιώντας το κλειδί της και δεν το δίνει στον Μπομπ.

- **Βήμα 2ο**: Ο Μπομπ εφαρμόζει μια ακόμη κλειδαριά στο κλουβί χρησιμοποιώντας το ιδιωτικό του κλειδί και στέλνει ξανά το κουτί στην Αλίκη. Έτσι, αφού ο Μπομπ έχει λάβει το κουτί για πρώτη φορά, δεν μπορεί να το ανοίξει. Απλώς προσθέτει μια ακόμη κλειδαριά στο κουτί

- **Βήμα 3ο**: Όταν η Alice λαμβάνει το κουτί για δεύτερη φορά, είναι ελεύθερη να αφαιρέσει το λουκέτο της, καθώς το κουτί παραμένει ασφαλισμένο με το κλειδί του Bob, όπως φαίνεται στο παρακάτω διάγραμμα, όταν η Alice ξαναστείλει το κουτί για τελευταία φορά. Να θυμάστε ότι το μήνυμα βρίσκεται πάντα μέσα στο πλαίσιο. Αυτήν τη στιγμή, το κουτί είναι κλειδωμένο μόνο με το λουκέτο του Μπομπ.

- **Βήμα 4ο**: Όταν ο Μπομπ λάβει το κουτί αυτή τη φορά, μπορεί να το ανοίξει, γιατί το κουτί παραμένει κλειδωμένο μόνο με το λουκέτο του. Τέλος, ο Μπομπ μπορεί να διαβάσει το περιεχόμενο του μηνύματος που έστειλε η Αλίκη που έχει διατηρηθεί μέσα στο κουτί.

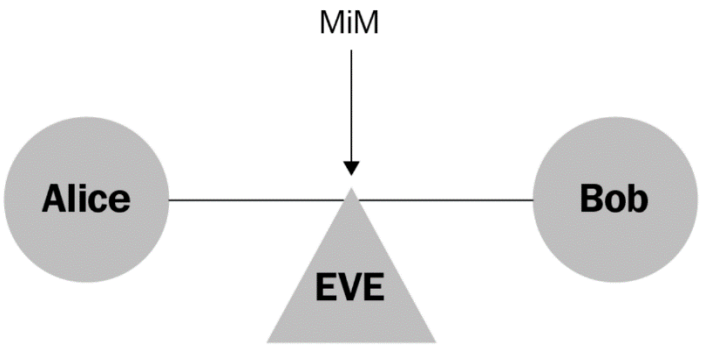


Εικόνα 2.2.2.1.1 *– Αλγόριθμος D-M χρησιμοποιώντας το παράδειγμα Alice & Bob. Διαθέσιμο στο βιβλίο [55], σελ. 96, Figure 3.1*

Ο D-H δεν είναι ένας εντελώς ασύμμετρος αλγόριθμος κρυπτογράφησης, αλλά μπορεί να ταξινομηθεί ως ένας αλγόριθμος δημόσιων/ιδιωτικών κλειδιών. Η διάκριση μεταξύ της ασύμμετρης και των δημόσιων/ιδιωτικών κλειδιών κρυπτογραφίας δεν είναι μόνο τυπική αλλά και πρακτική. Χρησιμοποιούμε τον αλγόριθμο D-H για να δημιουργήσουμε το κοινό μυστικό κλειδί, στη συνέχεια με τον AES ή άλλο συμμετρικό αλγόριθμο, κρυπτογραφούμε το μήνυμα. Ο D-H δεν κρυπτογραφεί απευθείας το μυστικό μήνυμα, μπορεί να καθορίσει μόνο ένα κοινό κλειδί μεταξύ δύο μερών.

ΑΔΥΝΑΜΙΕΣ ΑΛΓΟΡΙΘΜΟΥ

Η επίθεση man-in-the-middle (MiM) είναι η πιο διαδεδομένη επίθεση ενάντια στον αλγόριθμο D-H. Όταν ένας εισβολέας διεισδύει σε ένα κανάλι επικοινωνίας και κατασκοπεύει, μπλοκάρει ή τροποποιεί την επικοινωνία μεταξύ του αποστολέα και του παραλήπτη, αυτό είναι γνωστό ως επίθεση MiM. Η επίθεση συνήθως εκτελείται από την Εύα (η εισβολέας) που υποδύεται έναν από τους δύο γνήσιους παίκτες στη συνομιλία:



Εικόνα 2.2.2.1.2 *– Η Eve είναι ο ενδιάμεσος εισβολέας (Man In the Middle). Διαθέσιμο στο βιβλίο [55], σελ. 101, Figure 3.2*

Στο **3ο Βήμα** της μεθόδου D-H, η Alice και ο Bob αντάλλαξαν τις δημόσιες παραμέτρους τους, (A) και (B), αντίστοιχα (B). Η Αλίκη στέλνει το (A) στον Μπομπ και ο Μπομπ στέλνει το (B) στην Αλίκη σε αυτήν την περίπτωση.

Η Εύα (ο εισβολέας) παρεμβαίνει τώρα στον κανάλι επικοινωνίας υποδυόμενη την Αλίκη.

Για να δούμε πώς φαίνεται μια επίθεση MiM:

Βήμα 3^{ο}: Η Αλίκη στέλνει (Α) και ο Μπομπ στέλνει (Β) ο ένας στον άλλο.

Εδώ έχουμε την επίθεση MiM: η Εύα στέλνει (Ε) στον Μπομπ και ο Μπομπ, υποθέτοντας ότι είναι η Αλίκη, στέλνει (Β) στην Εύα.

Με αυτόν τον τρόπο η Εύα απλώς πρέπει να περιμένει η Άλις να αφαιρέσει το λουκέτο της (Α) και μετά έχει την ικανότητα να δει πρώτα αυτή το μυστικό μήνυμα και εάν επιθυμεί να το αλλοιώσει πριν το στείλει στον Μπομπ.

Μια άλλη πιθανή επίθεση, κοινώς γνωστή ως **επίθεση γενεθλίων (birthday attack),** είναι μια από τις πιο γνωστές επιθέσεις σε διακριτούς λογάριθμους. Έχει διαπιστωθεί ότι τουλάχιστον δύο μέλη σε μια ομάδα θα μοιράζονται μια ημερομηνία γέννησης, επιτρέποντας σε μια κυκλική ομάδα να αναγνωρίσει κάποιες ισοδύναμες τιμές (συγκρούσεις) προκειμένου να λύσει έναν διακριτό λογάριθμο.

2.2.2.2 Αλγόριθμος RSA

Ο RSA λάμπει σαν αστέρι μεταξύ των κρυπτογραφικών αλγορίθμων. Η ομορφιά του είναι συγκρίσιμη με τη απλή λογική του και μέσα του κρύβεται μια τέτοια δύναμη που εξακολουθεί να χρησιμοποιείται για να προστατεύει πάνω από το 80% των επιχειρηματικών συναλλαγών στον κόσμο μετά από τα 40 χρόνια χρονιά δημιουργίας του.

ΙΣΤΟΡΙΚΗ ΑΝΑΦΟΡΑ

Rivest, **S**hamir και **A**ldemar είναι τα ονόματα των τριών καινοτόμων που δημιούργησαν τον αλγόριθμο **RSA**. Ο RSA θα μπορούσε να θεωρηθεί ο τέλειος αλγόριθμος ασύμμετρης κρυπτογραφίας. Στην πραγματικότητα, το CESG, ένας αγγλικός οργανισμός κρυπτογραφίας, απέδωσε στον Τζέιμς Άλις την επινόηση της κρυπτογράφησης με δημόσιο κλειδί το 1970 και η ίδια υπηρεσία ισχυρίστηκε ότι ο Κλίφορντ Κοκς έγραψε ένα έγγραφο το 1973 που παρουσίαζε μια παρόμοια έκδοση της μεθόδου RSA.

Αυτός ο αλγόριθμος δημιουργήθηκε από τρία άτομα, όπως αναφέρθηκε προηγουμένως. Εκείνη την εποχή, ήταν όλοι ερευνητές στο MIT στη Βοστώνη. Μιλάμε για τα τέλη της δεκαετίας του 1970. Ο Ronald Rivest ενθουσιάστηκε από αυτό το νέο είδος κρυπτογραφίας με την ανακάλυψη του αλγορίθμου D-H. Ξεκίνησε ζητώντας τη βοήθεια ενός μαθηματικού, του Leonard Adleman, και αργότερα ενός συναδέλφου του από το τμήμα Computer Science, Aid Shamir. Ο Rivest προσπαθούσε να επινοήσει μια μαθηματική μέθοδο για την αποστολή ενός μυστικού μηνύματος κρυπτογραφημένου με ένα δημόσιο κλειδί και την αποκρυπτογράφηση του χρησιμοποιώντας το ιδιωτικό κλειδί του δέκτη.

Στο D-H, ωστόσο, το μήνυμα μπορεί να κρυπτογραφηθεί μόνο αφού το κλειδί έχει μεταφερθεί και έχει χρησιμοποιηθεί το ίδιο κοινόχρηστο κλειδί. Το πρόβλημα εδώ ήταν να καταλάβουμε πώς να παραδώσουμε ένα μήνυμα που είχε κρυπτογραφηθεί χρησιμοποιώντας ένα δημόσιο κλειδί και είχε αποκρυπτογραφηθεί χρησιμοποιώντας ένα ιδιωτικό κλειδί. Αλλά, απαιτούσε μια εξαιρετικά συγκεκριμένη αντίστροφη μαθηματική συνάρτηση. Αυτή είναι η πραγματική προστιθέμενη αξία της καινοτομίας RSA, την οποία θα δούμε σε λίγο. Η βασική ιδέα αυτού του ασύμμετρου αλγορίθμου είναι ότι τα κλειδιά κρυπτογράφησης και αποκρυπτογράφησης είναι διακριτά.

ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ

Σκεφτείτε ότι η Αλίκη και ο Μπομπ ανταλλάσσουν ένα μυστικό μήνυμα για να κατανοήσετε καλύτερα αυτήν τη μέθοδο.

Ας φανταστούμε ότι ο Μπομπ θέλει να στείλει στην Αλίκη ένα μυστικό μήνυμα και έχει τις ακόλουθες πληροφορίες:

• **M**: Ένα κωδικοποιημένο μήνυμα

• **e**: Μια παράμετρος που είναι ανοιχτή στο κοινό (συνήθως ένας σταθερός αριθμός)

• **c**: Το κρυπτόγραμμα ή κρυπτοκείμενο

• **p**, **q**: Δύο τεράστιοι πρώτοι αριθμοί στην τύχη (τα ιδιωτικά κλειδιά της Alice)

Ακολουθούν τα βήματα για τη δημιουργία δημόσιων και ιδιωτικών κλειδιών. Όπως θα δείτε, η κυριά λειτουργία του RSA είναι να δημιουργήσει το **ιδιωτικό κλειδί της Alice**, **[d]**.

Δημιουργία κλειδιών:

Το **δημόσιο κλειδί της Alice**, **(N)**, δημιουργείται χρησιμοποιώντας τον ακόλουθο κώδικα:

$$p * q = N$$

Ο πολλαπλασιασμός δύο πολύ μεγάλων πρώτων ακεραίων καθιστά πολύ δύσκολη την παραγοντοποίηση του (N), καθώς και εξαιρετικά δύσκολο για έναν εισβολέα να εντοπίσει τα [p] και [q]. Ο παρακάτω κωδικός αντιπροσωπεύει το **ιδιωτικό κλειδί της Alice**, **[d]**:

$$[d] * e = 1 \ (\text{mod}[p-1] * [q-1])$$

Ο Μπομπ είναι αυτός που κάνει την κρυπτογράφηση:

$$c = M \ ^\wedge \ e \ (\text{mod } N)$$

Ο Μπομπ δίνει στην Αλίκη το κρυπτογραφημένο κείμενο (γ). Χρησιμοποιώντας το ιδιωτικό της κλειδί, μπορεί τώρα να το αποκρυπτογραφήσει (c):

$$c \ ^\wedge \ d = M \ (\text{mod } N)$$

ΑΔΥΝΑΜΙΕΣ ΑΛΓΟΡΙΘΜΟΥ

Οι τρεις πιο πιθανές μέθοδοι επίθεσης στο RSA σχετίζονται με τη δημόσια παράμετρο (mod N) ένας εισβολέας θα μπορούσε χρησιμοποιήσει τις ακόλουθες μεθόδους για να επιτεθεί στο N = p*q:

1) Να χρησιμοποιήσει έναν αλγόριθμο παραγοντοποίησης για να βρει τα p και q, σε ένα εύλογο χρονικό διάστημα.
2) Να χρησιμοποιήσει αλγόριθμους που μπορούν να εντοπίσουν αυτούς τους αριθμούς υπό συγκεκριμένες καταστάσεις.
3) Να παραγοντοποιήσει το N χρησιμοποιώντας κβαντικό υπολογιστή (στο μέλλον).

Ας εξετάστε ένα-ένα αυτά τα τρία σενάρια:

Στο **πρώτο σενάριο**, μια αποτελεσματική μέθοδος παραγοντοποίησης δεν είναι ακόμα γνωστή. Οι παρακάτω είναι οι πιο διαδεδομένες μέθοδοι:

I. Ο αλγόριθμος για ένα κόσκινο πεδίου καθολικού αριθμού (general number field sieve algorithm)
II. Ο αλγόριθμος τετραγωνικού κόσκινου (quadratic sieve algorithm)
III. Ο αλγόριθμος Pollard

Στο **δεύτερο σενάριο**, εάν (n) είναι ο αριθμός των ψηφίων σε N = p*q και ο εισβολέας γνωρίζει τα πρώτα (n/4) ή τα τελικά (n/4) ψηφία των [p] ή [q], τότε θα μπορούσε αποτελεσματικά να παραγοντοποιήσει τον (N). Είναι εφικτό να παραγοντοποιήσουμε το N εάν τα [p] και [q] περιέχουν και τα δύο 100 ψηφία και τα πρώτα (ή τελικά) 50 ψηφία του [p] είναι γνωστά.

Στο **τρίτο σενάριο**, στην περίπτωση που ο εισβολέας διαθέτει κβαντικό υπολογιστή, ο αλγόριθμος του Shor ενδέχεται να παραγοντοποιήσει το N σε σύντομο χρονικό διάστημα, αλλά στο μέλλον θα εμφανιστούν πιο αποτελεσματικοί κβαντικοί αλγόριθμοι κρυπτογράφησης.

Τέλος, ο RSA μπορεί να σπάσει αν έχουμε ένα πολύ μικρό κομμάτι απλού κειμένου. Αυτό οφείλεται στο γεγονός ότι η λειτουργία ισχύος, M^e, παραμένει μέσα στο modulo N. Για να το αντιμετωπίσουμε, μπορούμε να παρατείνουμε το μήνυμα προσθέτοντας τυχαία bits σε αυτό. Αυτή η τεχνική ονομάζεται padding και είναι μια τεχνική που χρησιμοποιείται συχνά στην κρυπτογραφία και την ασφάλεια στον κυβερνοχώρο.

2.2.2.3 Αλγόριθμος PGP

Το Pretty Good Privacy (PGP) είναι ίσως το πιο χρησιμοποιούμενο κρυπτογραφικό λογισμικό στον κόσμο.

ΙΣΤΟΡΙΚΗ ΑΝΑΦΟΡΑ

Κατά τη διάρκεια του Ψυχρού Πολέμου, ο Philip Zimmermann εφάρμοσε το PGP. Ο Φίλιπ άρχισε να προετοιμάζεται για ένα ταξίδι στη Νέα Ζηλανδία με την οικογένειά του επειδή

ένιωθε ότι, σε περίπτωση πυρηνικού χτυπήματος, η απομόνωση της χώρας από τον υπόλοιπο κόσμο θα έκανε την πυρηνική καταστροφή λιγότερο καταστροφική. Στο τελος, κάτι άλλαξε την γνώμη του σχετικά με τη μετακόμισή του στη Νέα Ζηλανδία και επέλεξε να μείνει στις Ηνωμένες Πολιτείες.

Ο Zimmermann, ένας ακτιβιστής κατά των πυρηνικών, δημιούργησε το PGP για να κρυπτογραφήσει τις επικοινωνίες και τα δεδομένα που αποστέλλονται μέσω του Διαδικτύου προκειμένου να συνδεθεί με τους συμμάχους του. Έκανε το πρόγραμμα ανοιχτού κώδικα και το έκανε διαθέσιμο για μη εμπορική χρήση.

Τα κρυπτοσυστήματα με περισσότερα από 40 bits θεωρούνταν ότι αποτελούσαν απειλή την εποχή. Δηλαδή, η κρυπτογραφία εξακολουθούσε να θεωρείται ως στρατιωτικό όπλο. Εάν θέλετε να κατοχυρώσετε με δίπλωμα ευρεσιτεχνίας ένα νέο κρυπτοσύστημα, πρέπει πρώτα να λάβετε άδεια από το Υπουργείο Άμυνας για να το δημοσιοποιήσετε.

Το PGP έπρεπε να αντιμετωπίσει αυτό το ζήτημα, καθώς ποτέ δεν χρησιμοποίησε κλειδιά μικρότερα από 128 bit. Για οποίον παραβιάσει αυτή την νομική απαίτηση, οι ποινικές κατηγορίες είναι αυστηρές, γι' αυτό το νομικό καθεστώς του Zimmermann παρέμεινε άλυτο για πολλά χρόνια έως ότου η αμερικανική κυβέρνηση επέλεξε να ολοκληρώσει την έρευνα και να τον αθωώσει.

ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ

Το PGP είναι περισσότερο **πρωτόκολλο παρά αλγόριθμος**. Το νέο στοιχείο εδώ είναι ότι **συνδυάζονται ασύμμετρη και συμμετρική κρυπτογράφηση**. Το κλειδί ανταλλάσσεται χρησιμοποιώντας μια τεχνική ασύμμετρης κρυπτογράφησης, το μήνυμα κρυπτογραφείται και υστέρα το κρυπτογραφημένο κείμενο λαμβάνεται χρησιμοποιώντας συμμετρική κρυπτογράφηση. Επιπλέον, για την αναγνώριση του χρήστη και την αποφυγή επίθεσης MiM, **απαιτείται μια ψηφιακή υπογραφή**.

Ακολουθούν τα βήματα του πρωτόκολλου:

- **Βήμα 1ο**: Το κλειδί μεταδίδεται χρησιμοποιώντας έναν αλγόριθμο ασύμμετρης κρυπτογράφησης (ElGamal, RSA)

- **Βήμα 2ο**: Το κλειδί που έχει μεταδοθεί με ασύμμετρη κρυπτογράφηση γίνεται το κλειδί συνεδρίας για τη συμμετρική κρυπτογράφηση (DES και AES – *βλ. Κεφ. 2.2.1*)

- **Βήμα 3ο**: Μια ψηφιακή υπογραφή χρησιμοποιείται για την αναγνώριση των χρηστών (*βλ. Κεφ. 2.3.1.1*)

- **Βήμα 4ο**: Η αποκρυπτογράφηση πραγματοποιείται χρησιμοποιώντας το συμμετρικό κλειδί

Το PGP είναι ένα καλό πρωτόκολλο για πολύ καλό απόρρητο και για διασφάλιση της μετάδοσης εμπορικών μυστικών.

2.2.2.4 Αλγόριθμος ElGamal

Ο ElGamal προσπαθεί να λύσει το πρόβλημα του MiM, καθώς και τη ανυπαρξία υπογραφών που δηλώνουν την ιδιοκτησίας των κλειδιών στον D-H. Ο ElGamal (όπως ο RSA) είναι επίσης ένας αυθεντικός ασύμμετρος αλγόριθμος, διότι κρυπτογραφεί το μήνυμα χωρίς να μοιράζεται το κλειδί εκ των προτέρων.

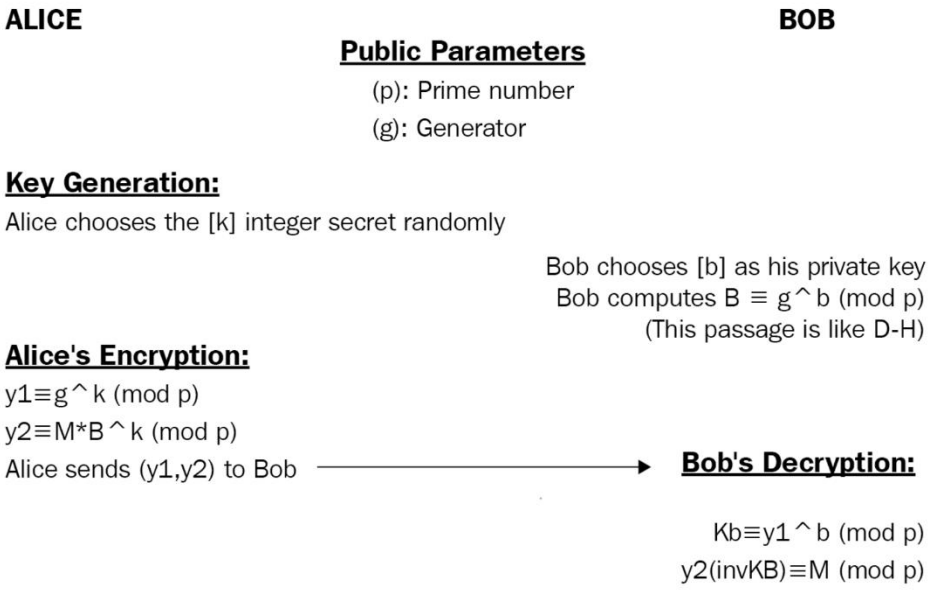
ΣΥΝΤΟΜΗ ΠΕΡΙΓΡΑΦΗ

Η δυσκολία εδώ συνήθως σχετίζεται με την επίλυση του διακριτού λογάριθμου. Υπάρχει επίσης μια δυσκολία με την παραγοντοποίηση, όπως θα ανακαλύψουμε αργότερα.

Ο ElGamal είναι ο πρώτος αλγόριθμος που εισάγει ένα νέο στοιχείο: έναν μυστικό ακέραιο τυχαίο αριθμό, [k], που επιλέγεται από τον αποστολέα. Είναι μια σημαντική καινοτομία, καθώς κάνει τη λειτουργία κρυπτογράφησης «εφήμερη», με την έννοια ότι την καθιστά απρόβλεπτη. Επιπλέον, αυτό το νέο στοιχείο θα συσχετίζεται συνήθως με το πρωτόκολλο μηδενικής γνώσης (Zero Knowledge Protocol).

Ας δούμε πώς εφαρμόζεται αυτή η τεχνική και πώς χρησιμοποιείται για την αποστολή του μυστικού μηνύματος [M].

Οι δύο ηθοποιοί είναι όπως πάντα, η Άλις και ο Μπομπ. Ο αποστολέας είναι η Άλις και ο παραλήπτης είναι ο Μπομπ. Η διαδικασία του αλγόριθμου ElGamal φαίνεται στη παρακάτω απεικόνιση.



Εικόνα 2.2.2.4.1 *– Κρυπτογράφηση / Αποκρυπτογράφηση χρησιμοποιώντας τον αλγόριθμο ElGamal. Διαθέσιμο στο βιβλίο [55], σελ. 115, Figure 3.5*

Μπορούμε να παρατηρήσουμε έναν αντίστροφο πολλαπλασιασμό στην τελική φάση της αποκρυπτογράφησης του Bob (mod p). Σε ένα πεπερασμένο πεδίο, αυτή η πράξη είναι ουσιαστικά μια διαίρεση. Άρα, αν το B είναι το αντίστροφο του A, παίρνουμε A*B = 1. (mod p).

2.2.3 Συναρτήσεις Κατακερματισμού & Ψηφιακές Υπογραφές (Hash Functions & Digital Signatures)

2.2.3.1 Εισαγωγή

Τα περισσότερα σύμβολα, που αναφέρονται σε κάθε είδους συμφωνία μεταξύ ατόμων ή οργανισμών, έχουν γραφτεί σε χαρτί και έχουν υπογραφεί χειροκίνητα από την αυγή του χρόνου, με συγκεκριμένη υπογραφή στο τέλος του εγγράφου για την επικύρωση του. Αυτό ήταν εφικτό δεδομένου ότι οι υπογράφοντες ήταν φυσικά παρόντες τη στιγμή της υπογραφής. Οι υπογράφοντες μπορούσαν γενικά να εμπιστεύονται ο ένας τον άλλον, καθώς η ταυτότητά τους ως υπερκομματική οντότητα πιστοποιήθηκε από τρίτο έμπιστο πρόσωπο (συμβολαιογράφο ή νομικό πρόσωπο).

Τα άτομα που επιθυμούν να υπογράψουν συμβόλαια στις μέρες μας συνήθως δεν γνωρίζουν ο ένας τον άλλον και κοινοποιούν έγγραφα που πρέπει να υπογραφούν μέσω email, υπογράφοντας τα χωρίς το όφελος ενός έμπιστου τρίτου μέρους για την επαλήθευση της ταυτότητάς τους.

2.2.3.2 Συναρτήσεις Κατακερματισμού (Hash Functions)

Στην κρυπτογραφία, οι συναρτήσεις κατακερματισμού χρησιμοποιούνται συχνά σε μια μεγάλη ποικιλία εφαρμογών.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Μια συνάρτηση μονής κατεύθυνσης αναφέρεται ως συνάρτηση κατακερματισμού (ή απλώς κατακερματισμός (hash). Θα δούμε αργότερα πώς αυτό το μονόδρομο χαρακτηριστικό είναι κρίσιμο για την κατηγοριοποίηση των συναρτήσεων κατακερματισμού.

Το να είναι μια συνάρτηση **μονόδρομη** σημαίνει ότι ο υπολογισμός του αποτελέσματος προς μία κατεύθυνση είναι απλός, αλλά η επιστροφή στο αρχικό μήνυμα από την έξοδο της συνάρτησης είναι πολύ δύσκολη (αν όχι αδύνατη).

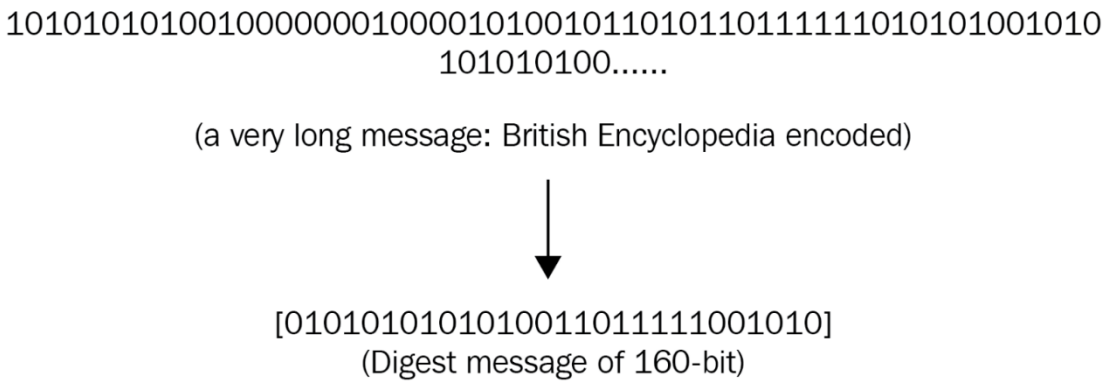
Ας δούμε τις ιδιότητες που ικανοποιεί μια συνάρτηση κατακερματισμού:

1. Το h(M) για ένα μήνυμα εισόδου [M] μπορεί να υπολογιστεί γρήγορα.

2. Η επιστροφή από την έξοδο (M') που υπολογίζεται χρησιμοποιώντας το h(M) στο αρχικό μήνυμα [M] θα πρέπει να είναι σχεδόν αδύνατη.

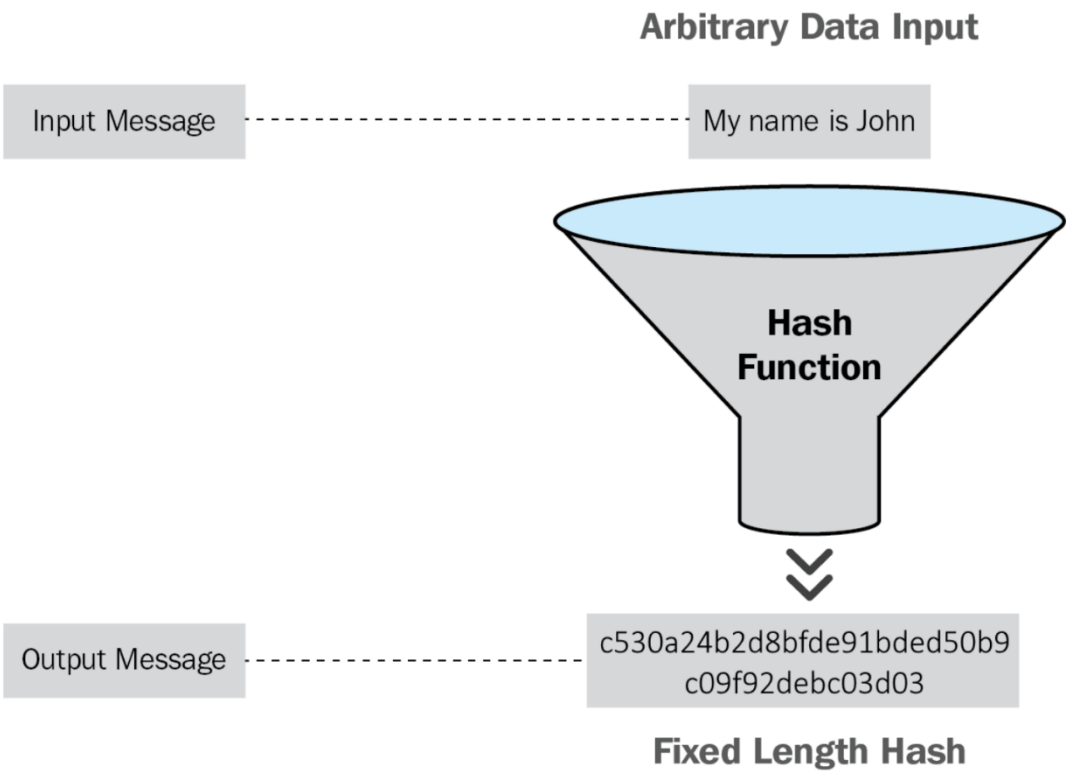
3. Εύρεση δύο εναλλακτικών μηνυμάτων εισαγωγής [m1] και [m2], έτσι ώστε: h(m1) = h(m2), να είναι υπολογιστικά δυσεπίλυτο.

Για την παροχή ενός παραδείγματος συνάρτησης κατακερματισμού, ας χρησιμοποιήσουμε όλο το περιεχόμενο της Wikipedia και ας το μετατρέπουμε σε ακολουθία bit σταθερού μήκους ως εξής:



Εικόνα 2.2.3.2.1 *– Παράδειγμα μιας εξόδου κατακερματισμού (hash digest). Διαθέσιμο στο βιβλίο [55], σελ. 122, Figure 4.1*

Ένας μηχανισμός κοπής διοχέτευσης, όπως αυτός που ακολουθεί, ο οποίος αφομοιώνει το απλό κείμενο ως είσοδο και επιστρέφει έναν κατακερματισμό σταθερού μήκους, είναι μια άλλη καλή μεταφορά για τις συναρτήσεις κατακερματισμού:



Εικόνα 2.2.3.2.1 *Παρομοίωση της συνάρτησης κατακερματισμού με ένα μηχανισμό κοπής διοχέτευσης. Διαθέσιμο στο βιβλίο [55], σελ. 123, Figure 4.2*

Όπως είπαμε στην αρχή, οι συναρτήσεις κατακερματισμού χρησιμοποιούνται συνήθως στην κρυπτογραφία για διάφορους σκοπούς:

1. Κατά τη συλλογή ψηφιακών υπογραφών σε ασύμμετρη κρυπτογράφηση, συνήθως χρησιμοποιούνται συναρτήσεις κατακερματισμού **για να αποτραπεί η αποκάλυψη του αρχικού μηνύματος** [M]. Επίσης, χρησιμοποιώντας το (M') ως διακομιστή μεσολάβησης, ο κατακερματισμός του αρχικού μηνύματος καθορίζει την ταυτότητα του πομπού.

2. Οι κατακερματισμοί χρησιμοποιούνται για την **επαλήθευση της ακεραιότητας ενός μηνύματος**. Σε αυτήν την περίπτωση, ο δέκτης μπορεί εύκολα να προσδιορίσει εάν το αρχικό μήνυμα [M] έχει ενημερωθεί με βάση τον ψηφιακό κατακερματισμό του αρχικού μηνύματος (M'). Πράγματι, αλλάζοντας μόνο ένα bit σε ολόκληρο το περιεχόμενο της Βρετανικής Εγκυκλοπαίδειας [M], η συνάρτηση κατακερματισμού της, h(M), θα έχει μια εντελώς διαφορετική τιμή κατακερματισμού h(M') από πριν. Αυτή η ιδιαιτερότητα των συναρτήσεων κατακερματισμού είναι κρίσιμη γιατί θέλουμε μια ισχυρή λειτουργία για να **διασφαλίσουμε ότι το αρχικό υλικό δεν έχει αλλοιωθεί**.

3. Οι συναρτήσεις κατακερματισμού χρησιμοποιούνται επίσης στην ευρετηρίαση βάσεων δεδομένων (database indexing).

4. Η βασική αρχή της δημιουργίας μιας ασφαλούς συνάρτησης κατακερματισμού είναι η λήψη του αρχικού μηνύματος από μια έξοδο h(M) να είναι σχεδόν αδύνατη.

Κυριοι Αλγοριθμοι Καταμερισμου

Επειδή ένας αλγόριθμος κατακερματισμού είναι ένα είδος μαθηματικής συνάρτησης που δημιουργεί μια σταθερή έξοδο bit από μια μεταβλητή είσοδού, θα πρέπει να είναι *collision-free*, πράγμα που σημαίνει ότι η παραγωγή δύο συναρτήσεων κατακερματισμού για την ίδια τιμή εισόδου θα πρέπει να είναι αδύνατη και το αντίστροφο.

MD5 (Message Digest 5)

Ο Rives δημιούργησε το MD5, μια εξέχουσα συνάρτηση κατακερματισμού της οικογένειας MD, το 1991. Η αρχιτεκτονική Merkle–Damgard χρησιμοποιείται σε αυτήν τη συνάρτηση κατακερματισμού. Ο αλγόριθμος MD5 δημιουργεί ένα μήκος 128 bit από ένα μήνυμα οποιουδήποτε μήκους ως είσοδο. Ωστόσο, το MD5 έχει γίνει στόχος διαφόρων επιθέσεων. Ο Bore και ο Bosselaers ανακάλυψαν επιθέσεις σύγκρουσης που στόχευαν τη λειτουργία συμπίεσης το 1992. Ο Dobbertin παρουσίασε τις πληροφορίες ότι ο MD5 ήταν αντικείμενο επιθέσεων σύγκρουσης το 1996. Στο [57] αποκαλύφθηκαν επίσης επιτυχημένες επιθέσεις σύγκρουσης κατά του MD5. Προηγούμενη έρευνα [58], [59] έδειξε ότι οι επιθέσεις σύγκρουσης στο MD5 μπορεί να βελτιωθούν.

RIPEMD-160

Είναι μια πολύ γνωστή συνάρτηση κατακερματισμού RIPEMD που δημιουργήθηκε από τους Dobbertin, Bosselaers και Perneel το 1996. Ορίζεται στο πρότυπο ISO/IEC10118-3:2004 για εξειδικευμένες συναρτήσεις κατακερματισμού. Χρησιμοποιεί επίσης το πλαίσιο Merkle-Damgrd. Δημιουργεί μια σύνοψη μηνυμάτων που έχει μήκος 160 bit [60]. [59] ανακαλύφθηκε ότι είναι ευάλωτος σε επιθέσεις σύγκρουσης ημιελεύθερης εκκίνησης ο RIPEMD-160.

Η ΟΙΚΟΓΕΝΕΙΑ – SHA (SECURE HASH ALGORITHM)

Η οικογένεια SHA είναι η πιο δημοφιλής, η οποία δημιουργήθηκε από την Εθνική Υπηρεσία Ασφαλείας (NSA). Ο SHA-256 είναι ο αλγόριθμος κατακερματισμού που χρησιμοποιείται επί του παρόντος στο Proof of Work (PoW) του Bitcoin κατά την εξόρυξη bitcoin. Η μέθοδος παραγωγής νέων Bitcoin είναι γνωστή ως εξόρυξη και περιλαμβάνει την επίλυση ενός εξαιρετικά δύσκολου μαθηματικού προβλήματος που βασίζεται στο SHA-256.

SHA-1

Ο ασφαλής αλγόριθμος κατακερματισμού 1 (SHA-1) δημιουργήθηκε από το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST) το 1995. Χρησιμοποιεί την ίδια δομή Merkle–Damgard με το MD5 και δημιουργεί μια σύνοψη μηνύματος 160-bit για οποιοδήποτε μήκος μηνύματος εισόδου. Προηγούμενη έρευνα [61]-[62] έδειξε ότι η επίθεση σύγκρουσης μπορεί να χρησιμοποιηθεί κατά του SHA-1. Ως αποτέλεσμα, το NIST δήλωσε ότι το SHA-1 θα καταργηθεί σταδιακά [63].

SHA-2

Το NIST (2002) εισήγαγε τρεις νέους αλγόριθμους στην οικογένεια SHA, τους SHA-256, SHA-384 και SHA512, με μήκη κωδικού κατακερματισμού 256, 348 και 512 bit, αντίστοιχα. Αυτά έχουν την ίδια δομή με τα MD5 και SHA-1, αλλά είναι πιο περίπλοκα αφού ο αλγόριθμος συμπίεσης

περιλαμβάνει μια μη γραμμική συνάρτηση. Ωστόσο, το SHA-2 δεν ευνοείται για τη διασφάλιση της ακεραιότητας, καθώς είναι πιο αργό από το SHA-1 [64]. Το Bitcoin, από την άλλη πλευρά, ως το πιο δημοφιλές κρυπτονόμισμα, χρησιμοποιεί το SHA256 για το Hashcash, το οποίο διασφαλίζει ότι οι συναλλαγές μεταξύ ομότιμων στο δίκτυο Bitcoin είναι ασφαλείς. Ωστόσο, δεδομένου ότι το SHA-256 δεν υποστηρίζει multithreading, δεν είναι αρκετά γρήγορο για συναλλαγές [65]. Προηγούμενες δημοσιεύσεις [66] έχουν δείξει τις πιο σύγχρονες επιθέσεις SHA-2.

SHA-3

Μετά από πολλές επιτυχημένες επιθέσεις σύγκρουσης με ολοένα και μικρότερη πολυπλοκότητα (όπως MD5, SHA-1 και SHA-2), η NIST ξεκίνησε έναν δημόσιο διαγωνισμό για να σχεδιάσει τον SHA-3, έναν εντελώς νέο αλγόριθμο κατακερματισμού, στο Federal Register. Η ανακοίνωση της πρωτοβουλίας δημοσιοποιήθηκε το 2007. Ο νικητής του διαγωνισμού, ο Keccak, αποκαλύφθηκε τέσσερα χρόνια αργότερα, στις 2 Οκτωβρίου 2012. Η NIST όρισε τον SHA-3 ως τυπικό αλγόριθμο κατακερματισμού το 2014. Αυτή η προσέγγιση, ωστόσο, είναι ευάλωτη σε τεχνικές εύρεσης σύγκρουσης [67], [68]. Ωστόσο, σε σύγκριση με άλλες συναρτήσεις κατακερματισμού, η προσέγγιση έχει κακή απόδοση λογισμικού [69]. Επίσης είναι ο αλγόριθμος καταμερισμού που χρησιμοποιεί το Ethereum Blockchain.

Properties	**Name of Algorithm**				
	MD5	**RIPEMD -160**	**SHA-1**	**SHA-2 256/512**	**SHA-3 256/512**
Block Size	512 bits	512 bits	512 bits	512/1024 bits	1088/576 bits
Word Size	32 bits	32bits	32bits	32/64 bits	320/320bits
Output Size	128bits	160 bits	160 bits	256/512 bits	1600/1600bits
Rounds	18	80	80	64/80	24/24
Operations	ADD,XOR, AND,OR, NOT, SHIFT	ADD,, ROTATE, XOR,AND, OR,NOT	ADD, XOR AND, OR,NOT, ROTATE.	ADD, XOR, OR, AND SHIFT,, ROTATE	-
Construction	Merkle-Damgard	Merkle-Damgard	Merkle-Damgard	Merkle-Damgard	Sponge

Πίνακας 2.2.3.2.1 *– Σύγκριση μεταξύ των αλγορίθμων κατακερματισμού. Διαθέσιμο στην ανασκόπηση [56], σελ. 3, Table I.*

2.2.3.3 Ψηφιακές Υπογραφές (Digital Signatures)

Το ζήτημα του ελέγχου ταυτότητας είναι ένα από τα πιο συναρπαστικά και πιο δύσκολα προβλήματα στην κρυπτογραφία. Ο έλεγχος ταυτότητας είναι μία από τις πιο ευαίσθητες (και συχνά χρησιμοποιούμενες) πτυχές της μεθόδου ελέγχου πρόσβασης.

Ο έλεγχος ταυτότητας βασίζεται σε τρεις μεθόδους[55]:

1. Σε ένα θέμα που γνωρίζει μόνο ο χρήστης (για παράδειγμα, ένας κωδικός πρόσβασης)

2. Σε οτιδήποτε ανήκει αποκλειστικά στον χρήστη (έξυπνη κάρτα, συσκευή ή διακριτικό)

3. Σε οτιδήποτε προσδιορίζει τον χρήστη (για παράδειγμα, δακτυλικά αποτυπώματα, σαρώσεις ίριδας και άλλα βιομετρικά χαρακτηριστικά ενός ατόμου)

Εκτός από αυτές τις τρεις προσεγγίσεις, υπάρχει μία ακόμη που περιλαμβάνει κάτι που έχει μόνο ο χρήστης και έχει να κάνει με κάτι που τις ορίζει: τα εγκεφαλικά κύματα. Όταν κοιτάζετε ή σκέφτεστε μια εικόνα, για παράδειγμα, τα εγκεφαλικά κύματα που παράγονται από τον εγκέφαλό σας διαφέρουν από οποιαδήποτε άλλα εγκεφαλικά κύματα που περιγράφουν ένα άλλο άτομο.

ΟΡΙΣΜΟΣ

Μια ψηφιακή υπογραφή επαληθεύει ότι ο αποστολέας του μηνύματος [M] έχει δώσει εντολή στον παραλήπτη να εκτελέσει τα ακόλουθα[55]:

- Πρέπει να εδραιώσουν την ταυτότητά τους.
- Βεβαιωθείτε ότι το μήνυμα [M] δεν έχει παραβιαστεί.
- Φροντίστε για τη μη απόρριψη του μηνύματος

2.3 Σύγχρονη Κρυπτογραφία

2.3.1 Πρωτοκολλά Μηδενικής-Γνώσης (Zero-Knowledge Protocols)

Το ζήτημα του ελέγχου ταυτότητας, όπως είδαμε στο Κεφ. 2.2.3, είναι ένα από τα πιο κρίσιμα, πολύπλοκα και συναρπαστικά προβλήματα που θα πρέπει να αντιμετωπίσει η κρυπτογραφία στο εγγύς μέλλον. Σκεφτείτε την κατάσταση όταν πρέπει να ταυτιστείτε με κάποιον που δεν γνωρίζετε στο διαδίκτυο. Θα σας ζητηθεί να υποβάλετε πρώτα το όνομα, το επώνυμο και τη διεύθυνσή σας, ακολουθούμενα από τον αριθμό κοινωνικής ασφάλισης και άλλες προσωπικές πληροφορίες.

Φυσικά, γνωρίζετε ότι η αποκάλυψη τέτοιων πληροφοριών μέσω του Διαδικτύου μπορεί να είναι αρκετά επικίνδυνη, καθώς κάποιος μπορεί να πάρει τα προσωπικά σας στοιχεία και να τα χρησιμοποιήσει για κακούς λόγους.

Με το να μην προβάλουμε οποιασδήποτε ευαίσθητης πληροφορίας είναι μια μέθοδος για την αντιμετώπιση αυτού του τύπου δυσκολιών, αλλά αυτό δεν είναι πάντα εφικτό. Μια άλλη επιλογή είναι η παροχή αποδεικτικών στοιχείων γνώσης για την αποτροπή της αποκάλυψης ευαίσθητων πληροφοριών. Πρωτόκολλα μηδενικής γνώσης είναι το όνομα που δίνεται σε αυτά τα κρυπτογραφικά πρωτόκολλα (ZKP).

Θα γίνει μια σύντομη αναφορά στα ακόλουθα είδη πρωτοκόλλων μηδενικής γνώσης:

- Non-Interactive ZKPs
- Schnorr's interactive ZKP
- zk-SNARKs

2.3.1.1 Non-interactive ZKPs

Το πρωτόκολλο Zero-Knowledge Knowledge (γνωστό και ως Zero-Knowledge Knowledge Password Proof, ή ZKP) είναι μια μέθοδος ελέγχου ταυτότητας στην οποία δεν κοινοποιούνται κωδικοί πρόσβασης, καθιστώντας αδύνατη την κλοπή τους.

Το ZKP σάς δίνει τη δυνατότητα να αποδείξετε στο άλλο «άκρο» της επικοινωνίας ότι γνωρίζετε κάποια (ή πολλά) μυστικά χωρίς να τα εκθέσετε πραγματικά. Η φράση "μηδενική γνώση" προέρχεται από το γεγονός ότι δεν παρέχονται ("μηδέν") πληροφορίες σχετικά με το μυστικό, ωστόσο το δεύτερο μέρος (που αναφέρεται ως "Επαληθευτής") πείθεται (δικαίως) ότι το πρώτο μέρος (αναφέρεται ως " Prover") γνωρίζει το εν λόγω μυστικό.

Στο μη διαδραστικό πρωτόκολλο ZKP ο επαληθευτής πρέπει να αποδείξει τον ισχυρισμό υποθέτοντας ότι ο επαληθευτής δεν γνωρίζει τη λύση (την ουσία της δήλωσης) και ότι η επαλήθευση πραγματοποιείται χωρίς ο επαληθευτής να ανταλλάξει πληροφορίες.

Το σχέδιο μπορεί να συνοψιστεί ως εξής:

Ο Αποδεικνύων (δήλωση) ———-> [Απόδειξη της Γνώσης] ———> Επικυρωτής (επαλήθευση).

2.3.1.2 Schnorr's interactive ZKP

Η κυριά διαφορά αναμεσά στο διαδραστικό πρωτόκολλο ZKP και στο μη διαδραστικό είναι πως τα δυο μέλη χρησιμοποιώντας μαθηματικούς υπολογισμούς προσπαθούν να πείσουν ο ένας τον άλλον χωρίς να ανταλλάξουν ευαίσθητες πληροφορίες.

Αυτό το πρωτόκολλο μπορεί να χρησιμοποιηθεί ως μηχανισμός επαλήθευσης ταυτότητας στον οποίο μια τράπεζα και η δημόσια παράμετρος (B) ενός πελάτη διατηρείται από την τράπεζα. Ο μυστικός κωδικός του πελάτη θα μπορούσε να είναι ένας μυστικός αριθμός [a] (για παράδειγμα το PIN). Ο πελάτης πρέπει να δείξει ότι γνωρίζει το [a] για να αποκτήσει πρόσβαση στον διαδικτυακό του λογαριασμό.

Σε ένα άλλο παράδειγμα χρήσης, η τράπεζα μπορεί να είναι μια κεντρική μονάδα υπολογιστικής ισχύος (διακομιστής - server) και ο πελάτης μπορεί να είναι ένας χρήστης που επιθυμεί να συνδεθεί στον διακομιστή μέσω μιας μη ασφαλούς γραμμής ή ο πελάτης μπορεί να είναι ένας άλλος διακομιστής. Ο σκοπός της χρήσης ενός ZKP είναι να αποτρέψει τον πελάτη από το να αποκαλύψει κρίσιμες πληροφορίες στο κοινό.

2.3.1.3 zk-SNARKs

Οι μη διαδραστικές αποδείξεις μηδενικής γνώσης (Non-interactive zero-knowledge proofs), επίσης γνωστές ως zk-SNARK ή zk-STARK, είναι ένα είδος ZKP που δεν απαιτεί αλληλεπίδραση μεταξύ του prover και του επαληθευτή.

Το όνομα zk-SNARK σημαίνει Μηδενική Γνώση Συνοπτικό Μη Διαδραστικό Επιχείρημα Γνώσης (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge). Έτσι, βρισκόμαστε αντιμέτωποι με συστήματα που χρειάζονται μόνο μία αλληλεπίδραση μεταξύ τους ο prover και ο επαληθευτής.

Το πρώτο κρυπτονόμισμα που υιοθέτησε αυτό το νέο σύστημα για να δημιουργήσει συναίνεση ήταν το Zcash[53].

Η χρήση των zk-SNARKs σε ένα blockchain είναι σημαντική, όπως θα δούμε στη συνέχεια, για τη χρήση έξυπνων συμβολαίων. Όπως ίσως γνωρίζετε, ένα έξυπνο συμβόλαιο είναι κώδικας που υπάρχει στο blockchain και ενεργοποιείται μετά την ολοκλήρωση μιας συμφωνημένης συνθήκης.

Για παράδειγμα, ας υποθέσουμε ότι η Peggy κάνει μια πληρωμή στο Ethereum για να εκτελέσει ένα έξυπνο συμβόλαιο που ανήκει στον Victor. Σε αυτήν την περίπτωση, τόσο η Peggy όσο και ο Victor θέλουν να είναι σίγουροι ότι η εκτέλεση του έξυπνου συμβολαίου (για την Peggy) και η πληρωμή που ελήφθη (για τον Victor) έχουν ολοκληρωθεί με επιτυχία. Ωστόσο, πολλές λεπτομέρειες που είναι εγγενείς στο έξυπνο συμβόλαιο δεν θα αποκαλυφθούν. Έτσι, ο ρόλος που διαδραματίζουν τα zk-SNARK είναι θεμελιώδης για την κάλυψη αυτών των μυστικών και την εκτέλεση έξυπνων συμβολαίων. Για να λειτουργήσει, το πρωτόκολλο πρέπει να είναι γρήγορο, ασφαλές και εύκολο να εφαρμοστεί. Όπως έχουμε ήδη δει, θα παρατηρήσετε ότι

αυτός είναι ακριβώς ο σκοπός ενός ZKP – να διευκολύνει την πλοήγηση σε ένα αναξιόπιστο περιβάλλον.

Έτσι, σε αυτό το περιβάλλον, οι zk-SNARK κρατούν μυστικά προστατεύοντας τα βήματα που περιλαμβάνονται σε ένα έξυπνο συμβόλαιο και, ταυτόχρονα, αποδεικνύοντας ότι όλα αυτά τα βήματα έχουν εκτελεστεί. Με αυτόν τον τρόπο, προστατεύουν το απόρρητο των ανθρώπων και των εταιρειών. Να θυμάστε ότι –όχι επειδή πρέπει να είστε εξαιρετικά δύσπιστοι, αλλά επειδή πρέπει να είστε και ρεαλιστές– αυτή η δήλωση ισχύει υπό καθορισμένες συνθήκες, τις οποίες θα προσπαθήσω να εξηγήσω ως εξής:

- Η απόδειξη που δίνεται από τον prover έχει τον ίδιο υπολογιστικό βαθμό δυσκολίας με τον υποκείμενο αλγόριθμο που επιλέχθηκε ως απόδειξη γνώσης.

- Δεν υπάρχει κανένας μαθηματικός τρόπος για να εξαπατήσετε τον επαληθευτή με μια συντόμευση ή ψεύτικη απόδειξη.

2.3.2 Ελλειπτικές Καμπύλες (Elliptic Curves)

2.3.2.1 Εισαγωγή

Το μελλοντικό όριο για την αποκεντρωμένη χρηματοδότηση είναι οι ελλειπτικές καμπύλες. Για να επιτρέψει τη μετάδοση ψηφιακού χρήματος σε Bitcoin, ο Satoshi Nakamoto χρησιμοποίησε το secp256K1, ένα είδος ελλειπτικής καμπύλης. Ας ρίξουμε μια ματιά στο πώς λειτουργεί και ποιες είναι οι βασικές ιδιότητες αυτής της πολύ ασφαλούς κρυπτογράφησης.

Οι Victor Miller και Neal Koblitz επινόησαν ελλειπτικές καμπύλες για εφαρμογές κρυπτογραφίας περίπου το 1985. Ο Hendrik Lenstra έδειξε αργότερα πώς να παραγοντοποιήσεις έναν ακέραιο αριθμό χρησιμοποιώντας αυτές.

Μερικοί από τους αλγόριθμους που εξετάσαμε σε προηγούμενα κεφάλαια, όπως οι RSA, Diffie–Hellman (D–H) και ElGamal, υλοποιούνται με χρήση της κρυπτογραφίας ελλειπτικών καμπυλών (ECC).

Επιπλέον, μετά την επανάσταση των ψηφιακών νομισμάτων, ένας συγκεκριμένος τύπος ελλειπτικής καμπύλης γνωστός ως secp256K1 και ένας αλγόριθμος ψηφιακής υπογραφής γνωστός ως Elliptic Curve Digital Signature Algorithm (ECDSA) χρησιμοποιήθηκαν για την εφαρμογή ψηφιακών υπογραφών στο Bitcoin προκειμένου να διασφαλιστεί ότι οι συναλλαγές ολοκληρώθηκαν με επιτυχία.

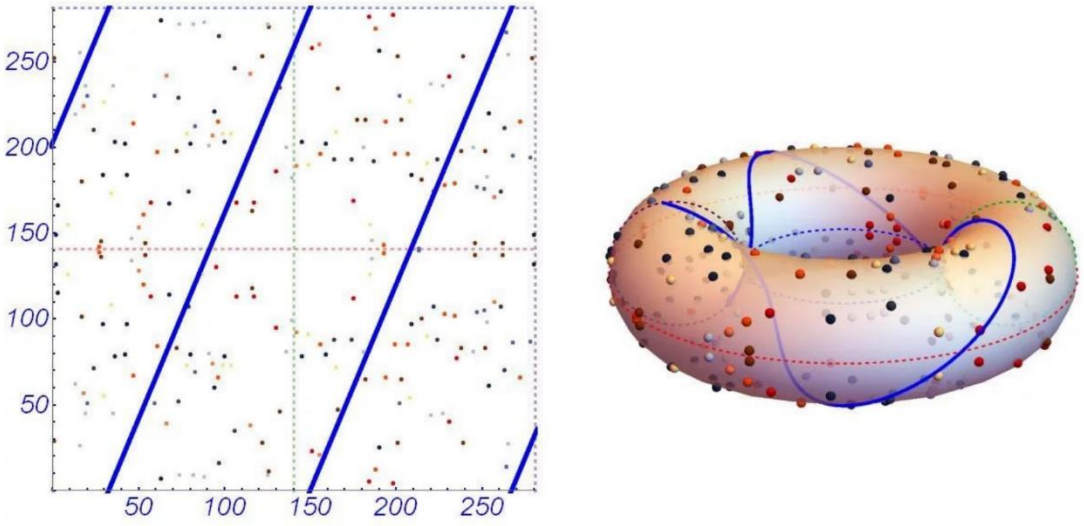
Στις ελλειπτικές καμπύλες, η κρυπτογράφηση 313 bit προβλέπεται να παρέχει έναν βαθμό ασφάλειας συγκρίσιμο με την κρυπτογράφηση 4.096 bit σε ένα τυπικό ασύμμετρο σύστημα. Πολλές υλοποιήσεις που απαιτούν εξαιρετική απόδοση στο χρονοδιάγραμμα και το εύρος ζώνης, όπως οι εφαρμογές για κινητά, μπορεί να ωφεληθούν από τόσο χαμηλούς αριθμούς bit.

Εάν και ελλειπτικές καμπύλες μπορούν να συνδυαστούν με αρκετούς ασύμμετρους αλγορίθμους για λογούς συντομίας θα αναφερθούμε μονάχα στο μοντέλο secp256k1 που χρησιμοποιεί το Bitcoin.

2.3.2.2 Η ψηφιακή υπογραφή του Bitcoin

Ο αλγόριθμος ψηφιακής υπογραφής ECDSA, ο οποίος χρησιμοποιεί μια ελλειπτική καμπύλη που ονομάζεται secp256k1 και καθορίζεται από την ομάδα Standards for Efficient Cryptography, χρησιμοποιείται στην αρχιτεκτονική Bitcoin (SECG).

Η ελλειπτική καμπύλη έχει δύο παραστάσεις: μία στο πραγματικό επίπεδο και μία στο φανταστικό επίπεδο. Όταν τα σημεία καθορίζονται σε ένα πεπερασμένο πεδίο, το σχήμα μιας ελλειπτικής καμπύλης μπορεί να αναπαρασταθεί τρισδιάστατα με έναν δακτύλιο, όπως φαίνεται στο διάγραμμα:



Εικόνα 2.3.1.1.1 *– Τρισδιάστατη Αναπαράσταση μιας ελλειπτικής καμπύλης σε πεπερασμένο πεδίο. Διαθέσιμο στο βιβλίο [55], σελ. 258, Figure 7.8*

Στο πεδίο Z, η καμπύλη secp256k1 ορίζεται ως εξής:

Z modulo 2^256 – 2^32 - 977

Έτσι εμφανίζεται ως ακέραιος:

115792089237316195423570985008687907853269984665640564039457584007908834671663

Πριν από την εφεύρεση του Bitcoin, η καμπύλη secp256k1 χρησιμοποιούταν σπάνια. Όπως μπορείτε να περιμένετε, κέρδισε δημοτικότητα αφού χρησιμοποιήθηκε για το Bitcoin. Η Secp256k1 έχει σχεδιαστεί για να είναι πιο αποδοτική σε θέμα χρόνου από τις περισσότερες «αδερφές» της, οι οποίες χρησιμοποιούν μια τυχαία δομή, είναι 30% πιο γρήγορη από άλλες.

Επιπλέον, σε αντίθεση με τις υπόλοιπες καμπύλες από το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST), ο εφευρέτης επέλεξε σταθερές (a και b) τέτοιες ώστε να υπάρχει μικρότερη πιθανότητα να τοποθετηθεί μια «πίσω πόρτα» σε αυτήν.

Τέλος, η γεννήτρια (G), η τάξη (n) και ο πρώτος (modulo p) στο secp256k1 δεν επιλέγονται τυχαία αλλά μάλλον ως συναρτήσεις άλλων παραγόντων. Όλα αυτά αθροίζονται σε μία από τις καλύτερες ελλειπτικές καμπύλες που μπορείτε να χρησιμοποιήσετε και είναι πολύ χρήσιμη στο πλαίσιο των ψηφιακών υπογραφών. Αυτός είναι ο λόγος που οι δημιουργοί του Bitcoin αποφάσισαν να τη χρησιμοποιήσουν.

3. Βασικά Ειδή Κρυπτονομισμάτων

3.1 Εισαγωγή – Ορισμός του Token

Ο όρος "token" προέρχεται από την παλιά αγγλική λέξη "tācen", που σημαίνει "σημάδι" ή "σύμβολο". Το token χρησιμοποιείται συνήθως για να αναφέρεται σε ιδιωτικά εκδοθέντα αντικείμενα που μοιάζουν με κέρματα ασήμαντης εγγενούς αξίας, όπως μάρκες μεταφοράς, μάρκες πλυντηρίου και μάρκες παιχνιδιών arcade.

Σήμερα, οι εκδόσεις που βασίζονται στο blockchain μπορούν να αντιπροσωπεύουν περιουσιακά στοιχεία, χρήματα ή δικαιώματα πρόσβασης αναφέρονται ως "tokens". Η συσχέτιση μεταξύ της λέξης "token" και της ασήμαντης αξίας έχει να κάνει πολύ με την περιορισμένη χρήση των φυσικών εκδόσεων των διακριτικών αυτών στοιχείων. Οι φυσικές μάρκες (tokens) δεν ανταλλάσσονται εύκολα και συχνά έχουν μόνο μία χρήση. Συχνά περιορίζονται σε ορισμένες εταιρείες, οργανισμούς ή μέρη. Αυτοί οι περιορισμοί αίρονται ή, ακριβέστερα, επαναπροσδιορίζονται πλήρως με ορισμό του token στο blockchain[25].

Πολλά διακριτικά στοιχεία (tokens) που βρίσκονται στο blockchain χρησιμοποιούνται για διάφορους σκοπούς σε όλο τον κόσμο και μπορούν να ανταλλάσσονται μεταξύ τους ή με άλλα νομίσματα σε παγκόσμιες αγορές ρευστότητας (liquidity marketplaces). Η υπόθεση της «ασήμαντης αξίας» ανήκει στο παρελθόν, τώρα πλέον έχουν αρθεί οι περιορισμοί στη χρήση και την ιδιοκτησία[25].

Το πρώτο blockchain που χρησιμοποίησε tokens είναι το Bitcoin. Κατά μια έννοια, το Bitcoin blockchain είναι και αυτό ένα είδος πρωτοτύπου token, αφού δημιουργήθηκαν και αλλά blockchain βασισμένα στο Bitcoin. Πριν από το Ethereum, πολλά συστήματα βασιζόμενα σε διακριτικά στοιχεία χτίστηκαν που ήταν παρόμοια με το Bitcoin και διαθέταν τα δικά τους κρυπτονομίσματα. Όταν όμως εκδόθηκε το πρώτο πρότυπο token του Ethereum δημιούργησε μια έκρηξη στην χρήση των token. Γι’ αυτόν τον λόγο **τα κρυπτονομίσματα που ακολουθούν είναι βασισμένα στο Ethereum Blockchain**[33].

3.2 Τυπικά Νομίσματα (ERC-20)

Το ERC-20 είναι ένα κρυπτονόμισμα, ή καλυτέρα ένα πρότυπο κρυπτονομίσματος (Token Standard), το οποίο έχει τα ιδιά χαρακτηριστικά με τα παραδοσιακά νομίσματα, δηλαδή είναι εναλλάξιμο και ανταλλάξιμο. Γι’ αυτόν το λόγω η πιο συχνή χρήση του είναι ως token που αναπαριστά την αξία ενός αντικειμένου ή υπηρεσίας.

Ο Fabian Vogelsteller παρουσίασε το πρώτο πρότυπο του ERC-20 τον Νοέμβριο του 2015 ως αίτημα στο Ethereum για σχολιασμό (ERC). Του δόθηκε ο αριθμός έκδοσης GitHub με αριθμό 20 από προεπιλογή, κερδίζοντας το παρατσούκλι "ERC20 token". Το πρότυπο ERC20 χρησιμοποιείται επί του παρόντος περισσότερο απ’ όλα τα αλλά είδη κρυπτονομισμάτων. Το αίτημα ERC20 εξελίχθηκε σε μια Πρόταση Βελτίωσης Ethereum 20 (EIP-20), ωστόσο εξακολουθεί να αναφέρεται συχνά ως ERC20.

Το ERC20 είναι ένα ανταλλάξιμο πρότυπο token, πράγμα που σημαίνει ότι οι μεμονωμένες μονάδες Token ERC20 είναι εναλλάξιμες και δεν έχουν μοναδικά χαρακτηριστικά.

Το πρότυπο ERC20 δημιουργεί μια ενιαία διεπαφή για συμβάσεις υλοποίησης διακριτικών (tokens), επιτρέποντας την πρόσβαση και τη χρήση οποιουδήποτε συμβατού διακριτικού με τον ίδιο τρόπο. Η διεπαφή αποτελείται από ένα σύνολο λειτουργιών που πρέπει να περιλαμβάνονται σε οποιαδήποτε τυπική υλοποίηση, καθώς και ορισμένες προαιρετικές λειτουργίες και χαρακτηριστικά που ενδέχεται να προσθέσουν οι προγραμματιστές.

3.3 Μοναδικά Τεκμήρια (NFT, ERC-721)

Το *Μοναδικό Τεκμήριο* (**N**on-**F**ungible **T**oken, γνωστό και ως **NFT**) είναι ένα κρυπτονόμισμα [32] που αναπτύχθηκε από τα έξυπνα συμβόλαια στο Ethereum blockchain[33]. Το NFT προτάθηκε αρχικά στο EIP-721 [34] και αναπτύχθηκε περαιτέρω στο EIP-1155 [35].

Όσον αφορά για τις βασικές του ιδιότητες, το NFT διαφέρει από άλλα κρυπτονομίσματα [36] όπως για παράδειγμα το Bitcoin [37]. Το Bitcoin είναι ένα τυπικό νόμισμα στο οποίο όλα τα νομίσματα είναι ίσα και δεν διακρίνονται μεταξύ τους, δηλαδή μπορεί να παρομοιαστεί με ERC-20 στο Ethereum. Το NFT, από την άλλη πλευρά, είναι μοναδικό στο είδος του και δεν μπορεί να αντικατασταθεί όπως ένα τυπικό νόμισμα, καθιστώντας το ιδανικό για να αποδίδει μοναδικότητα σε κάτι ή κάποιον[38]. Για παράδειγμα, ένας πίνακας ζωγραφικής μπορεί να αντιστοιχηθεί με ένα NFT, διότι είναι μοναδικός, ανήκει (αρχικά) στον δημιουργό του και για να τον αξιολογήσουμε θα πρέπει να χρησιμοποιήσουμε κάποιο μετρήσιμο μέγεθος (πχ. Νομίσματα, USD, bitcoin ή ERC-20).

Συνεχίζοντας, **ένας δημιουργός χρησιμοποιώντας τις δυνατότητες των NFT** από τα έξυπνα συμβόλαια (στο Ethereum [33]) μπορεί εύκολα να **αποδείξει την ύπαρξη, γνησιότητα και τα δικαιώματα ιδιοκτησίας ψηφιακών στοιχείων** τα οποία μπορούν να έχουν την μορφή ταινιών, φωτογραφιών, έργων τέχνης [34], εισιτηρίων εκδηλώσεων [35], πιστοποιητικών εγγράφων, ταυτοτήτων, πτυχίων, κτλ.

Επιπλέον, ο δημιουργός μπορεί να λάβει ποσοστά για κάθε επιτυχημένη συναλλαγή σε οποιαδήποτε αγορά NFT ή ανταλλαγή peer-to-peer. Το NFT έχει τη δυνατότητα να γίνει μια βιώσιμη λύση για την προστασίας της πνευματικής ιδιοκτησίας (IP) λόγω της πλήρους εμπορεύσιμης ιστορίας του, της βαθιάς ρευστότητας (Deep Liquidity) και της εύκολης διαλειτουργικότητας του. Παρά το γεγονός ότι τα **NFTs είναι ουσιαστικά** απλώς **κώδικας**, αυτός ο κώδικας όμως **έχει χρηματική αξία** για έναν αγοραστή όταν λαμβάνεται υπόψη τη σχετική έλλειψή τους ως ψηφιακό περιουσιακό στοιχείο.

Η χρησιμότητα των NFT ξεχωρίζει ιδιαίτερα όσον αφορά τη δημιουργία τέχνης [108], έκδοσης εκδηλώσεων εισιτηρίων [40], IoT [100] καθώς και στα παιχνίδια και στις διάφορων ειδών συλλογές[38]. Άλλες χρήσεις είναι στατιστικοί ιστότοποι για NFTs (π.χ. NonFungible [49],

DappRadar [24], NFT Bank[45], DefiPulse [26], Coingecko [15]) και αγορές για συναλλαγές NFTs (cryptoslam [21], Opensea [50]], SuperRare[59], Nifty Gateway [48], Rarible [55], Zora [69]).

Παρά το γεγονός ότι οι NFT έχουν τεράστια δυνητική επιρροή στις παρούσες αποκεντρωμένες αγορές και στις μελλοντικές οικονομικές προοπτικές, βρίσκονται ακόμη σε πρώιμη φάση. Ορισμένα πιθανά εμπόδια πρέπει να αντιμετωπιστούν πρώτα, ενώ πρέπει να τονιστούν ευνοϊκές προοπτικές.

3.4 Παρτίδες Νομισμάτων (ERC-1155)

Το πρότυπο που χρησιμοποιεί παρτίδες διάφορων νομισμάτων ή διακριτών στοιχείων (tokens) είναι το ERC-1155 όπως και αυτά που προαναφέρθηκαν είναι κώδικας ο οποίος παρέχει λειτουργικότητα και επιτρέπει την δημιουργία πρωτοτύπων τα οποία οι προγραμματιστές πρέπει να εφαρμόσουν εάν επιθυμούν τα δικά τους νομίσματα να μπορούν να χρησιμοποιηθούν σε δημοσιά blockchain.

Όπως και στις αμαξοβιομηχανίες, είναι δύσκολο να κατασκευαστεί λογική που να αποδίδει στις παρτίδες μοναδικότητα, δυνατότητα διαχωρισμού ή συγχώνευσης. Για να αντιμετωπίσουμε το πρόβλημα που υπήρχε και στο blockchain, δημιουργήθηκαν λειτουργίες διακριτικών στοιχείων (tokens) βασισμένες στο πρότυπο διακριτικών στοιχείων ERC-1155 της Enjin (Rolovic coinfork, 2019).

Το ERC-1155 δημιουργήθηκε αρχικά από την Enjin για να βοηθήσει τους παραγωγούς παιχνιδιών blockchain να χειρίζονται πιο αποτελεσματικά τα εικονικά αντικείμενα των χρηστών τους. Ορισμένα εικονικά αντικείμενα είναι μοναδικά(one-of-a-kind), απαιτώντας τη χρήση του προτύπου ERC-721 για κάθε αντικείμενο. Άλλα εικονικά αντικείμενα, όπως τα χρήματα ενός παιχνιδιού, μπορεί να είναι εναλλάξιμα, καθιστώντας απαραίτητη τη χρήση του πρωτοτύπου ERC-20. Το ERC-1155 κατέστησε δυνατό τον χειρισμό και των δύο ειδών εικονικών αντικειμένων με ένα και μόνο έξυπνο συμβόλαιο (smart contract), με αποτέλεσμα να παρατηρηθούν σημαντικά πλεονεκτήματα όσον αφορά την απόδοσης και τον χώρο που απαιτείται στα μπλόκς (Castonguay, 2018).

Γενικότερα, τα κυριότερα οφέλη της εφαρμογής και της υιοθέτησης του πρωτοτύπου ERC-1155 είναι τα ακόλουθα:

- **Υποστηρίζει πολυάριθμα διακριτικά στοιχεία (tokens)**, τα οποία απαιτούνται για την αναπαράσταση διαφόρων ειδών αντικειμένων και περιπτώσεων.

- **Επιτρέπει τη σύνδεση των διακριτικών με ένα URI** (διαδρομή αρχείου) όπου μπορούν να διατηρούνται δεδομένα εκτός αλυσίδας(πχ. IPFS), όπως πρόσθετα δεδομένα παραγωγής.

• **Επιτρέπει την αναπαράσταση πολλών οντοτήτων του ίδιου διακριτικού στοιχείου**, που απαιτείται για την περιγραφή διαφόρων μετασχηματισμών εισόδου-εξόδου, όπως η κατανάλωση πολλών παρτίδων από μια νέα μονάδα.

Β. Πρακτικό Μέρος

1. Προδιάγραφες Εφαρμογής

1.1 Σκοπός Εφαρμογής

Η εφαρμογή που θα μελετήσουμε λεπτομερώς σε όλο το υπόλοιπο του συγκεκριμένου εγγράφου αφορά την χρήση του πρωτοτύπου κρυπτονομίσματος ERC-721 (NFT) για την δημιουργία μοναδικών τεκμήριων τα οποία θα αντιπροσωπεύουν πτυχία φοιτητών.

Η χρήση των τεχνολογιών blockchain έχει αρχίζει να κινεί το ενδιαφέρον πολλών κλάδων της επιστήμης και της αγοράς, ένας από αυτούς είναι και η εκπαίδευση. Το blockchain στο χώρο της εκπαίδευσης μπορεί να μας προσφέρει πολλά πλεονεκτήματα[76], μερικά από αυτά είναι:

1. Αυτοματοποίηση διαδικασιών
2. Εύκολη & γρήγορη επαλήθευση στοιχείων
3. Δικαιότερη αξιολόγηση (μαθητών & καθηγητών)
4. Ανώτερη οργάνωση
5. Καλύτερη ποιότητα παροχής υπηρεσιών

Στη παρούσα εφαρμογή θα ασχοληθούμε με την πρώτη κατηγορία πλεονεκτημάτων, τον αυτοματισμό διαδικασιών. Η ιδέα είναι πως με την χρήση έξυπνων συμβολαίων θα είναι εφικτό τα ανωτέρα στελέχη (πχ. Γραμματεία) ενός οργανισμού εκπαίδευσης να μπορούν να αποδίδουν κάποια δικαιώματα στους υπάλληλους (πχ. Καθηγητές) τα οποία θα τους επιτρέπουν να δημιουργούν ολοκληρωμένα προγράμματα σπουδών, να εγγράφουν νέους φοιτητές και να ανεβάζουν βαθμούς στο blockchain.

Στη συνέχεια, όταν ένας φοιτητής πιστεύει ότι πληροί όλες τις προϋποθέσεις για την έκδοση του πτυχίου του θα μπορεί να επισκεφτεί την αποκεντρωμένη εφαρμογή μας οπού πρώτα θα του ζητηθεί να συνδέσει το κρυπτό-πορτοφόλι του και έπειτα τοποθετώντας τον αριθμό μητρώου του θα γίνει ένας έλεγχος εάν πραγματικά πληροί όλες τις προϋπέθεσες για την

έκδοση του πτυχίου του. Μετά τον έλεγχο, εάν πληροί τις προϋποθέσεις θα εμφανιστεί ένα κουμπί που πάνω του θα αναγράφεται «Degree Issuance» και μόλις το πατήσει ο φοιτητής θα δημιουργηθεί ένα ERC-721 token που θα αντιστοιχεί στο πτυχίο του. Ακόμα, θα εμφανιστεί άλλο ένα κουμπί που πάνω του θα αναγράφεται «Download!» και μόλις το πατήσει ο φοιτητής θα κατεβεί σε μορφή pdf το πτυχίο του (πιο σωστά, ένα υπόδειγμά πτυχίου).

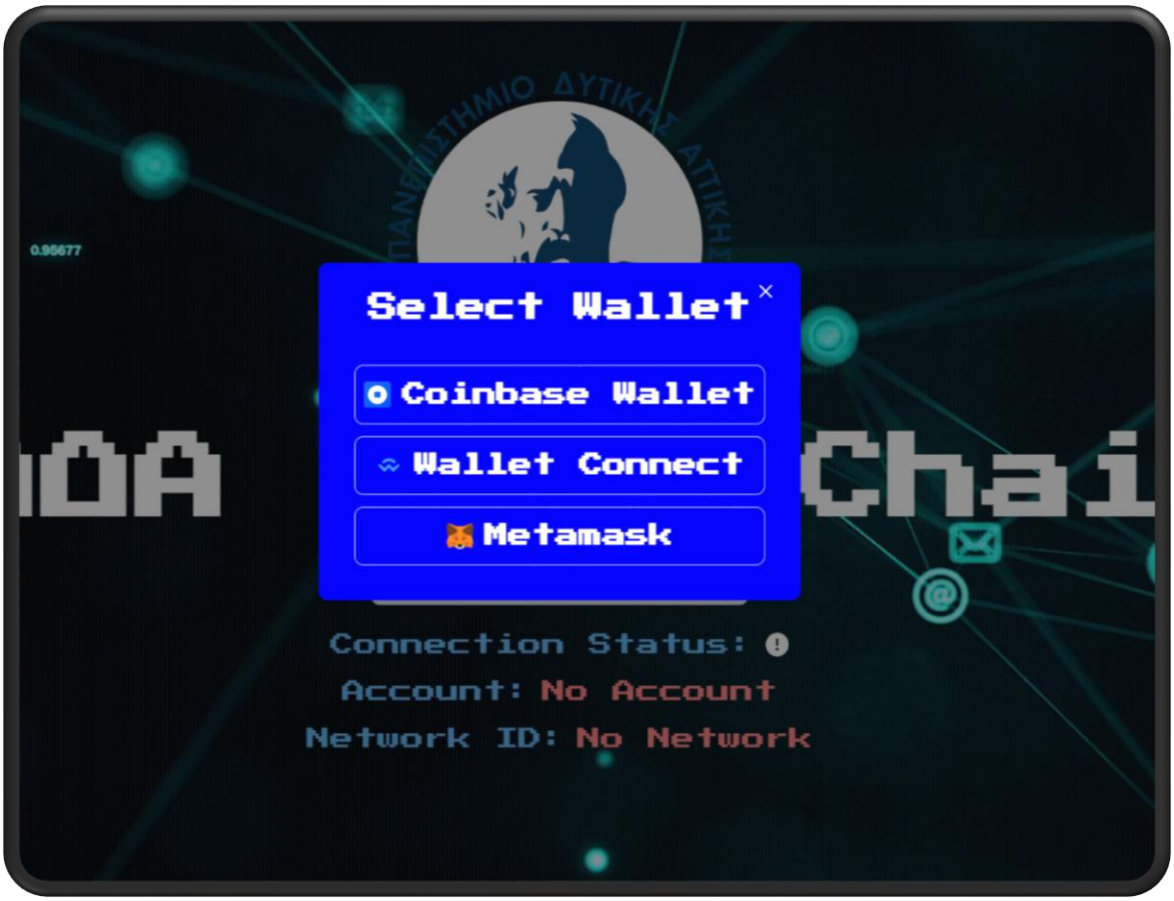
Στην περίπτωση που δεν πληροί τις προϋποθέσεις θα εμφανιστεί απλώς ένα μήνυμα που λέει ότι δεν μπορεί να εκδώσει το πτυχίο του.

1.2 Παρουσίαση Εφαρμογής

Πριν αρχίσουμε να αναλύουμε τις τεχνικές λεπτομέρειες της εφαρμογής, πιστεύω πως είναι προτιμότερο πρώτα να παρατηρήσουμε και να σχολιάσουμε την λειτουργικότητα της.

Ας ξεκινήσουμε με την την αρχική σελίδα που θα αντικρίσει ο χρήστης με το που θα επισκεφτεί την ιστοσελίδα της εφαρμογής μας.

Παρατηρούμε ότι απευθείας ανοίγει το Metamask (βλ. Κεφ. Χ.Χ.Χ), αλλά στην περίπτωση που αυτό δεν συμβεί, πατώντας το κουμπί “Connect Wallet” θα εμφανιστεί στο χρήστη ένα παραθυράκι που θα του δίνει την δυνατότητα να συνδεθεί με έναν από τους τρεις διαθέσιμους τρόπους (ή υποστηριζόμενα wallets) όπως φαίνεται στην εικόνα πιο κάτω.



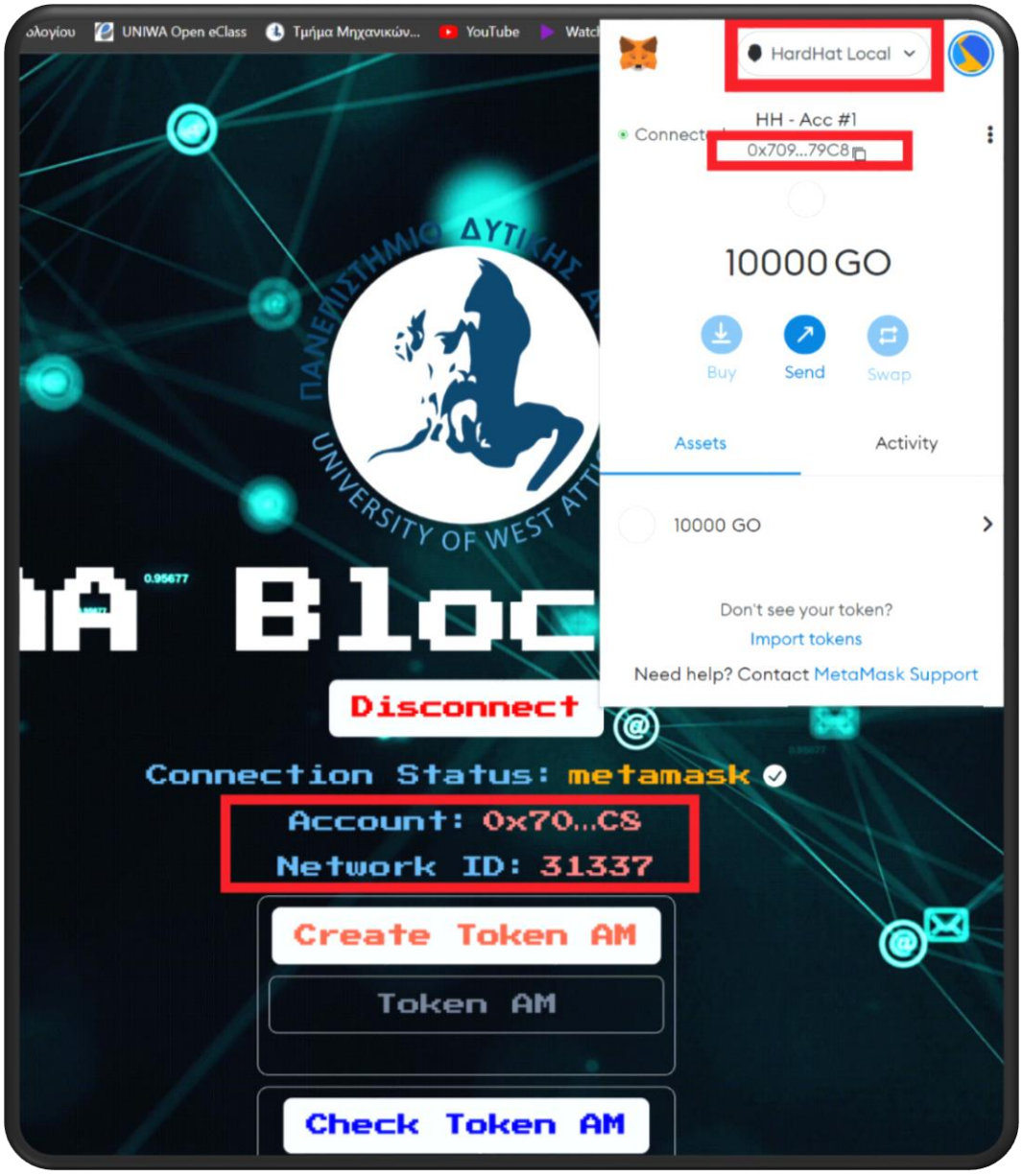
Η εφαρμογή έχει χτιστεί και δοκιμαστεί χρησιμοποιώντας μόνο το Metamask, οπότε υπάρχει μεγάλη πιθανότητα εάν χρησιμοποιήσετε κάποια από τις άλλες δυο επιλογές να συναντήσετέ δυσκολίες και προβλήματα.

Μόλις συνδέσουμε το wallet μας βάζοντας το κωδικό πρόσβασης, παρατηρούμε ότι η σελίδα αλλάζει μορφή χωρίς να χρειάζεται φόρτωση ή μετάβαση σε κάποια να νέα σελίδα. Αυτό συμβαίνει επειδή η εφαρμογή έχει δημιουργηθεί χρησιμοποιώντας την βιβλιοθήκη React που παρέχει αυτού του είδους δυναμικότητα στις εφαρμογές της. Επίσης παρατηρούμε ότι άλλαξε και το Metamask, τώρα μας δείχνει το προεπιλεγμένο blockchain δίκτυο (Ethereum Mainnet) καθώς το υπόλοιπο μας στο κρυπτονόμισμα ETH (ether).


UNIWA Open eClass
YouTube
Ethereum Mainnet
Connected
HH - Acc #0
0xf39...2266
0 ETH
$0.00 USD
Buy
Send
Swap
Assets
Activity
0 ETH
$0.00 USD
2000 DANK
Disconnect
Connection Status: metamask
Account: 0xf3...66
Network ID: 1
Create Token AM
Token AM
Check Token AM
Insert Token
Can I Graduate?

Ακόμα, βλέπουμε ότι υπάρχουν τρεις δείκτες (indicators) κάτω από το κουμπί “Disconnect”, ο πρώτος μας δείχνει το τρόπο με τον οποίο έχουμε επιλέξει να συνδεθούμε, ο δεύτερος τον λογαριασμό που χρησιμοποιούμε αυτή την στιγμή (εάν πάμε στο Metamask και αλλάξουμε λογαριασμό, αμέσως θα αλλάξει και ο δείκτης), και τέλος ο τρίτος μας παρουσιάζει το αναγνωριστικό δικτύου (network/chain identifier, επίσης ανανεώνεται άμεσα).

Ας αλλάξουμε το δίκτυο στο τοπικό δίκτυο blockchain που μπορούμε να δημιουργήσουμε μέσω του Hardhat (θα δούμε στο Κεφ. X.X.X αναλυτικά πως αυτό γίνεται) και τον λογαριασμό για να δούμε πως θα ανταποκριθεί η εφαρμογή μας.



Τώρα που είδαμε πως μπορούμε να συνδεθούμε στην εφαρμογή χρησιμοποιώντας το Metamask, ήρθε η στιγμή να παρατηρήσουμε την λειτουργικότητα της εφαρμογής.

Για λογούς συντομίας, έχω ήδη δημιουργήσει τα δεδομένα που θα χρειαστούμε για να δοκιμάσουμε την εφαρμογή μέσα στον κώδικα των έξυπνων συμβολαίων (βλ. Κεφ. Χ.Χ.Χ). Θα χρησιμοποιήσουμε δυο φοιτητές, τον **Α** (με ΑΜ: **45789**, που *δεν μπορεί* να εκδώσει το πτυχίο του) και τον **Β** (με ΑΜ: **43215**, που *μπορεί* να εκδώσει το πτυχίου του).

Περίπτωση **Α**:

1. Ο φοιτητής Α εισάγει το ΑΜ (Αριθμό Μητρώου) του στο πεδίου υποδοχής κειμένου κάτω από το κουμπί “Check Token AM”

2. Ο φοιτητής κλικάρει το κουμπί “Check Token AM” και του εμφανίζεται αμέσως από κάτω ένα μήνυμα με πράσινα γράμματα: “Accepted!”

3. Εάν δοκιμάσουμε ένα τυχαίο αριθμό, θα παρατηρήσουμε ότι μας εμφανίσει το εξής μήνυμα με κόκκινα γράμματα: “Rejected!”

4. Εφόσον υπάρχει καταχωρημένο το ΑΜ του φοιτητή, μπορεί να δοκιμάσει εάν είναι πληροί τις προϋποθέσεις για την έκδοση του πτυχίου του

5. Πιέζοντας το κουμπί “Can I Graduate?”, παρατηρούμε μια ελάχιστη καθυστέρηση, αυτό συμβαίνει διότι το κώδικας που καλεί αυτό το κουμπί είναι αρκετά πιο περίπλοκος από το πρώτο που απλός ελέγχει εάν εάν το ΑΜ του φοιτητή υπάρχει στο blockchain. Επειδή όμως, όπως ήδη γνωρίζαμε, ο φοιτητής Α δεν μπορεί να εκδώσει το πτυχίο του εμφανίζεται με κόκκινα γράμματα το μήνυμα: “Sadly, no 😔”


ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA
0.73455
ΠΑ BlockCha.
Disconnect
Connection Status: metamask
Account: 0x70...C8
Network ID: 31337
Create Token AM
Token AM
Check Token AM
45789
Accepted!
Can I Graduate?
Sadly, no

Περίπτωση **Β**:

1. Ο φοιτητής Β εισάγει το ΑΜ (Αριθμό Μητρώου) του στο πεδίου υποδοχής κειμένου κάτω από το κουμπί “Check Token AM”

2. Ο φοιτητής κλικάρει το κουμπί “Check Token AM” και του εμφανίζεται αμέσως από κάτω ένα μήνυμα με πράσινα γράμματα: “Accepted!”

3. Εφόσον υπάρχει καταχωρημένο το ΑΜ του φοιτητή, μπορεί να δοκιμάσει εάν είναι πληροί τις προϋποθέσεις για την έκδοση του πτυχίου του

4. Πιέζοντας το κουμπί “Can I Graduate?”, όπως ήδη γνωρίζαμε ο φοιτητής Β μπορεί να εκδώσει το πτυχίο του και γι’ αυτό, εμφανίζεται με πράσινα γράμματα το μήνυμα: “ 🏆 Yes, Congrats!” καθώς και ένα νέο κουμπί που ονομάζεται “Degree Issuance”

5. Πατώντας το κουμπί “Degree Issuance” δημιουργείται ένα μοναδικό τεκμήριο (NFT, ERC-721) που αντιστοιχεί στο πτυχίο του φοιτητή Β. Ακόμα, εμφανίζεται άλλο ένα κουμπί που αναγραφεί “Download!”

6. Πατώντας το νέο κουμπί, ο φοιτητής μπορεί να κατεβάσει σε μορφή pdf το πτυχίο του. Ο κώδικας που χειρίζεται το αρχείο pdf δεν είναι ακόμα δυναμικός γι’ αυτό το πτυχίο δεν διαθέτει τα προσωπικά δεδομένα του φοιτητή, είναι απλώς ένα πρόχειρο υπόδειγμα.


ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA
BlockCha
Disconnect
Connection Status: metamask
Account: 0x70...C8
Network ID: 31337
Create Token AM
Token AM
Check Token AM
43215
Accepted!
Can I Graduate?
Yes, Congrats!
Degree Issuance
Download!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla in velit at mauris posuere cursus vitae molestie eros. Integer ullamcorper dolor gravida est eleifend placerat. Mauris vulputate rutrum nisi non ornare. Aliquam sit amet ipsum ex. Vestibulum quis risus a leo aliquam tempor quis a velit. Praesent consectetur sollicitudin dolor, vel finibus ante semper ut. Ut arcu magna, dictum sit amet orci at, pretium faucibus est. Aliquam ultrices elit vitae metus semper, in venenatis tortor pellentesque. Donec mattis iaculis arcu nec viverra. Quisque nec imperdiet lectus, at laoreet ante. Nunc eget imperdiet sapien. Aliquam vulputate nunc metus, quis mattis purus pulvinar et. Pellentesque at lorem lobortis, sollicitudin sapien mattis

2. Τεχνική Αναφορά

2.1 Μεθοδολογία

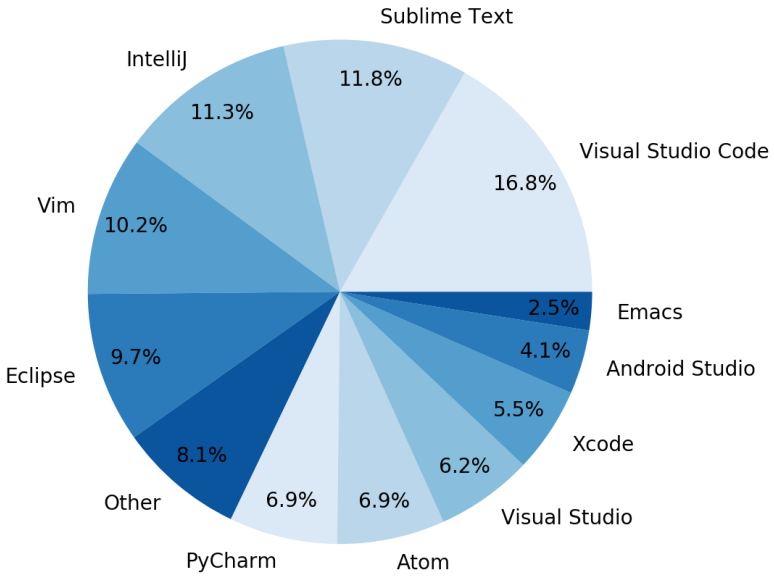
Σε αυτό το υπό-κεφάλαιο θα γίνει μια αναφορά στην μεθοδολογία που πρέπει να ακολουθηθεί για την ανάπτυξη της εφαρμογής. Λεπτομέρειες σχετικά με τον κώδικα, τα εργαλεία και τις τεχνολογίες θα αναφερθούν στο επόμενο υπό-κεφάλαιο.

Βασισμένοι σε όσα μελετήσαμε στο θεωρητικό σκέλος της εργασίας, θα δούμε πως μπορούμε στην πράξη να κατασκευάσουμε μια αποκεντρωμένη εφαρμογή (Dapp). Θα ξεκινήσουμε με την δημιουργία των έξυπνων συμβολαίων (*smart contracts*) χρησιμοποιώντας την γλώσσα προγραμματισμού *Solidity* στο προγραμματιστικό περιβάλλον *Remix IDE*.

Υστέρα, εφόσον έχουμε ολοκλήρωση την κατασκευή των έξυπνων συμβολαίων θα μεταφερθούμε σε τοπικό περιβάλλον προγραμματισμού, δηλαδή θα φύγουμε από το Remix IDE που είναι διαδικτυακό (δηλ., τρέχει στον περιηγητή) και δουλέψουμε στο λειτουργικό σύστημα του υπολογιστή μας.

Σε αυτό το σημείο θα χρειαστούμε ένα επεξεργαστή κειμένου. Ο επεξεργαστής κειμένου είναι ένα πρόγραμμα που όπως μας αποκαλύπτει και το όνομα του, απλώς μας επιτρέπει να γράψουμε και να επεξεργαστούμε κείμενο. Σε αυτήν την εργασία θα χρησιμοποιηθεί σαν επεξεργαστής κειμένου το "*Visual Studio Code*" που θεωρείται από τις πιο δημοφιλείς επιλογές τα τελευταία χρονιά όπως φαίνεται στην *εικόνα 2.1.1*.

Μετά, πρέπει να βρούμε ένα σημείο για να τοποθετήσουμε όλα τα αρχεία που θα απαρτίζουν την εφαρμογή μας. Μπορείτε να χρησιμοποιήσετε οποίο μονοπάτι (path) επιθυμείτε αλλά γενικά όσο πιο κοντά βρίσκεται στη ριζά (root) τόσο καλυτέρα. Για παράδειγμα το δικό μου είναι: "C:\Users\Jimzord\Documents\1_Programming_Stuff\Web3_Apps\degree-maker". Όπως παρατηρείται κατά την ονομασία των φακέλων δεν υπάρχουν κενά, αλλά "_" ή "-" αυτή είναι μέθοδος που χρησιμοποιούν οι προγραμματιστές ώστε να μην αντιμετωπίσουν τυχόν προβλήματα στο κώδικα, προτείνεται να ακολουθήσετέ την ιδιά τεχνική.



Εικόνα 2.1.1 *– Διάγραμμα πίτας που απεικονίζει την δημοτικότητα του κάθε επεξεργαστή κειμένου. Διαθέσιμο [Online]: The rise of Microsoft Visual Studio Code (triplebyte.com)*

Συνεχίζοντας, τώρα είναι η στιγμή που πρέπει να αναφέρουμε σύντομα στα εργαλεία που είναι απαραίτητα για αργότερα. Πρώτα απ' όλα, θα χρειαστούμε το Node.js, που είναι ένα περιβάλλον που μας επιτρέπει να τρέξουμε κώδικα γραμμένο σε JavaScript (ή ECMAScript) χωρίς να χρειαζόμαστε τον περιηγητή (browser). Η εγκατάσταση του είναι πάρα πολύ εύκολη, είναι παρόμοια με τα περισσότερα προγράμματα που θα έχετε εγκαταστήσει. Το καλό με το Node.js είναι πως μαζί του εμπεριέχεται και το "npm" (Node Package Manager) που μας επιτρέπει πολύ ευκολά να έχουμε πρόσβαση σε πηγαίο κώδικα (source code) που άλλοι προγραμματιστές έχουν γράψει και θέλουν να μοιραστούν. Ο κώδικας που παρέχεται από το npm ονομάζεται "εξάρτηση" (dependency).

Το επόμενο βήμα είναι να χρησιμοποιήσουμε το npm για να κατεβάσουμε όλα τα dependencies που είναι απαραίτητα για την εφαρμογή μας. Τα οποία είναι τα ακόλουθα:

Τα dependencies που αφορούν τα Smart Contracts:

1. @openzeppelin/contracts
2. hardhat
3. @nomiclabs/hardhat-ethers
4. @web3-react/core

Τα dependencies που αφορούν το User Interface (UI):

5. react
6. @chakra-ui/react
7. @react-pdf/renderer
8. Smooth-scrollbar

Για να κατεβάσουμε ένα dependency πρέπει να ανοίξουμε το VS Code να πάμε στο χώρο που θέλουμε να αποθηκεύσουμε τα αρχεία μας και μετά να ανοίξουμε την κονσόλα/τερματικό (console/terminal). Στο μενού στην πάνω μεριά του παραθύρου του VS Code υπάρχει μια καρτέλα “Terminal”, επιλέξτε την επιλογή “New Terminal”. Πληκτρολογώντας πρώτα npm «κενό» install «κενό» το «όνομα του dependency» που θέλουμε, το npm θα το κατεβάσει για εμάς. Μια συμβολή για να αποφεύγεται πονοκέφαλους, προτείνεται να επισκέπτεστε πρώτα την ιστοσελίδα του dependency που επιθυμείτε επειδή πάντα υπάρχει μια έτοιμη η εντολή που πρέπει να τοποθετήσετέ στην κονσόλα για να κατεβάσετέ το αντίστοιχο dependency. Ας χρησιμοποιήσουμε την ιστοσελίδα της βιβλιοθήκης Chakra UI ως παράδειγμα:

Εικόνα 2.1.2 *– Οδηγίες εγκατάστασης για την βιβλιοθήκη Chakra. Διαθέσιμο [Online]: https://chakra-ui.com/getting-started*

Εφόσον τώρα μάθαμε τις βασικές ιδιότητες των dependencies, ας προχωρήσουμε στα frameworks. Framework με απλά λογία είναι κώδικας ή ένα πρόγραμμα που δημιουργεί από μόνο του διαφορά αρχεία και φακέλους που είναι είτε απαραίτητα είτε απλώς χρήσιμα για την εφαρμογή που θέλουμε να χτίσουμε. Στην δική μας εφαρμογή θα χρησιμοποιήσουμε δυο frameworks, το **1ο** είναι για το **front-end** (Αυτό που βλέπει ο χρήστης και δέχεται δεδομένα από τον χρήστη) και το **2ο** για το **back-end** (Αυτό που δέχεται τα δεδομένα από το front-end, τα επεξεργάζεται και εκτελεί κάποια ενέργεια βάση των αποτελεσμάτων).

Για το **front-end**, θα χρησιμοποιήσουμε το “Create React App” που μας δημιουργεί ένα περιβάλλον για να φτιάξουμε το User Interface (UI). Για περισσότερες πληροφορίες ανατρέχατε στα έγγραφα της React εδώ: *https://reactjs.org/docs/create-a-new-react-app.html*

Για το **back-end**, θα βασιστούμε στο hardhat. Το οποίο εφόσον το εγκαταστήσουμε με το npm, θα πρέπει να καλέσουμε γράφοντας στην κονσόλα: “npx hardhat”. Θα εμφανιστούν μερικές επιλογές, ανάλογα με το ποσό περίπλοκο θέλουμε να είναι το project μας επιλέγουμε αντίστοιχα. Διότι η παρούσα εφαρμογή δεν είναι ιδιαίτερα περιπλοκή προτείνω την επιλογή: “basic sample project”. Όπως και το “Create React App” έτσι και το hardhat θα δημιουργήσει και θα κατεβάσει ότι χρειαζόμαστε για να μπορούμε να γράψουμε, δοκιμάσουμε, τρέξουμε και προωθήσουμε (deploy) τα έξυπνα συμβόλαια μας.

```
found 0 vulnerabilities
PS C:\Users\Jimzord\Documents\1_Programming_Stuff\Web3_Apps\asdasdas> npx hardhat
888    888                      888 888               888
888    888                      888 888               888
888    888                      888 888               888
8888888888  8888b.  888d888 .d88888 88888b.   8888b.  888888
888    888     "88b 888P"  d88" 888 888 "88b     "88b 888
888    888 .d888888 888    888  888 888  888 .d888888 888
888    888 888  888 888    Y88b 888 888  888 888  888 Y88b.
888    888 "Y888888 888     "Y88888 888  888 "Y888888  "Y888

Welcome to Hardhat v2.9.9

? What do you want to do? ...
> Create a basic sample project
  Create an advanced sample project
  Create an advanced sample project that uses TypeScript
  Create an empty hardhat.config.js
  Quit
```

Εικόνα 2.1.3 *– Οδηγίες χρήσης Hardhat για την δημιουργία ενός προτύπου (template) βασικού project.*

Περιφημά, πλέον έχουμε κατανόηση το μονοπάτι που πρέπει να ακολουθήσουμε για να δημιουργήσουμε την αποκεντρωμένη εφαρμογή μας. Στο υπό-κεφάλαιο που ακολουθεί θα αναλύσουμε με μεγαλύτερη λεπτομέρεια τις λειτουργίες και τους τρόπους χρήσης των τεχνολογιών που αναφέρθηκαν εδώ.

2.2 Εργαλεία & Τεχνολογίες

Σε αυτό το υπό-κεφάλαιο θα αναλύσουμε όλα τα εργαλεία και τεχνολογίες που είναι απαραίτητες για την ανάπτυξη της αποκεντρωμένης εφαρμογής μας. Για την βελτιστοποίηση της οργάνωσης, τα περιεχόμενα έχουν χωριστεί σε δυο κατηγορίες, στο front-end και στο back-end.

2.2.1 Front-End

2.2.1.1 JavaScript (Γλωσσά Προγραμματισμού)

Αν και είναι κυρίως διάσημη ως η γλώσσα δέσμης ενεργειών (scripting language) για ιστοσελίδες, η JavaScript (συνήθως συντομογραφείται ως JS) είναι μια ελαφριά, ερμηνευώμενη (interpreted), αντικειμενοστραφή γλώσσα με συναρτήσεις πρώτης κατηγορίας που χρησιμοποιείται επίσης σε άλλες εφαρμογές χωρίς να είναι απαραίτητο κάποιο πρόγραμμα περιήγησης. Είναι μια δυναμική γλώσσα προγραμματισμού πολλαπλών παραδειγμάτων (multi-paradigm) που βασίζεται σε πρωτότυπα (prototype-based) και υποστηρίζει επιτακτικά (imperative), λειτουργικά (functional) και (object-oriented, OOP) αντικειμενοστραφή παραδείγματα προγραμματισμού[77].

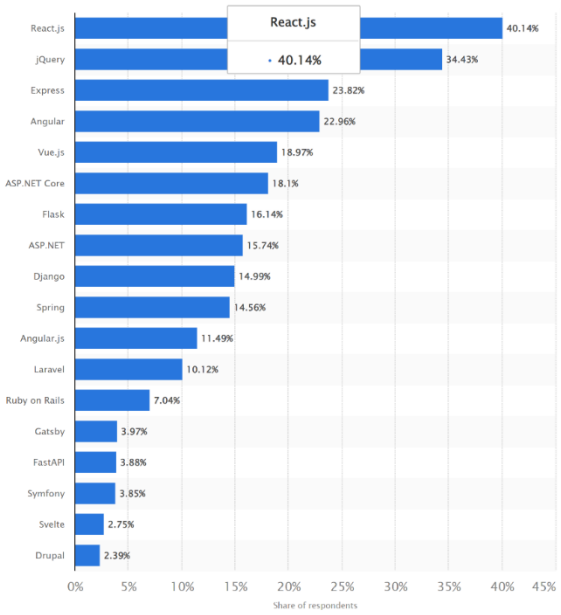
Σε αντίθεση με τις συντακτικές δηλώσεις κλάσεων (syntactic class declarations) τυπικές για μεταγλωττισμένες γλώσσες όπως η C++ και η Java, τα αντικείμενα στην JS σχηματίζονται δυναμικά προσθέτοντας μεθόδους και ιδιότητες σε διαφορετικά κενά αντικείμενα κατά τον χρόνο εκτέλεσης. Μόλις κατασκευαστεί ένα στοιχείο, μπορεί να χρησιμοποιηθεί ως μοντέλο (ή πρωτότυπο) για την κατασκευή άλλων παρόμοιων[77].

2.2.1.2 React (Βιβλιοθήκη)

Η React είναι μια βιβλιοθήκης που διευκολύνει σε μεγάλο βαθμό την κατασκευή User Interfaces (UIs). Ο Jordan Walke, προγραμματιστής στη Facebook, δημιούργησε τη React σε μια προσπάθεια να αντιμετωπίσει τα προβλήματα που υπήρχαν κατά την υλοποίηση των διαφημίσεων. Αν και το έργο του Walke ξεκίνησε το 2010, η εταιρεία του Mark Zuckerberg δεν παρείχε τη React ως λύση ανοιχτού κώδικα μέχρι τον Μάιο του 2013. Έκτοτε, η βιβλιοθήκη έχει προσελκύσει μεγάλο αριθμό υποστηρικτών που συνεχίζουν να εργάζονται για τη βελτίωσή της.

Άλλες μέθοδοι χειρισμού του DOM, όπως η JQuery ή ακόμα και η καθαρή Javascript, παράγουν κώδικα που είναι δυσανάγνωστος και δύσκολο να ενημερωθεί. Προτείνοντας μια νέα αρχιτεκτονική βασισμένη σε στοιχεία (components), τμήματα κώδικα που χρησιμοποιούν HTML, CSS και Javascript έτσι ώστε να περιλαμβάνουν τη λογική και τη παρουσίαση, η React ξεπερνά αυτά τα προβλήματα.

Εκτός από τη React, υπάρχουν άλλες δυο δημοφιλής επιλογές που αρκετοί προγραμματιστές χρησιμοποιούν, τη Angular (κατασκευασμένη από την Google) και τη Vue (κατασκευασμένη από τον Evan You). Βλέποντας όμως το παρακάτω στατιστικό διάγραμμα είναι φανερό ότι η React έχει πολύ μεγαλύτερη επιρροή από τις άλλες δυο. Πράγμα αρκετά σημαντικό στο χώρο του προγραμματισμού, διότι όσο μεγαλύτερο το οικοσύστημα μιας τεχνολογίας τόσο περισσότερο διαθέσιμο υλικό (άρθρα, βίντεο, κώδικας, εφαρμογές, κτλ.) υπάρχει.



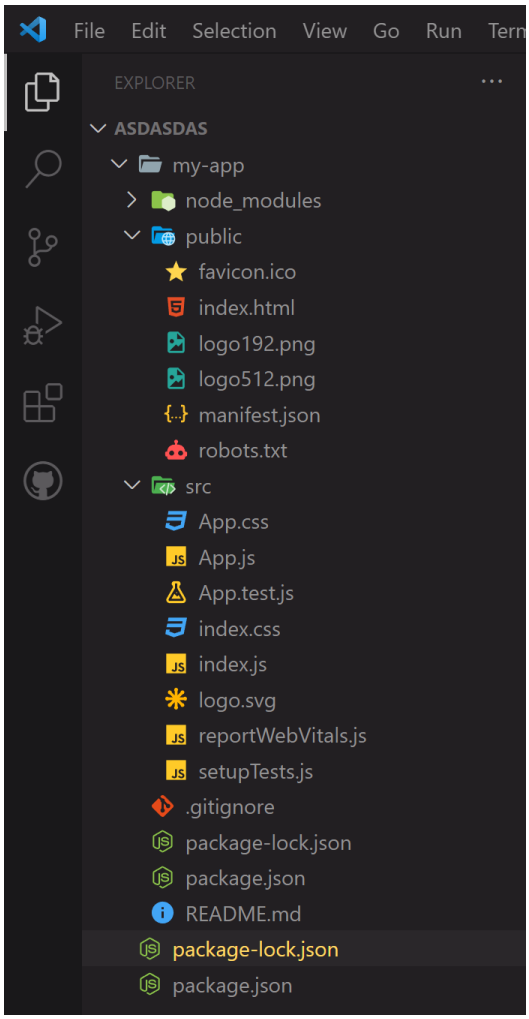
Εικόνα 2.2.1.2 *– Διάγραμμα που απεικονίζει την δημοτικότητα των Frameworks 2021. Διαθέσιμο [Online]: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/*

2.2.1.3 Create-React-App (Framework)

Το Create-React-App (CRA) είναι ένα framework που βασίζεται στην βιβλιοθήκη React. Γι' αυτόν τον λόγο, πολλές φορές αναφερόμαστε στη React και ως framework. Για να το χρησιμοποιήσουμε δεν έχουμε πάρα μόνο να ανοίξουμε το VS Code στον φάκελο που θα τοποθετήσουμε τα αρχεία της εφαρμογής μας, να ανοίξουμε το τερματικό (terminal) και να πληκτρολογήσουμε την ακόλουθη εντολή:

npx create-react-app «Όνομα-Εφαρμογής»

Στην συνέχεια θα δημιουργηθεί το παρακάτω ευρετήριο (directory).



2.2.1.4 Chakra UI (Βιβλιοθήκη)

Η βιβλιοθήκη Chakra μας επιτρέπει να σχεδιάσουμε το μεγάλη ευκολία και ταχύτητα όμορφα και οργανωμένα User Interfaces. Στην ουσία, μας προσφέρει μια μεγάλη ποικιλία ετοίμων εξαρτημάτων που είναι βασισμένα πάνω στη React, με αποτέλεσμα να μην χρειάζεται να ξανά εφεύρουμε τον τροχό.

Η Chakra έχει κερδίσει ενδιαφέρον πολλών προγραμματιστών, κάτι που επιβεβαιώνεται άμα λάβουμε υπόψιν τα παρακάτω στατιστικά στοιχεία:

1. 1,2 εκατ. λήψεις ανά μηνά
2. 27,3 χιλ. αστέρια στο GitHub
3. 7000+ μέλη στο Discord server

2.2.1.5 React-pdf (Βιβλιοθήκη)

Η React-pdf είναι και αυτή μια βιβλιοθήκη βασισμένη στην React. Μας παρέχει λειτουργικότητα όσον αφορά την δημιουργία, επεξεργασία και παρουσίαση αρχείων pdf.

Όπως και η Chakra, έτσι και η React-pdf εμπεριέχει ήδη αρκετά χρήσιμα εξαρτήματα που μπορούμε να χρησιμοποιήσουμε για δημιουργήσουμε το αρχείο pdf που επιθυμούμε.

Στην δική μας εφαρμογή, το αρχείο pdf που μπορεί να κατεβάσει ο φοιτητή έχει δημιουργηθεί με αυτήν την βιβλιοθήκη. Ακόμα, η λειτουργικότητα του «Download!» κουμπιού βασίζεται πάνω σε αυτήν την βιβλιοθήκη.

2.2.1.6 Web3-React (Βιβλιοθήκη)

Όπως πρέπει ήδη να έχετε παρατηρήσει, οι εξωτερικές βιβλιοθήκες που βασίζονται στη React λειτουργούν με την ιδιά φιλοσοφία, δηλαδή μας παρέχουν έτοιμα εξαρτήματα (κώδικα) για να ενισχύσουν την αποδοτικότητα μας καθώς και να αυξάνουν την οργάνωση του κώδικα της εφαρμογής μας.

Η Web3-React δεν αποτελεί εξαίρεση, και αυτήν με την σειρά της μας παρέχει πολλά χρήσιμα εξαρτήματα που μας βοηθούν να επικοινωνήσουμε με το blockchain. Στην δική μας εφαρμογή όμως, τη χρησιμοποιούμε μόνο για να πάρουμε τα δεδομένα που εμφανίζουν οι δείκτες (τρόπος σύνδεσης, επιλεγμένος λογαριασμός, αναγνωριστικό αλυσίδας ή network id).

2.2.1.7 Smooth-Scrollbar-React (Πηγαίος Κώδικας)

Ένα από τα πιο σημαντικά στοιχεία κάθε ιστοσελίδας είναι η δυνατότητα να «σκρολλάρεις». Αλλά όπως ήδη έχουμε αναφέρει δεν θέλουμε να εφευρίσκουμε συνεχώς τον τροχό και γι' αυτό τον λόγο θα χρησιμοποιήσουμε έτοιμο κώδικα.

Εάν και υπάρχουν πολλές άλλες παραλλαγές κώδικα που για «σκρολλάρισμα» επέλεξα τον συγκεκριμένο επειδή η χρήση του είναι ιδιαίτερα εύκολη και απλή, καθώς και διαθέτει ακόμα ενεργούς προγραμματιστές που τον ανανεώνουν.

Για να το χρησιμοποιήσει κάνεις, πρέπει πρώτα να το εισαγάγει (import) στο react αρχείο που θέλει να το χρησιμοποιήσει και υστέρα απλώς να δημιουργήσει ένα εξάρτημα (χωρίς αντίστοιχη ετικέτα κλεισίματος) με το όνομα το οποίο το εισήγαγε.

2.2.2 Back-End

2.2.2.1 Remix IDE (Εργαλείο/Περιβάλλον Προγραμματισμού)

Το Remix IDE είναι ένα διαδικτυακό εργαλείο (δηλ. απαιτείται πρόγραμμα περιηγητή για την χρήση του) που παρέχει την δυνατότητα στον προγραμματιστή/χρήστη να αναπτύξει έξυπνα συμβόλαια (smart contracts) καθώς και να αλληλεπίδρασή μαζί τους. Το χαρακτηριστικό που κάνει το Remix IDE ιδανικό για νέους προγράμματος είναι ότι αυτοματοποιεί αρκετές χρονοβόρες διαδικασίες (διαθέτει UI για τις συναρτήσεις, οργανώνει τις συναρτήσεις ανάλογα με την «ορατότητα» τους, διαθέτει ετοίμους λογαριασμούς, εύχρηστο terminal, κ.α.π.), επιτρέποντας στο χρήστη να συγκεντρωθεί μονάχα στην λογική των έξυπνων συμβολαίων.

2.2.2.2 Solidity (Γλωσσά Προγραμματισμού)

Η Solidity, όπως και η JavaScript, είναι μια αντικειμενοστραφή γλώσσα προγραμματισμού υψηλού επιπέδου. Πολλοί υποστηρίζουν ότι η σύνταξη της είναι παρόμοια με την JavaScript, αλλά προσωπικά ανήκω στην άλλη κατηγορία που πιστεύει ότι μοιάζει περισσότερο στη C++. Στην C++ συχνά συναντάμε ορούς όπως: “struct”, “visibility modifiers”, static typing, κ.τ.λ.

Συνεχίζοντας, με την Solidity μπορούμε να γράψουμε προγράμματα που ονομάζονται έξυπνα συμβόλαια (smart contracts), τα οποία εμφανίστηκαν για πρώτη φορά όταν δημιουργήθηκε το Ethereum Blockchain. Τα έξυπνα συμβόλαια είναι κώδικας που «ζει» στο blockchain, συνήθως περιμένουν να λάβουν κάποιου είδους ερέθισμα (είτε μέσω ενός χρήστη είτε από κάποιο άλλο έξυπνο συμβόλαιο) για να ενεργοποιηθούν και να πράξουν κάποια ενέργεια.

Η Solidity υποστηρίζει την δυνατότητα της κληρονομικότητας. Αυτό σημαίνει ότι μπορούμε να δημιουργήσουμε μια κλάση (π.χ. αμάξι) με κάποια χαρακτηριστικά (π.χ. χρώμα και μάρκα). Υστέρα, μπορούμε να δημιουργήσουμε μια υπό-κλάση (πχ. SUV) και μέσω της κληρονομικότητας μπορούμε να χρησιμοποιήσουμε τον ήδη υπάρχον κώδικα της πρώτης κλάσης.

2.2.2.3 Hardhat (Framework)

Το Hardhat είναι ένα προγραμματιστικό περιβάλλον που χρησιμοποιείται για τη δημιουργία, τη δοκιμή, την ανάπτυξη και τον εντοπισμό σφαλμάτων σε αποκεντρωμένες εφαρμογές που είναι συμβατές με την εικονική μηχανή EVM. Επιτρέπει στους προγραμματιστές να προσθέτουν λειτουργικότητα στη ροή εργασίας τους, καθώς και να διαχειρίζονται και να αυτοματοποιούν τις επαναλαμβανόμενες λειτουργίες που είναι απαραίτητες για τη δημιουργία έξυπνων συμβολαίων και dApps. Αυτό συνεπάγεται ουσιαστικά την κατασκευή, την εκτέλεση και τη δοκιμή έξυπνων συμβολαίων[78].

Ένα μικρό δίκτυο Ethereum που δημιουργήθηκε για ανάπτυξη και δοκιμές έξυπνων συμβολαίων, το Hardhat Network, περιλαμβάνεται ήδη στο Hardhat. Η λειτουργικότητά του επικεντρώνεται στον εντοπισμό σφαλμάτων Solidity και περιλαμβάνει console.log(), ίχνη στοίβας (stack traces) και ρητά μηνύματα σφάλματος για αποτυχημένες συναλλαγές[78].

Το πρόγραμμα CLI που χρησιμοποιείται για την επικοινωνία με το Hardhat, ονομάζεται Hardhat Runner, είναι ένα επεκτάσιμο πρόγραμμα εκτέλεσης εργασιών. Η δομή του βασίζεται στις ιδέες των εργασιών και των πρόσθετων. Κάθε φορά που εκκινείτε το Hardhat μέσω του CLI, εκκινείτε μια εργασία. Για παράδειγμα, η ενσωματωμένη εργασία μεταγλώττισης γίνεται μέσω npx hardhat compile. Επειδή οι εργασίες ενδέχεται να έρχονται σε επαφή μεταξύ τους, μπορούν να δημιουργηθούν εξελιγμένες ροές εργασίας. Οι υπάρχουσες εργασίες ενδέχεται να αντικατασταθούν από τους χρήστες, επιτρέποντας στους χρήστες να τροποποιούν και να επεκτείνουν τις διαδικασίες[78].

Η πλειονότητα της λειτουργικότητας του Hardhat παρέχεται μέσω προσθηκών (plug-ins) και ως προγραμματιστής, είστε ελεύθεροι να τις χρησιμοποιήσετε όπως εσείς θέλετε[78].

2.2.2.4 Ethers.js (Βιβλιοθήκη)

Η βιβλιοθήκη ethers.js είναι μια πλήρης και ελαφριά εργαλειοθήκη για διασύνδεση με το Ethereum Blockchain και τα οικοσύστημά του. Αρχικά προοριζόταν να χρησιμοποιηθεί με το ethers.io, αλλά στη συνέχεια εξελίχθηκε σε μια πιο ευέλικτη βιβλιοθήκη.

Με τη ethers.js, γράφοντας ελάχιστες γραμμές κώδικα είμαστε ικανοί να δημιουργήσουμε και να προωθήσουμε τα έξυπνα συμβόλαια μας στο blockchain. Να αλληλοεπιδράσουμε με το blockchain (καλώντας συναρτήσεις άλλων έξυπνων συμβολαίων, παίρνοντας δεδομένα από το blockchain, κ.α.π.) καθώς και με το πορτοφόλι του χρήστη (ζητώντας του να υπογράψει «ψηφιακά» ένα μνήμα, κ.τ.λ.)

2.2.2.5 Metamask (Εργαλείο)

Ένα από τα πιο δημοφιλή πορτοφόλια κρυπτονομισμάτων, το MetaMask, ενσωματώνεται στο πρόγραμμα περιήγησης και με τον ελκυστικό σχεδιασμό του που λειτουργεί ως βασικό σημείο εισόδου για το δίκτυο του Web3, της αποκεντρωμένης οικονομίας (DeFi) και των NFT.

Όπως και με άλλα πρόσθετα προγράμματος περιήγησης, το MetaMask πρέπει να εγκατασταθεί για να μπορέσει να χρησιμοποιηθεί ως πορτοφόλι Ethereum. Μετά τη φόρτωση, οι χρήστες μπορούν να χρησιμοποιήσουν οποιαδήποτε διεύθυνση Ethereum για συναλλαγές αποθηκεύοντας Ether και άλλα διακριτικά ERC-20.

Οι χρήστες μπορούν να χρησιμοποιήσουν τα χρήματά τους σε παιχνίδια, να ποντάρουν μάρκες σε εφαρμογές τζόγου και να τα πουλήσουν κρυπτονομίσματα σε αποκεντρωμένα χρηματιστήρια συνδέοντας το MetaMask με dapps που βασίζονται στο Ethereum.

Για την χρήση του MetaMask, πρέπει απλώς να πατέ στην ιστοσελίδα του και να το κατεβάσετέ (κατά την συγγραφή της εργασίας, υποστηρίζει τους 4 δημοφηλότερους περιηγητές. Chrome, Firefox, Brave, Microsoft Edge).

Όταν δημιουργήσετε ένα λογαριασμό στο MetaMask, θα σας παρουσιάσει 12 λέξεις που αποτελούν τον μόνο τρόπο που μπορείτε να ανακτήσετέ το λογαριασμό σας εάν χάσετε τον κωδικό. Συμβουλεύεται να μην αποθηκεύσετε αυτές τις 12 λέξεις σε ψηφιακή μορφή, αλλά να τις γράψετε σε ένα χαρτί και να το αποθηκεύσετε σε ασφαλές μέρος.

2.2.2.6 OpenZeppelin (Πηγαίος Κώδικας)

Μια πλατφόρμα ανοιχτού κώδικα για τη δημιουργία ασφαλών έξυπνων συμβολαίων ονομάζεται OpenZeppelin. Για την κατασκευή, τη διαχείριση και την εξέταση κάθε πτυχής της ανάπτυξης λογισμικού και των λειτουργιών για αποκεντρωμένες εφαρμογές, το OpenZeppelin προσφέρει ένα πλήρες φάσμα λύσεων ασφαλείας και υπηρεσιών ελέγχου.

Στην ουσία, είναι μια συλλογή από δοκιμασμένα έξυπνα συμβόλαια που βασίζονται στα πρότυπα του Ethereum. Το OpenZeppelin είναι κυρίως γνωστάο για τα έξυπνα συμβόλαια τους που αφορούν τα δημοφιλή κρυπτονομίσματα (ERC-20, ERC-721, ERC-1155). Στην παρούσα εργασία θα χρησιμοποιήσουμε το ERC-721 έξυπνο συμβόλαιο του OpenZeppelin.

2.3 Ανάλυση Κώδικα

Σε αυτό το υποκεφαλαίου θα αναλύσουμε τον κώδικα της αποκεντρωμένης εφαρμογής μας. Διότι όμως, ο στόχος της συγκεκριμένης εργασίας αφορά την παρουσίαση και εφαρμογή των τεχνολογιών που σχετίζομαι με το blockchain, θα επικεντρωθούμε μονάχα στο κώδικα που αφορά τα έξυπνα σύμβολα.

Αρχίζοντας με τον πηγαίο κώδικα γραμμένο σε Solidity, θα μάθουμε τα κυριά χαρακτηριστικά της γλώσσας καθώς και μερικές τεχνικές που χρησιμοποιούνται συνεχώς από τους προγραμματιστές blockchain.

Στη συνέχεια, θα εξερευνήσουμε το framework και περιβάλλον προγραμματισμού Hardhat. Θα μάθουμε τις βασικότερες λειτουργίες του Hardhat οι οποίες είναι οι ακόλουθες:

1. Μεταγλώττιση έξυπνων συμβολαίων (Smart Contract Compilation)
2. Παράταση έξυπνων συμβολαίων (Smart Contract Deployment)
3. Δημιουργία τοπικού δικτύου blockchain (Hardhat Local Network)

Για την επίτευξη των παραπάνω λειτουργιών θα είναι απαιτητή και η χρήση της γλώσσας JavaScript σε συνδυασμό με την βιβλιοθήκη ethers.js. Ακόμα, θα δούμε πως μπορούμε να χρησιμοποιήσουμε την γλώσσα JavaScript για να αυτοματοποιήσουμε μερικές ανιαρές διαδικασίες κάνοντας επίσης τον κώδικα μας πιο εύρωστο και επαναχρησιμοποιήσιμο για μελλοντικά projects.

Τέλος, για την προβολή και λήψη του κώδικα που θα αναλυθεί προσεχώς, μπορείτε να επισκεφτείτε το παρακάτω link που θα σας μεταφέρει στο **GitHub repository** της εφαρμογής. Πρέπει να αναφερθεί ότι δεν περιέχονται τα αρχεία εικόνων του UI καθώς και τα node modules λόγω του μεγάλου τους όγκου.

https://github.com/jimzord12/degree-maker

2.3.1 Έξυπνα Συμβόλαια (Solidity)

Πριν ξεκινήσουμε να αναφερόμαστε στο κώδικα είναι πολύ σημαντικό να κατανοήσουμε την λειτουργικότητα κάθε έξυπνου συμβολαίου που πρόκειται να χρησιμοποιήσουμε.

Εάν και ο βασικός μας στόχος είναι η δημιουργία μιας εφαρμογής που θα μπορεί αυτόματα να πραγματοποιεί εκδόσεις πτυχίων, εμείς θα αναπτύξουμε μια πιο ολοκληρωμένη προσέγγιση αυτού το προβλήματος.

Όπως αναφέραμε και στην αρχή του κεφαλαίου Β: 1.1 θέλουμε να υπάρχουν οι εξής λειτουργίες στα έξυπνα συμβόλαια μας:

1. Ύπαρξη επίπεδων πρόσβασης
2. Δημιουργία ολοκληρωμένων Π.Σ (Προγράμματα Σπουδών)
3. Εγγραφή φοιτητών
4. Προσθήκη βαθμολογίων
5. Αυτόματη έκδοση πτυχίου

Μπορούμε να χωρίσουμε αυτές τις λειτουργίες σε τέσσερα έξυπνα συμβόλαια.

Την **διαχείριση** των **επίπεδων πρόσβασης** θα την αναλάβει το έξυπνο συμβόλαιο εν' όμματι "**Secretery.sol**".

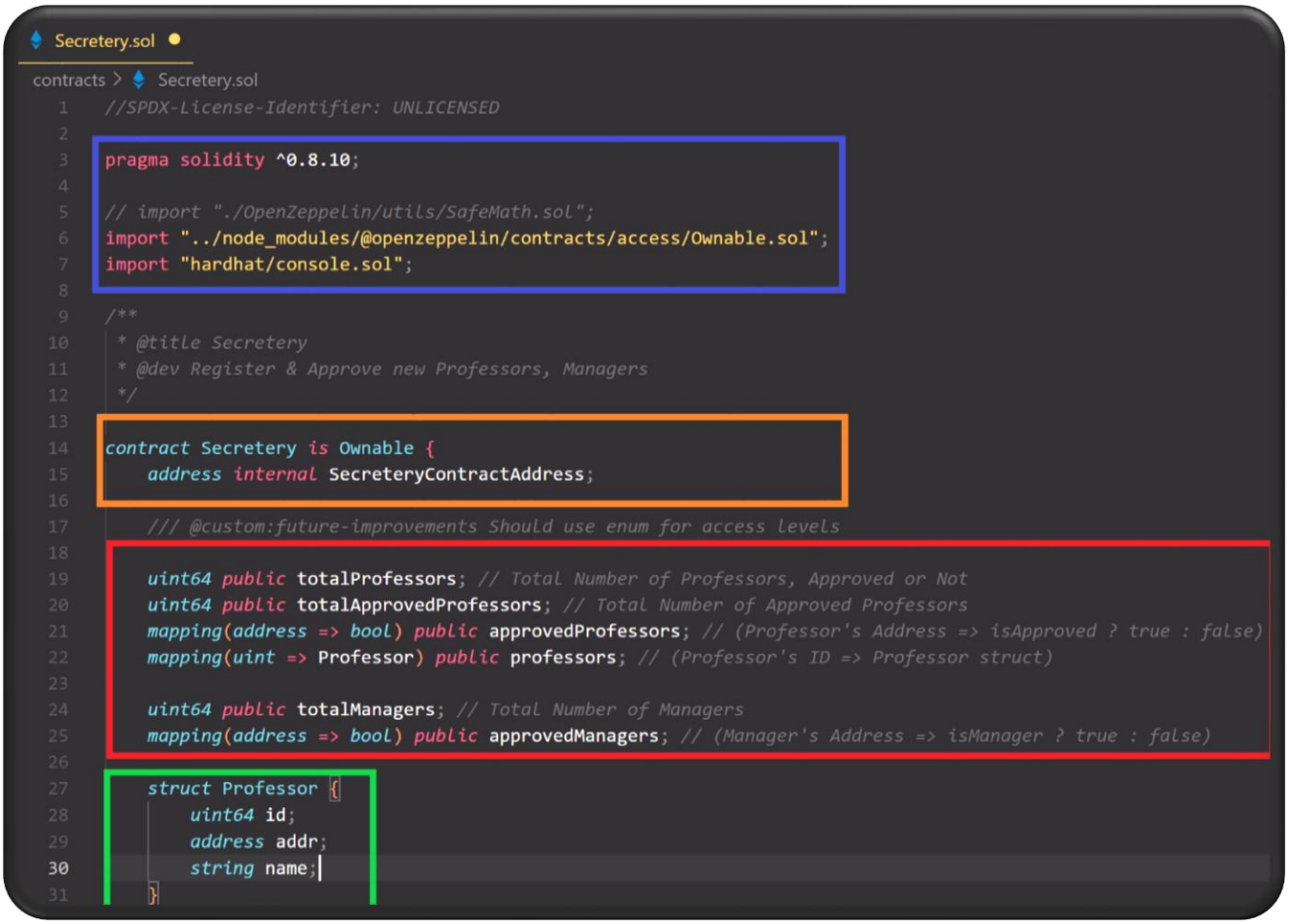
Την **δημιουργία ολοκληρωμένων Π.Σ** το "**CourseManager.sol**".

Τη **διαχείριση** των **φοιτητών** καθώς και των **βαθμολογίων** τους το "**GradesManager.sol**".

Τέλος, για ότι αφορά την **έκδοση πτυχίου** θα είναι υπεύθυνο το "**DegreeManager.sol**".

2.3.1.1 Secretery.sol

Στο Secretery.sol θα σταθούμε λίγο περισσότερο από τα υπόλοιπα, επειδή καθώς εξερευνούμε την λειτουργικότητα του παράλληλα θα μάθουμε τα βασικότερα χαρακτηριστικά της γλώσσας προγραμματισμού Solidity. Χωρίς περαιτέρω καθυστερήσεις ας εξετάσουμε τον κώδικα.



Ξεκινώντας από το **σκούρο μπλε πλαίσιο**, παρατηρούμε δυο πράγματα, το πρώτο είναι: “pragma solidity ^0.8.10” που υποδηλώνει την έκδοση της Solidity που χρησιμοποιούμε (ή πιο τεχνικά, την έκδοση του compiler που πρέπει να χρησιμοποιηθεί) στην δική μας περίπτωση υποδηλώνουμε ότι χρησιμοποιούμε την έκδοση 0.8.10 και άνω.

Το δεύτερο πράγμα που παρατηρούμε είναι δυο εισαγωγές (imports) εξωτερικών αρχείων. Η πρώτη εισαγωγή αφορά ένα άλλο έξυπνο συμβόλαιο, αναπτυγμένο από τον οργανισμό OpenZeppelin, το οποίο μέσω της κληρονομικότητας (ιδιότητα της Solidity) μας δίνει την ικανότητα να παρέχουμε δικαιώματα ιδιοκτησίας (ownership) στο λογαριασμό οποίος θα προωθήσει το έξυπνο συμβόλαιο.

Η δεύτερη εισαγωγή προέρχεται από το Hardhat, το οποίο θα μελετήσουμε αργότερα, και απλώς μας επιτρέπει να χρησιμοποιήσουμε την συνάρτηση “console.log()” (η οποία είναι η πιο συχνή εντολή που χρησιμοποιούμε στην JavaScript όταν προσπαθούμε να εντοπίσουμε προβλήματα στο κώδικα μας).

Συνεχίζοντας στο **πορτοκαλί πλαίσιο**, βλέπουμε πως δημιουργούμε ένα συμβόλαιο (παρόμοιο με τις κλάσεις σε άλλες γλώσσες προγραμματισμού, π.χ. C++). Η λέξη κλειδί “is” μετά το όνομα του συμβολαίου υποδηλώνει από ποια αλλά συμβόλαια το συγκεκριμένο συμβόλαιο κληρονομεί κώδικα. Εδώ θα τοποθετήσουμε όλο τον κώδικα που χρειαζόμαστε για την διαχείριση των επίπεδων πρόσβασης.

Στο **κόκκινο πλαίσιο**, παρατηρούμε τις αναθέσεις των παγκοσμίων μεταβλητών (global variables). Κατά κύριο λόγο, υπάρχουν δυο επίπεδα προσβασιμότητας όσον αφορά τις μεταβλητές. Τις *παγκόσμιες μεταβλητές*, οι οποίες είναι προσβάσιμες σε όλα τα μέρη του κώδικα και τις *τοπικές μεταβλητές* τις οποίες συναντάμε μέσα σε συναρτήσεις. Όταν η εκτέλεση της συνάρτησης τελειώσει, οι τοπικές μεταβλητές αυτόματα διαγράφονται. Γενικά, για λογούς ασφάλειας προτείνεται να αποφεύγεται, όσο αυτό είναι δυνατόν, η χρήση παγκοσμίων μεταβλητών.

Επιστρέφοντας στο δικό μας κώδικα, βλέπουμε ότι υπάρχουν τρεις αναθέσεις παγκοσμίων μεταβλητών και τρεις αναθέσεις χαρτογραφήσεων. Επιπλέον παρατηρούμε ότι πρέπει να προσδιορίσουμε εξ’ αρχής τον τύπο δεδομένων που θα αποθηκευτεί στην κάθε μεταβλητή, αυτό είναι ένα χαρακτηριστικό που το συναντήσαμε στις στατικές (statically-typed) γλώσσες προγραμματισμού όπως είναι η C και η C++. Ακόμα, μας δίνεται η δυνατότητα να τοποθετήσουμε “visibility modifiers” στις αναθέσεις μας.

Ας περιγράψουμε σύντομα τι είναι οι *χαρτογραφήσεις* (mappings) και οι *visibility modifiers*.

Οι *χαρτογραφήσεις* είναι μια δομή δεδομένων που μας επιτρέπει να αντιστοιχίσουμε ένα τύπο δεδομένων (π.χ. μια διεύθυνση ενός λογαριασμού, *address*) σε έναν άλλο (π.χ. μια τιμή που

μπορεί να είναι είτε αληθής είτε ψευδής, *bool*). Χρησιμοποιώντας μια τέτοια χαρτογράφηση μπορούμε πολύ ευκολά να ελέγξουμε εάν ένας χρήστης ανήκει σε κάποια συγκεκριμένη κατηγορία.

Visibility modifiers είναι κώδικας που υποδηλώνει την προσβασιμότητα σε μεταβλητές και συναρτήσεις. Η Solidity διαθέτει τέσσερα είδη τέτοιων modifier, μπορούμε να κατασκευάσουμε και δικούς μας όπως θα δούμε σε λίγο, τα οποία φαίνονται από κάτω:

1. *Public*, μπορεί να χρησιμοποιηθεί από όλους
2. *External*, μπορεί να χρησιμοποιηθεί μόνο εκτός του συμβολαίου
3. *Private*, μπορεί να χρησιμοποιηθεί μόνο από το συμβόλαιο στο οποίο εμπεριέχεται
4. *Internal*, είναι σαν την private αλλά μπορεί να χρησιμοποιηθεί και από συμβόλαια που κληρονομούν από το αυτό στο οποίο εμπεριέχεται.

Τώρα, ας δούμε πως θα χρησιμοποιηθούν αυτές τις μεταβλητές και τις χαρτογραφήσεις.

Στις μεταβλητές θα αποθηκεύουμε το συνολικό αριθμό των καταχωρημένων χρηστών. Αυτό είναι απαραίτητο διότι η χαρτογραφήσεις δεν έχουν προ-εγκατεστημένη την δυνατότητα για την ανάκτηση του αριθμό των θέσεων που έχουμε χρησιμοποιήσει.

Η οργάνωση των χαρτογραφήσεων είναι λίγο πιο περιπλοκή, οι δυο (*approvedProfessors* και *approvedManagers*) θα χρησιμοποιηθούν ως «αρχεία» για να γνωρίζουμε ανά πασά στιγμή τα δικαιώματα πρόσβασης κάθε καταχωρημένου χρήστη. Η τρίτη (*professors*) είναι πιο περιπλοκή επειδή αντιστοιχεί έναν ακέραιο χωρίς πρόσημο (unsigned integer, *uint*) σε έναν εξιδανικευμένο τύπο δεδομένων (struct) που εμείς σαν προγραμματιστές έχουμε δημιουργήσει.

Ένα struct είναι ένας εξιδανικευμένος τύπος δεδομένων, είναι ο κώδικας στο **πράσινο πλαίσιο**. Μας επιτρέπει να δημιουργήσουμε αντικείμενα τα οποία διαθέτουν χαρακτηριστικά. Στην δική μας περίπτωση, το struct που έχουμε δημιουργήσει ονομάζεται “Professor” και διαθέτει τρία χαρακτηριστικά, ένα σειριακό αριθμό (*id*), μια διεύθυνση (*addr*) και τέλος ένα όνομα (*name*).

Στην συνέχεια θα δούμε την σύνταξη μιας συνάρτησης στην γλώσσα Solidity, η οποία θα είναι υπεύθυνη για την δημιουργία αντικειμένων τύπου *Professor* και την αποθήκευση τους στην χαρτογράφηση *professors*.

```
/**
 * @dev Creates a professor obj
 * @notice Only a Manager can call this function
 * @notice 🚨 A newly created professor obj does NOT have Professor level access! 🚨
 * @param _profID = The new professor's ID, _profAddress = address, _name = professor'/name
 */
function registerProfessor(
    uint64 _profID,
    address _profAddress,
    string memory _name
) public onlyManager{
    Professor storage prof = professors[totalProfessors];
    prof.id = _profID;
    prof.addr = _profAddress;
    prof.name = _name;
    totalProfessors++;
}
```

Πιθανόν το πρώτο πράγμα που θα παρατηρούσε κάνεις είναι την περίεργη σύνταξη που έχει χρησιμοποιηθεί για την συγγραφή των σχολίων. Η Solidity υποστηρίζει μια ιδιαίτερη σύνταξη όσον αφορά τα σχόλια που ονομάζεται “NatSpec”, για περισσότερες πληροφορίες ανατρέξτε εδώ:

Η σύνταξη των συναρτήσεων στην Solidity είναι παρόμοια με την JavaScript (εάν εξαιρέσουμε τους τύπους δεδομένων και τους modifiers). Το επόμενο στοιχείο το οποίο δεν έχουμε αναφέρει ακόμα είναι οι λέξεις κλειδιά “memory” και “storage”. Αυτά τα δυο στοιχεία τα συναντάμε όταν χρησιμοποιούμε τύπους δεδομένων όπως συμβολοσειρές (string) και φωλιασμένους (nested) τύπους δεδομένων (όπως structs μέσα σε χαρτογραφήσεις). Το *memory* υποδηλώνει ότι θα αντλήσουμε δεδομένα από μνήμη υπολογιστή και όχι από το blockchain. Το *storage* υποδηλώνει το αντίστροφο.

Όπως παρατηρούμε η συνάρτηση εκτός το public modifier διαθέτει ακόμα έναν, τον “onlyManger”. Ώρα να αναφερθούμε στους εξιδανικευμένους modifiers. Παρακάτω βλέπεται τον κώδικα που απαρτίζει το onlyManager modifier.

```
/**
 * @dev Requires the one calling a function containing this modifier to be a Manager
 *   🚨 Uncomment this after testing! 🚨
 */
modifier onlyManager() {
    bool ptr = approvedManagers[msg.sender];
    require(ptr, "Secretery.sol: Only a Manager can perform this action!");
    _;
}
```

Η σύνταξη ενός modifier είναι παρόμοια με αυτή μιας συνάρτησης με την διαφορά της λέξης κλειδί “modifier” αντί για “function” και της κάτω παύλας που πρέπει να υπάρχει. Η κάτω παύλα χρησιμοποιείται για να ξέρει η Solidity εάν θα πρέπει να τρέξει τον κώδικα του modifier πριν ή μετά την εκτέλεση του κώδικα της συνάρτησης στην οποία ανήκει.

Ο συγκεκριμένος modifier είναι πολύ απλός, χρησιμοποιείται για να αποτρέψει χρήστες οι οποίοι δεν έχουν δικαιώματα manager να καλέσουν συναρτήσεις που διαθέτουν το modifier *onlyManager*.

Αρχικά χρησιμοποιεί την τιμή “msg.sender”, η διεύθυνση της οντότητας που καλεί την συνάρτηση, για να δει ποια τιμή της συγκεκριμένης διεύθυνσης αντιστοιχεί στην χαρτογράφηση *approvedManagers* (αυτή μπορεί είναι true (αληθής) ή false (ψευδής). Στην συνέχεια αποθηκεύει την τιμή στην τοπική μεταβλητή “ptr”, με σκοπό να την εισάγει στην συνάρτηση “require()” η οποία είναι προ-εγκατεστημένη στη Solidity. Η require() απαιτητεί τουλάχιστον ένα επιχείρημα αλλά προτείνεται η χρήση και των δυο. Το πρώτο πρέπει να είναι τύπου *bool*, και το δεύτερο τύπου *string*. Εάν το πρώτο επιχείρημα έχει αληθής τιμή η require() θα επιτρέψει την εκτέλεση του κώδικα που ακολουθεί, εάν όμως είναι ψευδής θα σταματήσει την εκτέλεση και θα “πετάξει” σφάλμα στο οποίο θα εμπεριέχει το string που έχουμε τοποθετήσει ως δεύτερο επιχείρημα της require().

Συνεχίζοντας με την εξερεύνηση μας, συναντάμε την συνάρτηση “approveProfessor()” η οποία, όπως μας αποκαλύπτει και το όνομα, δίνει σε ένα καθηγητή τα δικαιώματα πρόσβασης ενός καθηγητή. Έχετε στο νου σας ότι όταν καταχωρούμε έναν καθηγητή αυτός δεν αποκτά αυτόματα δικαιώματα πρόσβασης ενός καθηγητή.

```
/**
 * @dev Gives Professor level access
 * @notice Only a manager can call this function
 * @param _profID = Professor's ID
 */
function approveProfessor(
    uint64 _profID
) public onlyManager {
    for (uint i = 0; i <= totalProfessors; i++) {
        Professor storage prof = professors[i];
        if (prof.id == _profID) {
            approvedProfessors[prof.addr] = true;
            totalApprovedProfessors++;
        }
    }
}
```

Το ενδιαφέρον κομμάτι αυτής της συνάρτησης είναι η χρήση του βρόχου “for”. Η σύνταξη της *for* είναι ακριβώς πανομοιότυπη με αυτήν στην C++ και μπορεί να χρησιμοποιηθεί και στην JavaScript (εάν και η JS διαθέτει πιο συγχρόνους μεθόδους επανάληψης). Στην δική μας περίπτωση, την χρησιμοποιούμε για να αντλήσουμε το χαρακτηριστικό *id*, εάν αυτό υπάρχει, από κάθε αντικείμενο *Professor* μέσα στην χαρτογράφηση *professors*, με σκοπό να επιβεβαιώσουμε ότι ο καθηγητής στον οποίο επιθυμούμε να δώσουμε τα ανάλογα δικαιώματα είναι καταχωρημένος στο blockchain.

Όπως αναφέραμε και προηγουμένως, ο **κυρίως σκοπός της παρούσας εργασίας** είναι η **επίδειξη των τεχνολογιών blockchain**. Θεωρητικά, εάν αυτή η εφαρμογή προοριζόταν για πραγματική χρήση, θα έπρεπε να κατασκευάσουμε ένα περίπλοκο *user interface* το οποίο θα έπρεπε στον χρήστη να καλεί όλες τις συναρτήσεις του έξυπνου συμβολαίου μας. Για να το αποφύγουμε αυτό, θα κατασκευάσουμε μια συνάρτηση η οποία θα χρησιμοποιήσει εικονικά δεδομένα (dummy/test data) για καλέσει όλες τις συναρτήσεις που δημιουργήσαμε ώστε αργότερα να μπορούμε να ελέγξουμε εάν ο κώδικας κάνει πραγματικά αυτό που επιθυμούμε, κάτι που σχεδόν ποτέ δεν συμβαίνει με την πρώτη.

```
    function hardCodeForTesting() public {
        appointManager(0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2); // No.2 Account Remix-IDE
        appointManager(0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db); // No.3 Account Remix-IDE
        appointManager(0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB); // No.4 Account Remix-IDE
        appointManager(0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266); // No.1 Account Hard Hat

        registerProfessor(
            422,
            0x5B38Da6a701c568545dCfcB03FcB875f56beddC4,
            "Koulis"
        ); // No.1 Account
        registerProfessor(
            13,
            0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2,
            "Giannis"
        ); // No.2 Account
        registerProfessor(
            5571,
            0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db,
            "Xristos"
        ); // No.3 Account
        registerProfessor(
            31337,
            0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266,
            "Hard_Hat_#1"
        ); //No.1 Account Hard Hat

        approveProfessor(422);
        approveProfessor(13);
        approveProfessor(31337); //No.1 Account Hard Hat
    }
}
```

Οι τρεις πρωτοι λογαριασμοι που εχουν χρησιμοποιηθει προερχονται από το Remix IDE, ενώ ο τεταρτος από το Hardhat. Όπως φαινεται, δινουμε σε ολους δικαιωματα manager. Στην συνεχεια, επιλεγουμε τους υπογραμισμενους λογαριασμους συν έναν ακομα από το Remix IDE και τους καταχωρουμε ως καθηγητες. Τελος, παραδιδουμε δικαιωματα καθηγητη στους τρεις από τους τεσσερεις. Με αυτό τον τροπο εχουμε την δυνατοτητα τα πραγματοποιησουμε πολλαπλα τεστς για ειμαστε βεβαιει ότι όλα λειτουργουν όπως είναι προορισμενα

Τέλος, πριν προχωρήσουμε στην ανάλυση των υπολοίπων έξυπνων συμβολαίων πρέπει να αναφέρουμε μια πολύ σημαντική και ιδιαίτερη συνάρτηση, τον “constructor”. Μπορεί να υπάρχει μονάχα ένας *constructor* ανά *contract* και η λειτουργία του είναι να εκτελέσει τον κώδικα που βρίσκεται μέσα του αμέσως μόλις το συμβόλαιο προωθηθεί (deployed).

Ο *constructor* μπορεί επίσης να έχει παραμέτρους, οι τιμές των οποίων εισάγονται στο κώδικα JavaScript της βιβλιοθήκης ethers.js που προωθεί το έξυπνο συμβόλαια όπως θα δούμε πιο μετά.

Ας εξετάσουμε τον *constructor* του *Secretery.sol*.

```
/**
 * @dev Upon deployment, saves the contract's address to a global var
 */
constructor() {
    hardCodeForTesting();
    SecreteryContractAddress = address(this);
}
```

Στην συγκεκριμένη περίπτωση, το μόνο που πράττει ο constructor είναι να καλέσει την συνάρτηση “hardCodeForTesting()” και στην συνέχεια να αποθηκεύσει σε μια παγκόσμια μεταβλητή την διεύθυνση του έξυπνο συμβολαίου μόλις αυτό προωθηθεί.

2.3.1.2 CourseManager.sol

Το συγκεκριμένο έξυπνο συμβόλαιο περιέχει τον περισσότερο κώδικα και είναι υπεύθυνο για την δημιουργία και διαχείριση ολοκληρωμένων προγραμμάτων σπουδών (Π.Σ).

Ας αρχίσουμε από την αρχή του κώδικα, δηλαδή το κομμάτι που εισάγουμε εξωτερικά αρχεία και δηλώνουμε τις παγκόσμιες μεταβλητές μας.

contracts > CoursesManager.sol

```
1  // SPDX-License-Identifier: UNLICENSED
2
3  pragma solidity ^0.8.10;
4
5  import "../node_modules/@openzeppelin/contracts/access/Ownable.sol";
6  import "./Secretery.sol";
7  import "hardhat/console.sol";
8
9  /**
10  * @title Courses Manager
11  * @dev Creates & Manages curriculums
12  */
13 contract CoursesManager is Ownable {
14     address public SecreteryContractAddress_T;
15     uint8 public currentNumOfCourses;
16     uint64 public totalPS;
17     mapping(uint64 => ProgrammaSpoudwn) public programmataSpoudwn;
18
```

Παρατηρούμε ότι υπάρχουν τρεις εισαγωγές εξωτερικών αρχείων. Όπως και στο Secretery.sol έχουμε το συμβόλαιο *Ownable.sol* από τους *OpenZeppelin* και το *console.sol* από το *Hardhat*. Ακόμα παρατηρούμε ότι υπάρχει και το συμβόλαιο που μόλις εξετάσαμε, το *Secretery.sol*, το οποίο θα χρειαστούμε για να ελέγχουμε εάν ένας χρήστης έχει δικαιώματα καθηγητή ή όχι.

Προχωρώντας παρακάτω, υπάρχουν τρεις μεταβλητές και μια χαρτογράφηση.

Η πρώτη μεταβλητή τύπου *address* θα χρησιμοποιηθεί για την αποθήκευση της διεύθυνσης του συμβολαίου Secretery.sol.

Η δεύτερη είναι τύπου *uint8* (το 8 υποδηλώνει ότι θα δεσμεύσει 8-bits ή ένα byte, αυτό μας περιορίζει σε αριθμούς από 0-255, διότι $2^8 = 256$) και χρησιμοποιείται για λογούς εξοικονόμησης αποθηκευτικού χώρου, διότι ο αποθηκευτικός χώρος σε ένα blockchain είναι συνήθως πολύτιμος και περιορισμένος. Όσον αφορά την χρησιμότητα της μεταβλητής, υπάρχουν συναρτήσεις στο κώδικα για εύρεση φωλιασμένων δεδομένων οπού η γνώση του συνόλου των καταχωρημένων μαθήματων είναι απαραίτητη.

Η τρίτη είναι τύπου uint64 (δέχεται αριθμούς από 0 έως 18.446.744.073.709.551.615 ή $2^{64} – 1$) και θα χρησιμοποιηθεί για την αποθήκευση του συνόλου των καταχωρημένων Π.Σ, ώστε αργότερα να μπορούμε να τα προσπελάσουμε .

Τέλος, η χαρτογράφηση *programmataSpoudwn* χρησιμοποιείται για την αποθήκευση των όλων των καταχωρημένων Π.Σ.

Για να δημιουργήσουμε ολοκληρωμένα Π.Σ, θα χρειαζόμαστε τα ακόλουθα:

1. Εάν **struct** για να μπορούμε να δημιουργούμε αντικείμενα Π.Σ. (*ProgrammaSpoudwn*)
2. Εάν **struct** για τα εξάμηνα του κάθε Π.Σ. (*Semester*)
3. Εάν **struct** για τα μαθήματα του κάθε εξάμηνου. (*Course*)

Τώρα θα δούμε μια τεχνική που μας επιτρέπει να φωλιάσουμε ένα struct μέσα σένα άλλο δημιουργώντας με αυτόν τον τρόπο μια πιο περιπλοκή δομή δεδομένων.

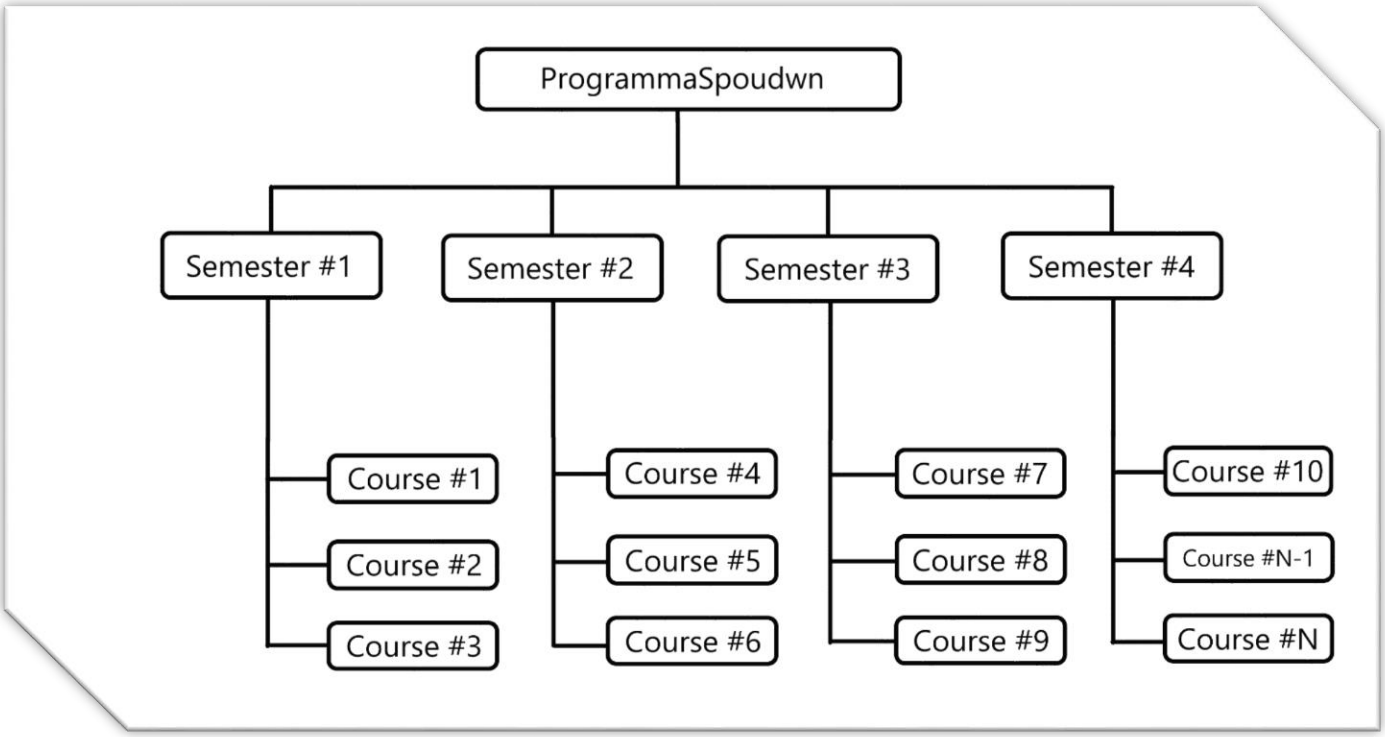
Για να το επιτύχουμε αυτό πρέπει να δηλώσουμε μια χαρτογράφηση σαν χαρακτηριστικό του struct. Με αυτό τον τρόπο μπορούμε να αποθηκεύσουμε structs μέσα structs. Ακολουθούν δυο εικόνες μια με τον κώδικα και μια που απεικονίζει τα επίπεδα φωλιάσουν των structs που θα χρησιμοποιήσουμε.

Έχοντας ήδη αναφέρει πως λειτουργούν τα structs, στο Secretery.sol, δεν θα εμβαθύνω περισσότερο. Παρατηρώντας τα σχόλια διπλά από τα χαρακτηρίστηκα των structs μπορείτε να κατανοήσετέ την χρήση τους.

```
struct ProgrammaSpoudwn {
    uint64 id;
    uint8 maxCoursesNum; // All the Courses this program has to offer (🤣 Should make a getter() for this)
    uint8 maxSemesters; // Max number of Semesters this program has
    uint8 maxCourseTypes; // Max number of Course Types (Ex. Y=0, EY=1, E=2, etc...)
    mapping(uint8 => Semester) semesters; // (Semester_ID => Corresponding Semester)
}

struct Semester {
    uint8 id;
    uint8 CoursesPerSem; // Ex. 1st Semester has 6 Courses, 8th has 4 Courses, etc...
    mapping(uint8 => Course) courses; // (Course ID (1001) => Course struct)
    mapping(uint8 => uint8) requirements; // (Courses Type ( Y=0, EY=1 ) => How many of these must be passed (3x Y, 3x EY))
}

struct Course {
    uint32 id;
    uint8 semester;
    int8 type_; // For example, Y=0 || EY=1 || E=2
    string title;
}
```



Συνεχίζοντας την ανάλυση του κώδικα, ερχόμαστε σε επαφή με τον modifier *onlyProf_T*. Εδώ το ενδιαφέρον κομμάτι είναι οι πρώτες δυο γραμμές κώδικα μέσα στο σώμα του modifier. Η πρώτη μας επιτρέπει να πραγματοποιήσουμε μια σύνδεση με προηγούμενο έξυπνο συμβόλαιο (Secretery.sol) ενώ η δεύτερη μας δίνει την δυνατότητα να καλέσουμε μια συνάρτηση από το Secretery.sol.

Επίσης βλέπουμε πως μπορούμε να εισάγουμε παραμέτρους στους modifiers.

```
/**
 * @dev Checks if an address has professor level access
 * @dev Mainly used from other contracts for access validation
 * @param _caller the account's address in question
 */
function isApprovedProf(address _caller) public view returns (bool) {
    console.log("The param: ", _caller);
    console.log(
        "Is an approved Professor?  Answer: ",
        approvedProfessors[_caller]
    );
    return approvedProfessors[_caller];
}
```

Από Secretery.sol

```
// -- Modifiers

modifier onlyProf_T(address _caller) {
    Secretery secreteryContract_T = Secretery(SecreteryContractAddress_T);
    bool isProf = secreteryContract_T.isApprovedProf(_caller);
    require(isProf, "mod_onlyProf: Caller is not a professor!");
    console.log("[CM] Deployer - Msg.Sender", msg.sender);
    _;
}
```

Από CourseManager.sol

Το επόμενο κομμάτι κώδικα παρέχει την δυνατότατά **δημιουργίας**, **επεξεργασίας** και **διαγράφης** ενός Π.Σ.

Επειδή ο κώδικας που πραγματοποιεί τις ιδίες λειτουργίες για τα εξάμηνα και τα μαθήματα είναι σχεδόν πανομοιότυπος θα παραλειφθεί.

```
// -- PS - Functionality --

// @FutureUpdate: Make a getter() for PS 💡

function createPS(
    uint64 _PS,
    uint8 _maxSemesters,
    uint8 _maxCoursesNum,
    uint8 _maxCourseTypes
) public onlyProf_T(msg.sender) {
    ProgrammaSpoudwn storage ps = programmataSpoudwn[_PS];
    ps.id = _PS;
    ps.maxSemesters = _maxSemesters;
    ps.maxCoursesNum = _maxCoursesNum;
    ps.maxCourseTypes = _maxCourseTypes;
    totalPS++;
}

function modifyPS(
    uint64 old_PS,
    uint64 new_PS,
    uint8 _maxSemesters,
    uint8 _maxCoursesNum,
    uint8 _maxCourseTypes
) public onlyProf_T(msg.sender) {
    // @FutureUpdate: Check if new props already exist 💡
    ProgrammaSpoudwn storage ps = programmataSpoudwn[old_PS];
    ps.id = new_PS;
    ps.maxSemesters = _maxSemesters;
    ps.maxCoursesNum = _maxCoursesNum;
    ps.maxCourseTypes = _maxCourseTypes;
}

function deltePS(uint64 _PS) public onlyProf_T(msg.sender) {
    // @FutureUpdate: Check if PS exists 💡
    delete programmataSpoudwn[_PS];
    totalPS--;
}
```

Πριν συνεχίσουμε με την ανάλυση του επομένου έξυπνου συμβολαίου, αξίζει να εξετάσουμε την συνάρτηση η οποία θέτει τις προϋποθέσεις που θα έχει κάθε εξάμηνο, την setSemRequirements().

```
function setSemRequirements(
    uint64 _PS,
    uint8 _semID,
    uint8[] memory _numOfEachCourseType
) public onlyProf_T(msg.sender) {
    ProgrammaSpoudwn storage ps = programmataSpoudwn[_PS];  // Ex. PS: #0
    Semester storage sem = ps.semesters[_semID];            // Ex. Semester: #1
    require(
        _numOfEachCourseType.length <= ps.maxCourseTypes,
        "CourseManager.sol: You set more Course Types that you should!"
    );
    for (uint8 i = 0; i < ps.maxCourseTypes; i++) {
        sem.requirements[i] = _numOfEachCourseType[i];
        // numOfEachCourseType: [6, 0 ,0]
        // *We have 6 Courses with Type:  Y=0
        // *We have 0 Courses with Type: EY=1
        // *We have 0 Course with Type:  E=2
        // numOfEachCourseType: [1, 4, 1]
        // *We have 1 Course with Type:  Y=0
        // *We have 4 Courses with Type: EY=1
        // *We have 1 Course with Type:  E=2
    }
}
```

Το ενδιαφέρον στοιχείο είναι η χρήση μιας δομής δεδομένων που δεν έχουμε αναφέρει έως τώρα, την παράταξη *(array)*. Η παράταξη είναι μια πιο απλή μορφή της χαρτογράφησης, διότι εμείς απλώς προσθέτουμε μέσα της δεδομένα χωρίς την ανάγκη για αντιστοίχιση.

Στη Solidity υπάρχουν 2 βασικά είδη παρατάξεων, οι *στατικές* και *δυναμικές*. Στη **στατική** πρέπει να δηλώσουμε εξαρχής το μέγεθος που πρέπει να έχει ή ποσά στοιχεία θα αποθηκεύσουμε σε αυτήν. Ενώ στη **δυναμική**, όσο προσθέτουμε στοιχεία αυτή μεγαλώνει σε μέγεθος αυτόματα.

Το ιδανικό θα ήταν να μπορούσαμε να χρησιμοποιούμε μόνο τις δυναμικές ώστε να κάνουμε την ζωή μας πιο εύκολη, αλλά δυστυχώς εάν θέλουμε να δημιουργήσουμε μια παράταξη μέσα σε μια συνάρτηση αυτή θα πρέπει να είναι στατική.

Στην δική μας περίπτωση, χρησιμοποιούμε μια παράταξη ως παράμετρο συνάρτησης. Αυτή η παράταξη θα περιέχει αριθμητικές τιμές οι οποίες αντιπροσωπεύουν το σύνολο των μαθήματων που πρέπει να περάσει ένας φοιτητής για να ολοκληρώσει το εξάμηνο, ενώ επίσης χρησιμοποιούμε και την αυτόματη αρίθμηση, ιδιότητα που διαθέτουν όλες οι παρατάξεις, για να αντιστοιχίσουμε αυτό το σύνολο με το *τύπο των μαθήματων*. Στο τέλος του κώδικα της συνάρτησης υπάρχουν δυο παραδείγματα , στην μορφή σχολίων, για να μπορέσετε να κατανοήσετε τη συγκεκριμένη λογική καλυτέρα.

2.3.1.3 GradesManager.sol

Στο παρόν έξυπνο συμβόλαιο θα ξεκινήσουμε μελετώντας μια νέα τεχνική η οποία θα μας επιτρέπει να δημιουργήσουμε, για άλλη μια φορά, μια πιο περιπλοκή δομή δεδομένων. Έπειτα, θα δούμε την λειτουργία της συνάρτησης που ελέγχει εάν ένας φοιτητής έχει ολοκληρώσει τις προϋποθέσεις ενός εξάμηνου. Τέλος, θα χρησιμοποιήσουμε την βασική ιδέα του προτύπου ERC-1155 για τη δημιουργία της συνάρτησης η οποία θα ανεβάζει τους βαθμούς στο blockchain.

Η νέα τεχνική είναι η τοποθέτηση μιας χαρτογράφησης μέσα σε μια άλλη χαρτογράφηση (*double-mapping*). Στην δική μας περίπτωση αυτή η τεχνική θα μας βοηθήσει κατά τον έλεγχο των προϋποθέσεων για την ολοκλήρωση ενός εξάμηνου. Ακολουθεί ο κώδικας που κάνει χρήση αυτής της τεχνικής.

```
struct Student {
    address addr;
    uint am;
    uint64 ps;
    bool canGraduate;
    mapping(uint8 => mapping(uint8 => PassedCourse[])) semesters; // (Semester ID => Course Type => Array of Passed Courses of that Type
}
```

Όπως παρατηρούμε χρησιμοποιούμε το double-mapping μέσα στο struct *Student*. Αυτή η τεχνική μας επιτρέπει να έχουμε πρόσβαση σε όλα τα περασμένα μαθήματα ενός φοιτητή ανά τύπο μαθήματος και ανά εξάμηνο. Αυτή η οργάνωση δεδομένων κάνει τον έλεγχο των προϋποθέσεων αρκετά εύκολο, διότι πρέπει απλώς να γνωρίζουμε το σύνολο των μαθήματων που πρέπει να περαστούν ανά τύπο μαθήματος σε ένα συγκεκριμένο εξάμηνο (πληροφορία που περνούμε από το *CourseManager.sol*) .

Το επόμενο θέμα που θα αναλύσουμε είναι η συνάρτηση που ελέγχει εάν ο φοιτητής έχει ολοκληρώσει ένα εξάμηνο και ονομάζεται *hasPassedAllCourses()*. Ο κύριος σκοπός της είναι να παρέχει δεδομένα στο τελευταίο έξυπνο συμβόλαιο, το *DegreeManager.sol*. Για να την καλέσουμε πρέπει να εισάγουμε τα εξής επιχειρήματα:

1. Τον ΑΜ του φοιτητή
2. Το ΠΣ στο οποίο υπάγεται
3. Το ID του εξάμηνου που θέλουμε να ελέγξουμε
4. Μια παράταξη που περιέχει τις προϋποθέσεις του εξάμηνου
5. Το συνολικό αριθμό των προϋποθέσεων

Η συνάρτηση ξεκινάει με το να ελέγχει εάν αρχικά ο μαθητής υπάγεται στο συγκεκριμένο Π.Σ. Στη συνέχεια, δηλώνεται μια τοπική μεταβλητή που χρησιμοποιείται σαν μετρητής που είναι απαραίτητη για την λογική που υπάρχει στο βρόχος *for* οποίος πραγματοποιεί τον έλεγχο των προϋποθέσεων. Τέλος, ελέγχεται εάν ο μετρητής είναι ίσως με το μήκος της παράταξης *_CoursesTypeNum* που υποδηλώνει την επιτυχής ολοκλήρωση του εξάμηνου.

Τέλος, θα εξετάσουμε και την συνάρτηση *uploadGrades()*, που επιτρέπει στους καθηγητές να ανεβάσουν τους βαθμούς των φοιτητών στο blockchain.

Ας ξεκινήσουμε με την λογική της συνάρτησης, ο καθηγητής πρέπει να εισαγάγει:

1. Το ID του μαθήματος για το οποίο σκοπεύει να ανεβάσει τις βαθμολογίες
2. Το εξάμηνο στο οποίο ανήκει το μάθημα
3. Τον τύπο του μαθήματος (π.χ. 0 για Υποχρεωτικό, 1 για Επιλογής, κτλ.)
4. Μια παράταξη που θα περιέχει τους αριθμούς μητρώου (ΑΜ) των φοιτητών
5. Μια παράταξη που θα περιέχει τους βαθμούς των φοιτητών

Πρέπει να επισημάνουμε ότι **η σειρά** με την οποία είναι τοποθετημένα τα στοιχεία σε αυτές τις δυο παρατάξεις **είναι μέγιστης σημασίας**, διότι χρησιμοποιούμε την ιδιότητα της αυτόματης αρίθμησης που διαθέτουν οι παρατάξεις ώστε να αντιστοιχίσουμε τους ΑΜ με τους βαθμούς.

Στην συνέχεια ας αναλύσουμε τους δυο ελέγχους. Ο πρώτος ελέγχει εάν υπάρχουν στοιχεία μέσα στην παράταξη, ενώ ο δεύτερος ελέγχει εάν οι δυο παρατάξεις έχουν το ίδιο μήκος ή μέγεθος εάν θέλετε. Αυτό είναι πολύ σημαντικό γιατί εάν δεν έχουν το ίδιο μέγεθος σιγουρά ο καθηγητής έχει κάνει κάποιο λάθος.

Ο βρόχος *for* χρησιμοποιείται για την προσπέλαση των δυο παρατάξεων, οπού περνούν τα δεδομένα ένα-ένα και τα προσθέτουν στο *double-mapping* του κάθε φοιτητή.

```
/**
 * @dev 1) Creating Storage Pointer to access the particular Student in the students Mapping
 * @dev 2) We grab the Grade for that Student (cuz of the identical array indexs its fairly easy).
 * @dev 3) We access the semesters Mapping that lives inside the specific Student,
 *      we use the arg: _courseSemester to find the correct index,
 *      we push a new PassedCourse struct inside the subArray of the semesters Mapping.
 * @notice Indexing is CRUCIAL for this function to work as intended!
 * @notice Only a approved professor can call this function!
 * @param _courseID is the Course's ID
 * @param _courseSemester is the Semester's ID
 * @param _courseType is the Course's type
 * @param _studentsAM is an array containing the Students' IDs
 * @param _studentGrades is an array containing the Students' grades
 * @custom:future-improvement Inside for-loop check if another PassedCourse struct exists with the same _courseID
 */
function uploadGrades(
    uint32 _courseID,
    uint8 _courseSemester,
    uint8 _courseType,
    uint256[] memory _studentsAM,
    uint8[] memory _studentGrades
) public onlyProf {
    require(
        _studentsAM.length > 0,
        "fromFun_uploadGrades: No Addresses have been inputed!"
    );
    require(
        _studentsAM.length == _studentGrades.length,
        "fromFun_uploadGrades: Studebt ID's and the amount of grades are not equal!"
    );

    for (uint32 i = 0; i < _studentsAM.length; i++) {
        Student storage currentStudent = students[_studentsAM[i]]; // 1
        uint8 currentGrade = _studentGrades[i]; // 2
        currentStudent.semesters[_courseSemester][_courseType].push(
            PassedCourse(
                _courseID,
                _courseSemester,
                currentGrade,
                _courseType
            )
        );
    }
}
```

2.3.1.4 DegreeManager.sol

Φτάσαμε, επιτέλους, στο τελευταίο έξυπνο συμβόλαιο της εφαρμογής μας. Το συγκεκριμένο χρησιμοποιεί όλες τις πληροφορίες που υπάρχουν στα προηγούμενα συμβόλαια, με σκοπό την δημιουργία NFTs (ERC-721) που θα αντιπροσωπεύουν τα πτυχία των φοιτητών.

Ας ξεκινήσουμε από τις εισαγωγές των εξωτερικών αρχείων.

```
pragma solidity ^0.8.10;

import "../node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "../node_modules/@openzeppelin/contracts/access/Ownable.sol";
import "../node_modules/@openzeppelin/contracts/utils/Counters.sol";
import "./My_utils/Course_Essentials.sol";
import "./My_utils/Grades_Essentials.sol";

contract DegreeManager is GradesEssentials, CourseEssentials, ERC721 {

    using Counters for Counters.Counter;
//  using SafeMath for uint;
    Counters.Counter private _tokenIdCounter;
    mapping(uint => bool) graduableStudents; // (AM => CanGraduate?)

    constructor(address _CMaddr, address _GMaddr) ERC721("DegreeToken", "DTN") {
        setCMaddress(_CMaddr);
        setGMaddress(_GMaddr);
    }
```

Αμέσως παρατηρούμε ότι αυτό το συμβόλαιο διαθέτει τις περισσότερες εισαγωγές εξωτερικών αρχείων. Η σημαντικότερη όλων, είναι φυσικά το έξυπνο συμβόλαιο *ERC721.sol*, όπως πάντα αναπτυγμένο από τους OpenZeppelin, το οποίο μας παρέχει όλη την λειτουργικότητα που θα χρειαστούμε για κατασκευάσουμε τα NFTs μας γράφοντας μονάχα ελάχιστες γραμμές κώδικα.

Το συμβόλαιο *Counters.sol*, είναι μια βιβλιοθήκη. Με λίγα λογία, είναι κώδικας που προορίζεται για να μας παρέχει εξτρά λειτουργικότητα. Μια άλλη διάσημη βιβλιοθήκη από τους OpenZeppelin είναι η *SafeMath.sol*, η οποία δεν επιτρέπει στις μεταβλητές μας να πάθουν *overflow* (συμβαίνει όταν ξεπεραστεί ο μέγιστος επιτρεπτός αριθμός, π.χ. εάν μια μεταβλητή

uint8 έχει τιμή 255 και εμείς προσθέσουμε ακόμα 1, αντί για 256 θα γίνει 0) ή *underflow* (είναι το ανάποδο, εάν η μεταβλητή είναι 0 και αφαιρέσουμε 1, γίνεται 255).

Οι τελευταίες δυο εισαγωγές είναι αφηρημένα συμβόλαια (abstract contracts) τα οποία είναι συμβόλαια που απλώς παρέχουν λειτουργικότητα σε αλλά συμβόλαια. Στην δική μας περίπτωση χρησιμοποιούνται για να αποφύγουμε την άμεση κληρονομικότητα (διότι αυτό θα «βάραινε» αρκετά το παρόν συμβόλαιο), παίρνοντας μονάχα ότι χρειαζόμαστε από τα προηγούμενα συμβόλαια.

Παρατηρούμε επίσης την περίπτωση οπού ο constructor περιέχει παραμέτρους, όπως ακόμα και το πως μπορούμε να καλέσουμε τον constructor ενός αλλού συμβολαίου. Ο κώδικας που καλεί τον constructor του ERC721.sol είναι ο παρακάτω:

ERC721 (“DegreeToken”, “DTN”)

Το πρώτο επιχείρημα υποδηλώνει το όνομα του token που θέλουμε να δημιουργήσουμε, ενώ το δεύτερο το σύμβολο (symbol) του.

Στο σώμα του constructor χρησιμοποιούμε δυο συναρτήσεις (τις οποίες περνούμε από τα αφηρημένα συμβόλαια) για να αποθηκεύσουμε τις διευθύνσεις των προηγουμένων συμβολαίων ώστε να μπορούμε να αντλήσουμε δεδομένα από αυτά αργότερα.

Ακολουθεί η κυριά συνάρτηση του έξυπνου συμβολαίου, *η checkIfStudentCanGraduate()*, η οποία θα πραγματοποιήσει τον έλεγχο των προϋποθέσεων για το εάν ένας φοιτητής μπορεί να εκδώσει το πτυχίο του. Στην περίπτωση που μπορεί, θα αποθηκεύσει τον ΑΜ του φοιτητή σε μια χαρτογράφηση με σκοπό να χρησιμοποιηθεί αργότερα από την συνάρτηση που θα δημιουργεί το αντιπροσωπευτικό NFT του πτυχίου.

Στην αρχή της συνάρτησης παρατηρούμε ότι έχοντας μόνο το ΑΜ σαν δεδομένο εισόδου, μπορούμε να πάμε στα προηγούμενα συμβόλαια και να αντλήσουμε όλες τις πληροφορίες που χρειαζόμαστε. Στην συνέχεια, χρησιμοποιούμε την συνάρτηση *hasPassedAllCourses()*, παρμένη από το GradesManager.sol, η οποία βρίσκεται μέσα σένα βρόχο for. Εάν θυμάστε, αυτή η συνάρτηση ελέγχει εάν ο φοιτητής έχει ολοκληρώσει τις προϋποθέσεις ενός εξάμηνου, εμείς όμως θέλουμε να ελέγξουμε όλα τα εξάμηνα, και για να το πετύχουμε αυτό πρέπει να την χρησιμοποιήσουμε επανειλημμένα. Στο τέλος της συνάρτησης υπάρχει μια συνθήκη που

ελέγχει εάν τελικά ο φοιτητής έχει ολοκληρώσει επιτυχώς όλα τα εξάμηνα και εάν αποδειχθεί αληθής θα τοποθετήσει το ΑΜ του στην χαρτογράφηση *graduableStundents*.

```
// ✓ Phase: Complete ✓
function checkIfStudentCanGraduate(
    uint _am
) public
returns (bool result) {
    uint64 studentPS = _getStudentPS(_am); // From: GradesEssentials Contract
    uint8 maxSemesters = _getSems(studentPS); // From: CoursesEssentials Contract
    uint8 maxCourseTypes = _getMaxCourseTypes(studentPS); // From: CoursesEssentials Contract
    require(
        maxSemesters != 0,
        "DegreeManager.sol: MaxSemester == 0, however it shouldn't!"
    );
    uint8 successCounter = 0;
    for (uint8 i = 0; i < maxSemesters; i++) {
        uint8[] memory coursesReq = _getRequirementsFormSem(studentPS, i); // Array of Requirements, Index = Type, Value = Amount
        // uint8 coursesPerSem = _getCourses(studentPS, i);                    // From: CoursesEssentials Contract
        if (
            hasPassedAllCourses(
                _am,
                studentPS,
                i,
                coursesReq,
                maxCourseTypes
            )
        )
            // Checks if all requierd Courses has been passed, in a Specific Sem
            successCounter++;
    }
    if (successCounter == maxSemesters) {
        // If all requierd Courses has been passed in every Semester
        graduableStudents[_am] = true;
    }
    result = graduableStudents[_am]; // returns the result from accesing a global mapping
}
```

Τώρα ήρθε η στιγμή να χρησιμοποιήσουμε την λειτουργικότητα του ERC721.sol για την δημιουργία των NFTs. Πρέπει να ευχαριστήσουμε τους OpenZeppelin διότι, με χρήση ελάχιστων γραμμών κώδικα μπορούμε με ασφάλεια να κατασκευάσουμε τα ERC-721 tokens που χρειαζόμαστε.

Βλέπουμε ότι απαιτείται η εισαγωγή μόνο του ΑΜ για την εκτέλεση της συνάρτησης. Στην συνέχεια, ακολουθούν δυο έλεγχοι. Ο πρώτος βλέπει εάν ο φοιτητής διαθέτει δικαίωμα έκδοσης πτυχίου, ενώ ο δεύτερος ελέγχει εάν ο φοιτητής έχει ήδη εκδώσει το πτυχίο του.

Προχωρώντας παρακάτω, παρατηρούμε ότι χρησιμοποιώντας την λειτουργικότητα της βιβλιοθήκης Counters, περνούμε το τρέχων ID του token που θα δημιουργήσουμε, μετά το αυξάνουμε κατά 1, για να είμαστε έτοιμοι για το επόμενο token. Τέλος, καλούμε την

συνάρτηση *_safeMint()* την οποία μας παρέχει το ERC721.sol και είναι υπεύθυνη για την δημιουργία του NFT. Το NFT αποθηκεύεται στην ιδιωτική χαρτογράφηση του ERC721.sol για να επιπρόσθετη ασφάλεια.

```
// 🔧 This is were the minting occurs... 🔧
function mint(uint _am) public returns (address) {
    require(
        graduableStudents[_am],
        "DegreeManager.sol: You can NOT mint your Degree NFT!"
    );
    require(
        balanceOf(msg.sender) < 1,
        "DegreeManager.sol: Don't be greedy! One Degree is good enough"
    );
    uint256 tokenId = _tokenIdCounter.current();
    _tokenIdCounter.increment();
    _safeMint(msg.sender, tokenId);
    return msg.sender;
}
```

2.3.2 Hardhat

2.3.2.1 Ανάλυση Ρυθμίσεων

Το συγκεκριμένο υποκεφάλαιο αποτελεί το τελευταίο κομμάτι της εργασίας και θα μάθουμε πως να χρησιμοποιήσουμε το Hardhat για να δημιουργήσουμε ένα τοπικό blockchain δίκτυο (το οποίο βασίζεται στην έκδοση *GETH* του Ethereum), να προωθήσουμε τα έξυπνα συμβόλαια μας στο τοπικό δίκτυο και τέλος θα δούμε μερικές τεχνικές που αυτοματοποιούν κάποιες ανιαρές διαδικασίες που θα αυξήσουν σημαντικά την αποδοτικότητα κατά τη φάση εύρεσης / διόρθωσης σφαλμάτων.

Το πρώτο πράγμα με το οποίο πρέπει να ασχοληθούμε είναι το αρχείο *hardhat.config.js*. Αυτό αρχείο περιέχει τις ρυθμίσεις του hardhat για διαφορά σημαντικά θέματα, τα δυο που θα χρησιμοποιήσουμε εμείς είναι: α) την έκδοση της Solidity θα πρέπει να χρησιμοποιηθεί, β) το δίκτυο στο οποίο θα παραταϊστούν τα έξυπνα συμβόλαια.

hardhat.config.js > ...

```
require('@nomiclabs/hardhat-waffle');
require('@nomiclabs/hardhat-etherscan');

// dotenv.config(); // <-- Enables us ot use env files

task('accounts', 'Prints the list of accounts', async (taskArgs, hre) => {
  const accounts = await hre.ethers.getSigners(); // Receives a Array containing the accounts

  for (const account of accounts) {
    console.log(account.address);
  }
});

/**
 * @type import('hardhat/config').HardhatUserConfig
 */
module.exports = {
  solidity: '0.8.10',
  networks: {
    localhost: {
      url: 'http://127.0.0.1:8545',
    },
  },
};

```

Επίσης είναι το σημείο οπού μπορούμε να δημιουργήσουμε τα λεγόμενα “tasks” για να αυτοματοποιήσουμε διαδικασίες, αλλά αυτή η τεχνική είναι αρκετά προχωρημένη και δεν θα ασχοληθούμε μαζί της στη συγκεκριμένη εργασία. Ακολουθεί η εικόνα του αρχείου.

Το επόμενο κομμάτι που θα αναλύσουμε είναι ο κώδικας (γραμμένος σε JavaScript) που θα προωθήσει τα συμβόλαια μας και επίσης θα αυτοματοποίηση της ανιαρές διαδικασίες που αναφέραμε πιο πριν.

2.3.2.2 Κώδικας Προώθησης Έξυπνων Συμβολαίων (Deployment)

Ας ξεκινούμε με τις δηλώσεις των πασουμιών μεταβλητών, παρατηρούμε την χρήση της συνάρτησης *require()*, αλλά η συγκεκριμένη require δεν έχει κάποια σχέση με αυτή στη Solidity. Αυτού του είδους το συντακτικό χρησιμοποιείται για την εισαγωγή κώδικα από εξωτερικά αρχεία στην σύγχρονη JavaScript. Αξίζει να αναφερθεί η περίεργη σύνταξη της 2ης γραμμής κώδικα, αυτή είναι μια καινούργια μέθοδος εν όμματι αποδομισμός (destructuring) (διαθέσιμη από την έκδοση ES6 της JavaScriptκαι μετά) που μας επιτρέπει να αντλήσουμε όσο λειτουργικότητα χρειαζόμαστε από τον κώδικα ενός αλλού αρχείου μειώνοντας το συνολικό μέγεθος του αρχείο μας. Το *hre* (**H**ardhat **R**untime **E**nvironment) είναι ένα αντικείμενο που περιέχει όλες τις λειτουργίες που εκθέτει το Hardhat κατά την εκτέλεση μιας εργασίας (task), δοκιμής ή σεναρίου (script). Στην πραγματικότητα, το Hardhat είναι το HRE. Οπότε στην ουσίαεισάγουμε την λειτουργικότητα του Hardhat στο αρχείο μας. Το *ethers* και το *BigNumber* είναι εσωτερικές βιβλιοθήκες του Hardhat (για την ακρίβεια η κληρονομικότητα πάει κάπως έτσι Hardhat -> ethers.js -> BigNumber), όπως φαίνεται στο VS Code χρησιμοποιούμε τον αποδομισμό για να αντλήσουμε αυτές τις βιβλιοθήκες από το Hardhat αλλά παρατηρούμε ότι απεικονίζονται «ξεθωριασμένες», αυτό συμβαίνει διότι εισάγοντας το hardhat στην πρώτη γραμμή κώδικα, αυτομάτως εμπεριέχονται και αυτές. Θεωρείται καλή πρακτική να τις αναφέρουμε, ακόμα και ας μην είναι απαραίτητο, για την καλύτερη αναγνωσιμότητα του κώδικα μας.

Οι δυο τελευταίες εισαγωγές fs (**F**ile **S**ystem) και fse (**F**ile **S**ystem **E**xtra) θα μας επιτρέπουν να αυτοματοποιήσουμε δυο διαδικασίες για αύξηση της παραγωγικότητας, παρέχοντας μας την δυνατότητα να αντιγράφουμε αρχεία από μια τοποθεσία και να τα επικολλήσουμε σε μια άλλη.

Χρησιμοποιείται αυτή η μέθοδος διότι το framework Create React App (CRA) δεν επιτρέπει στα αρχεία που βρίσκονται στο φάκελο “src” να επικοινωνήσουν με εξωτερικά αρχεία απευθείας.

scripts > deployThemAll_V3.js > ...

```
const hre = require('hardhat');
const { ethers, BigNumber } = require('hardhat');
const fs = require('fs');
const fse = require('fs-extra');

const dsDir =
  'C:/Users/Jimzord/Documents/1_Programming_Stuff/Web3_Apps/degree-maker/degree-maker-app/src/contractAddresses.json';

const dsDir1 =
  'C:/Users/Jimzord/Documents/1_Programming_Stuff/Web3_Apps/degree-maker/degree-maker-app/src/contractABIs';

const srcDir1 =
  'C:/Users/Jimzord/Documents/1_Programming_Stuff/Web3_Apps/degree-maker/degree-maker-app/artifacts/contracts';

const privKey1 =
  '0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80';

// 1 - Grades Manager Contract
const SecreteryDeploy = async function () {
  console.log('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$');
  console.log('$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$');
  console.log('=== #1 - Gets in Secr ===');
  const SecreteryFactory = await hre.ethers.getContractFactory(
    'Secretery',
    privKey1
  );
  console.log('=== #2 (Secr)Factory Created! ===');
  const secreteryContract = await SecreteryFactory.deploy();
  secrAddress = secreteryContract.address;
  console.log('Secr Generated Address: ', secrAddress);
  console.log('=== #3 (Secr) Contract is Deploying... ===');
  await secreteryContract.deployed();
  console.log('=== #4 ===');
  console.log('Secretery deployed to:', secrAddress);
  return secrAddress;
};
```

Συνεχίζοντας αντικρίζουμε τις μεταβλητές *dsDir*, *dsDir1*,*srcDir1* καθώς και τη *privKey1*. Οι πρώτες τρεις περιέχουν τις τοποθεσίες που χρειαζόμαστε για να χρησιμοποιήσουμε τα fs & fse. Η τελευταία περιέχει το ιδιωτικό κλειδί ενός λογισμού που χρησιμοποιείται για δοκιμές, ώστε να γίνει παρουσίαση του τρόπου με τον οποίο μπορούμε να διαλέξουμε ποιος λογαριασμός θα προωθήσει το έξυπνο συμβόλαιο. *Ποτέ και για κανένα λόγω μην επιδείξετέ το δικό σας ιδιωτικό κλειδί, διότι ισοδύναμο με το να δημοσιεύεται κάπου το username και passwords σας.*

Παρακάτω βλέπουμε την συνάρτηση *SecreteryDeploy()* η οποία είναι υπεύθυνη για την παράταση του αντίστοιχου συμβολαίου. Ίσως παρατηρήσατε τη λέξη κλειδί “async” στην δήλωση της συνάρτησης, αυτό υποδηλώνει ότι η συνάρτηση είναι ασύγχρονη, δηλαδή

γνωρίζουμε ότι θα χρειαστεί κάποιο χρονικό διάστημα για να εκτελεστεί πλήρως. Χρησιμοποιείται αυτή η σύνταξη στην JavaScript διότι είναι μια “Single-Threaded” γλώσσα προγραμματισμού με συνέπεια να μην μπορεί να εκτελεί πολλαπλές λειτουργίες ταυτόχρονα (multi-tasking) όπως άλλες γλώσσες (π.χ. Python).

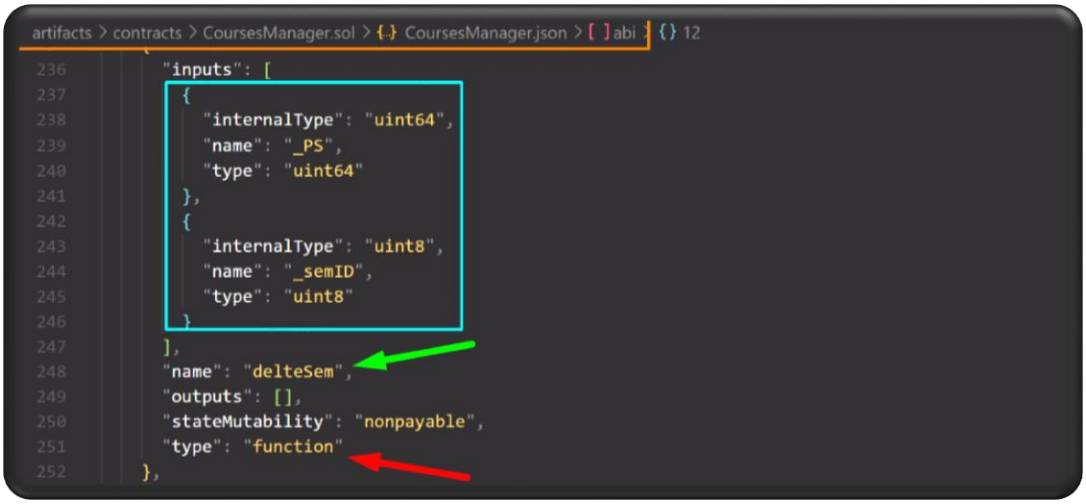
Επιστρέφοντας στο θέμα μας, εάν παραλείψουμε τα console.log()ς παρατήσουμε μια από τις πιο κρίσιμες εντολές που αφορούν την παράταση συμβολαίων, την *hre.ethers.getContractFactory()*. Η λειτουργία αυτής της συνάρτησης είναι η δημιουργία ενός «εργοστασίου» με τις προδιάγραφες του εκάστοτε συμβολαίου. Το εργαστήριο περιέχει διάφορες χρήσιμες μεθόδους που μπορούμε να καλέσουμε αλλά η πιο σημαντική είναι η *deploy()*. Αυτή θα προωθήσει το συμβόλαιο μας και επιστρέφει ένα αντικείμενο που περιέχει την διεύθυνση του νέου συμβολαίου. Τέλος, καλούμε την μέθοδο deployed() του αντικειμένου *secreteryContract* για να μάθουμε ποτέ η παράταση έχει ολοκληρωθεί. Τα υπόλοιπα συμβόλαια ακολουθούν ακριβώς την ιδιά λογική γι’ αυτό δεν θα αναφερθούν.

Υστέρα από τις συναρτήσεις προωθήσεων, ερχόμαστε αντιμέτωποί με ένα προχωρημένο είδος δεδομένου της JavaScript, το οποίο ονομάζεται “Promise”. Εφόσον, ο σκοπός της εργασίας δεν είναι η ανάπτυξη εφαρμογών στην JavaScript δεν θα μελετήσουμε την λειτουργία/χρήση τους, οποίος θέλει να μάθει περισσότερα ας ανατρέξει εδώ:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

2.3.2.3 Αυτόματη Ενημέρωση Δεδομένων

Πριν προχωρήσουμε, πρέπει να κάνουμε μια σύντομη στάση και να μιλήσουμε για το τι είναι τα ABIs. ABI (**A**pplication **B**inary **I**nterface) είναι ο κώδικας που επιτρέπει στο Front-End να επικοινωνεί με τις συναρτήσεις των έξυπνων συμβολαίων. Ένα ABI περιέχει τις απαραίτητες



πληροφορίες των συναρτήσεων (των συμβολαίων), για παράδειγμα, ποιο είναι το όνομα της συνάρτησης, πόσες παραμέτρους διαθέτει, εάν έχει παραμέτρους ποιοι είναι οι τύποι τους, κτλ. Το κομμάτι του ABI που βλέπεται αντιστοιχεί στην συνάρτηση “delteSem()” από το συμβόλαιο CourseManager.sol.

Ας δούμε τώρα, τον τρόπο που θα μας κάνει την ζωή αρκετά ευκολότερη με την διατήρηση του κώδικα μας. Στην μεταβλητή *resultingAddresses* αποθηκεύονται όλες οι διευθύνσεις των συμβολαίων, σε μορφή αντικειμένου, όταν αυτά ολοκληρώσουν τις προωθήσεις τους.

Έπειτα, μετατρέπουμε το περιεχόμενο της μεταβλητής σε μορφή JSON, που είναι πιο διαδεδομένη μορφή αρχείων για αποστολή ή λήψη δεδομένων στο χώρο του διαδικτύου. Χρησιμοποιώντας την λειτουργικότητα της βιβλιοθήκης *fs* γράφουμε ή επικολλούμε το αρχείο JSON στο *φάκελο src*. Επαναλάβουμε την ιδιά διαδικασία και για τα ABIs τα αλλά επειδή αυτήν την φορά θέλουμε να γράψουμε ολόκληρο φάκελο χρησιμοποιούμε την *fse* βιβλιοθήκη.

```
    resultingAddresses = result4;
    console.log('************************************************************');
    console.log(`SecrAddress: ${resultingAddresses.SecreteryAddr}`);
    console.log(`GMAddress:   ${resultingAddresses.GMAddress}`);
    console.log(`CMAddress:   ${resultingAddresses.CMAddress}`);
    console.log(`DMAddress:   ${resultingAddresses.DMAddress}`);
    console.log('************************************************************');
    const jsonContent = JSON.stringify(resultingAddresses);

    fs.writeFile(dsDir, jsonContent, 'utf8', function (err) {
      if (err) {
        console.log('An error occured while writing JSON Object to File.');
        return console.log(err);
      }

      console.log(
        'A JSON file containing the Addresses has been saved to /src dir.'
      );
    });

    fse.copySync(
      srcDir1,
      dsDir1,
      {
        overwrite: true,
      },
      (err) => {
        if (err) {
          console.error(err);
        }
      }
    );
    console.log('A folder containing the ABIs has been saved to /src dir.');
    console.log('************************************************************');
  });
```

Έτσι, κάθε φορά που κάνετε την παραμικρή αλλαγή στον κώδικα των έξυπνων συμβολαίων σας δεν θα χρειάζεται χειροκίνητα να αντιγράφεται και να επικολλάται τις διευθύνσεις και τα ABIs που θα δημιουργούνται.

2.3.2.4 Τοπικό Δίκτυο & Εκτέλεση Σεναρίων

Φτάνοντας στο τέλος της εργασίας δεν μένει πάρα να δημιουργήσουμε το τοπικό δίκτυο και να προωθήσουμε σε αυτό τα συμβόλαια μας εκτελώντας το σενάριο (script) που είδαμε στο 2.3.2.2.

Χρησιμοποιώντας το Hardhat, η έναρξη ενός τοπικό blockchain απαιτεί τη εισαγωγή μονάχα μιας εντολής στο τερματικό (terminal), την ακόλουθη:

npx hardhat node

```
PS C:\Users\Jimzord\Documents\1_Programming_Stuff\Web3_Apps\degree-maker\degree-maker-app> npx hardhat node
The Hardhat Network tracing engine could not be initialized. Run Hardhat with --verbose to learn more.
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

Accounts
========

WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.

Account #0: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266 (10000 ETH)
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80

Account #1: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8 (10000 ETH)
Private Key: 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d

Account #2: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC (10000 ETH)
Private Key: 0x5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a

Account #3: 0x90F79bf6EB2c4f870365E785982E1f101E93b906 (10000 ETH)
Private Key: 0x7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6

Account #4: 0x15d34AAf54267DB7D7c367839AAf71A00a2C6A65 (10000 ETH)
Private Key: 0x47e179ec197488593b187f80a00eb0da91f1b9d0b13f8733639f19c30a34926a
```

Για την προώθηση των έξυπνων συμβολαίων, θα χρειαστεί να ανοίξουμε ένα καινούργιο τερματικό και να πληκτρολογήσουμε την ακόλουθη εντολή:

npx hardhat run –network localhost scripts/<το όνομα του αρχείου σας>.js

```
PS C:\Users\Jimzord\Documents\1_Programming_Stuff\Web3_Apps\degree-maker\degree-maker-app> npx hardhat run --network localhost scripts/deployThemAll_V3.js
Compiled 6 Solidity files successfully
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
=== #1 - Gets in Secr ===
GM did NOT deploy
CM did NOT deploy
=== #2 (Secr)Factory Created! ===
Secr Generated Address:  0x5FbDB2315678afecb367f032d93F642f64180aa3
=== #3 (Secr) Contract is Deploying... ===
=== #4 ===
Secretery deployed to: 0x5FbDB2315678afecb367f032d93F642f64180aa3
=== #5 - Gets in GM  ===
=== #6 - (GM)Factory Created! ===
[GM] Secr Addr:  0x5FbDB2315678afecb367f032d93F642f64180aa3
GM Generated Address:  0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
=== #7 - (GM) Contract is Deploying... ===
GradesManager deployed to: 0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
2nd Then() {
  _secrAddress: '0x5FbDB2315678afecb367f032d93F642f64180aa3',
  GMAddress: '0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512'
}
=== #8 - Gets in CM ===
=== #9 - (CM)Factory Created! ===
[CM] Secr Addr:  0x5FbDB2315678afecb367f032d93F642f64180aa3
[CM] GM Addr:  0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
CM Generated Address:  0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0
```

```
CM Generated Address:  0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0
=== #10 - (CM) Contract is Deploying... ===
CoursesManager deployed to: 0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0
DM did NOT deploy
=== #11 - Gets in DM  ===
=== #12 - (DM)Factory Created! ===
[DM] CM Addr:  0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0
[DM] GM Addr:  0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
DM Generated Address:  0xCf7Ed3AccA5a467e9e704C703E8D87F634fB0Fc9
=== #13 - (DM) Contract is Deploying... ===
DegreeManager deployed to: 0xCf7Ed3AccA5a467e9e704C703E8D87F634fB0Fc9
[DM] Ak-47 ---
{
  SecreteryAddr: '0x5FbDB2315678afecb367f032d93F642f64180aa3',
  GMAddress: '0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512',
  DMAddress: '0xCf7Ed3AccA5a467e9e704C703E8D87F634fB0Fc9',
  CMAddress: '0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0'
}
**********************************************************
SecrAddress: 0x5FbDB2315678afecb367f032d93F642f64180aa3
GMAddress:   0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
CMAddress:   0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0
DMAddress:   0xCf7Ed3AccA5a467e9e704C703E8D87F634fB0Fc9
**********************************************************
A folder containing the ABIs has been saved to /src dir.
**********************************************************
A JSON file containing the Addresses has been saved to /src dir.
```

Γ. Συμπεράσματα

Το blockchain όπως είδαμε, ξεκίνησε ως μια τεχνολογία με βασικό σκοπό να παρέχει μια νέα πιο αποκεντρωμένη προσέγγιση στο τομέα των οικονομικών συναλλαγών (Bitcoin).

Στην συνέχεια, με τον ερχομό του Ethereum, ο κόσμος του προγραμματισμού άρχισε να οραματίζεται ένα νέο διαδίκτυο το οποίο θα είναι τελείως αποκεντρωμένο και ο κάθε χρήστης θα έχει τον πλήρη έλεγχο των προσωπικών του δεδομένων.

Αυτό το όραμά έδωσε κίνητρο σε πολλούς οργανισμούς να χρησιμοποιήσουν την τεχνολογία blockchain για να δημιουργήσουν αποκεντρωμένες εφαρμογές και πλατφόρμες. Ξεχωρίζουν ιδιαίτερα οι εφαρμογές που ανήκουν στους τομείς:

- DeFi (Decentralized Finance)
- DAO (Decentralized Autonomous Organization)
- P2E (Play-to-Earn)

Η παρούσα διπλωματική εργασία έχει ως σκοπό την παροχή των βασικών γνώσεων για την κατανόηση του τρόπου λειτουργίας ενός δικτύου blockchain καθώς και την επίδειξη των δομικών στοιχείων, έξυπνα συμβόλαια, που είναι υπεύθυνα για την λειτουργικότητα που υφίσταται σε ένα τέτοιο δίκτυο.

Όπως είδαμε με την χρήση των έξυπνων συμβολαίων μπορούμε να εφαρμόσουμε σχεδόν οποιουδήποτε είδους λογική με αποτέλεσμα να μπορούμε να κατασκευάσουμε ισχυρά συστήματα που θα επιτρέψουν σε τομείς όπως τη δικαιοσύνη, τη βιομηχανία, το εμπόριο , κ.π.α. να αυτοματοποιηθούν. Μειώνοντας περιττά έξοδα και αυξάνοντας την ποιότητα.

Ένα από τα πιο σημαντικά πλεονεκτήματα της χρήσης των τεχνολογιών blockchain στην σύγχρονη εποχή είναι η «ιδιότητα της ιδιοκτησίας». Με την χρήση των διακριτών προτύπων (Token Standards) και συγκεκριμένα το πρότυπο ERC-721 μπορούμε να δημιουργήσουμε πλατφόρμες που θα αποθηκεύουν στο blockchain τεκμήρια τα οποία θα αντιστοιχούν σε πραγματικά αγαθά χωρίς την ανάγκη για παραχώρηση προσωπικών δεδομένων σε τρίτα μέλη ή την συμπλήρωση πολλαπλών εγγράφων.

Δ. Βιβλιογραφικές Αναφορές

[1] Takashima, “The Ultimate Guide To The World Of Blockchain Technology, Bitcoin, Ethereum”, 2017.

[2] Manav Gupta, “Blockchain For Dummies (IBM)”, John Wiley & Sons Inc., United States of America, 2017.

[3] M. Glaser, ‘‘Pervasive decentralization of digital infrastructures: A framework for blockchain-enabled system and use case analysis,’’ presented at the 50th Hawaii Int. Conf. Syst. Sci., Waikoloa, HI, USA, 2017.

[4] R. Beck, C. Müller-Bloch, and J. L. King, ‘‘Governance in the blockchain economy: A framework and research agenda,’’ J. Assoc. Inf. Syst., vol. 19, no. 10, pp. 1020–1034, 2018, DOI: 10.17705/1jais.00518.

[5] K. Nærland, C. Müller-Bloch, R. Beck, and S. Palmund, ‘‘Blockchain to rule the waves: Nascent design principles for reducing risk and uncertainty in decentralized environments,’’ in Proc. 38th Int. Conf. Inf. Syst., Seoul, South Korea, 2017, pp. 1–16.

[6] E. K. Clemons, R. M. Dewan, R. J. Kauffman, and T. A. Weber, ‘‘Understanding the information-based transformation of strategy and society,’’ J. Manage. Inf. Syst., vol. 34, no. 2, pp. 425–456, Apr. 2017.

[7] M. Iansiti and K. R. Lakhani, ‘‘The truth about blockchain,’’ Harvard Bus. Rev., vol. 95, no. 1, pp. 118–127, 2017.

[8] I. Milic. (2019). Blockchain Statistics and Facts That Will Make You Think: The Dawn of Hyper Capitalism. [Online]. Διαθέσιμο: https://fortunly.com/statistics/blockchain-statistics/#gref

[9] 6 Major Features Of Blockchain | Why Blockchain is Popular?. [Online]. Διαθέσιμο: https://data-flair.training/blogs/features-of-blockchain/

[10] Χριστίδης Ιωάννης, “Τεχνολογία Blockchain και ανάπτυξη έξυπνων συμβολαίων για την αγοραπωλησία μοναδικών τεκμηρίων”, Διπλωματική Εργασία, Πανεπιστήμιο Δυτικής Αττικής, 2020.

[11] L. Lamport, ‘‘Paxos made simple,’’ ACM SIGACT News, vol. 32, no. 4,

pp. 18–25, 2001

[12] S. Nakamoto. (2017). Bitcoin: A Peer-to-Peer Electronic Cash System, Oct. 2008. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[13] C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz, ‘‘Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications, and opportunities,’’ IEEE Access, vol. 7, pp. 85727–85745, 2019.IEEE Access, vol. 7, pp. 85727–85745, 2019

[14] L. M. Bach, B. Mihaljevic, and M. Zagar, ‘‘Comparative analysis of blockchain consensus algorithms,’’ in Proc. 41st Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO), May 2018, pp. 1545–1550

[15] (2020). VeriCoin. [Online]. Available: https://vericonomy.ams3.cdn.digitaloceanspaces.com/documents/VeriCoin-Proof-of-Stake-TimeWhitepaper.pdf

[16] (2020). VeroCoin. [Online]. Available: https://vericoin.info/vericoindigital-currency/

[17] F. Tschorsch and B. Scheuermann, ‘‘Bitcoin and beyond: A technical survey on decentralized digital currencies,’’ IEEE Commun. Surveys Tuts., vol. 18, no. 3, pp. 2084–2123, 3rd Quart., 2016.

[18] K. Karantias, A. Kiayias, and D. Zindros, ‘‘Proof-of-burn,’’ in Proc. Int. Conf. Financial Cryptogr. Data Security., 2019, pp. 523–540.

[19] L. Ismail and H. Materwala, ‘‘A review of blockchain architecture and consensus protocols: Use cases, challenges, and solutions,’’ Symmetry, vol. 11, no. 10, p. 1198, Sep. 2019.

[20] A. Yakovenko, ‘‘Solana: A new architecture for a high-performance blockchain,’’ Solana Labs, Tech. Rep., 2018.

[21] (2020). Solana. [Online]. Available: https://solana.com/

[22] A. Shahaab, B. Lidgey, C. Hewage, and I. Khan, ‘‘Applicability and appropriateness of distributed ledgers consensus protocols in public and private sectors: A systematic review,’’ IEEE Access, vol. 7, pp. 43622–43636, 2019

[23] (2020). VeriBlock. [Online]. Available: https://mirror1.veriblock.org/ Proof-of-Proof_and_VeriBlock_Blockchain_Protocol_Consensus_ Algorithm_and_Economic_Incentivization_v1.0.pdf

[24] (2020). Veriblock. [Online]. Available: https://www.veriblock.com/

[25] (2018). Dylan Yaga Peter Mell, Nik Roby, Karen Scarfone, ‘‘NISTIR 8202 Blockchain Technology Overview’’, U.S. Department of Commerce

[25] Andreas M. Antonopoulos and Dr. Gavin Wood, “Mastering Ethereum”, O’Reilly Media, Inc, United States of America, 2019.

[26] (2019) Leonardo Maria, Gianluca Salviotti, Nico Abbatemarco, “Towards a Comprehensive Blockchain Architecture Continuum”, Proceedings of the 52nd Hawaii International Conference on System Sciences

[27] T. Neudecker and H. Hartenstein, “Short paper: An empirical analysis of blockchain forks in bitcoin,” in International Conference on Financial Cryptography and Data Security, pp. 84–92, Springer, 2019.

[28] Neo C.K. Yiu, Member, IEEE Department of Computer Science, University of Oxford, “An Overview of Forks and Coordination in Blockchain Development”

[29] EC-Council, “The need for forking in blockchain,” Available at: https://blog.eccouncil.org/ the-need-for-forking-in-blockchain, 2020

[30] Sumedha Bose, “Major Differences Between Full Node & Lightweight Node”, 2018, Διαθέσιμο [Online]: https://www.btcwires.com/block-o-pedia/major-differences-between-full-node-lightweight-node/

[31] Elad Elrom, “The Blockchain Developer: A Practical Guide for Designing, Implementing, Publishing, Testing, and Securing Distributed Blockchain-based Projects”, New York, NY, USA, 2019, Ch. 2 p. 31-32.

[32] Fairfield, J.: Tokenized: The law of non-fungible tokens and unique digital property. Indiana Law Journal, Forthcoming (2021)

[33] Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper 151(2014), 1–32 (2014)

[34] William, E., Dieter, S., Jacob, E., Nastassia, S.: Erc-721 non-fungible token standard. Ethereum Improvement Protocol, EIP-721, Διαθέσιμο: https://eips.eth ereum.org/EIPS/eip-721. (2018)

[35] Witek, R., Andrew, C., Philippe, C., James, T., Eric, B., Ronan, S.: Eip-1155: Erc-1155 multi token standard. Ethereum Improvement Protocol, EIP-1155, Διαθέσιμο: https://eips.ethereum.org/EIPS/eip-1155. (2018)

[36] Shirole, M., Darisi, M., Bhirud, S.: Cryptocurrency token: An overview. IC-BCT

2019 pp. 133–140 (2020)

[37] Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Tech. rep., Manubot

(2019)

[38] Qin Wang, Rujia Li, Qi Wang, Shiping Chen, "Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges (Tech Report^{V2})" (May 2021)

[39] Franceschet, M., Colavizza, G., Smith, T., et al.: Crypto art: A decentralized

view. Leonardo pp. 1–8 (2020)

[40] Regner, F., Urbach, N., Schweizer, A.: Nfts in practice–non-fungible tokens as core component of a blockchain-based event ticketing application (2019)

[41] Omar, A.S., Basir, O.: Capability-based non-fungible tokens approach for a decentralized aaa framework in iot. In: Blockchain Cybersecurity, Trust and Privacy, pp. 7–31. Springer (2020)

[42] Nonfungible website. Διαθέσιμο: https://NonFungible.com (2021)

[43] Dappradar website. Διαθέσιμο: https://dappradar.com/ (2021)

[44] Nft bank. Project Διαθέσιμο: https://nftbank.ai/ (2021)

[45] Deipulse website. Διαθέσιμο: https://defipulse.com/ (2021)

[46] Coingecko website. Διαθέσιμο: https://coingecko.com/en (2021)

[47] Cryptoslam. Project Διαθέσιμο: https://cryptoslam.io/ (2021)

[48] Opensea platform. Διαθέσιμο: https://opensea.io/ (2021)

[49] Superrare. Project Διαθέσιμο: https://superrare.co/ (2021)

[50] Nifty gateway. Project Διαθέσιμο: https://niftygateway.com/ (2021)

[51] Rarible. Project Διαθέσιμο: https://rarible.com/ (2021)

[52] Zora. Project Διαθέσιμο: https://zora.co/ (2021)

[53] Zcash Project. Διαθέσιμο: https://z.cash/ (2022)

[54] Monero Project. Διαθέσιμο: https://www.getmonero.org/ (2022)

[55] Massimo Bertaccini, “Cryptography Algorithms”, Packt> BIRMINGHAM—MUMBAI (2022)

[56] Ali Maetouq, Salwani Mohd Daud, Noor Azurati Ahmad, Nurazean Maarop, Nilam Nur Amir Sjarif, Hafiza Abas, “Comparison of Hash Function Algorithms Against Attacks: A Review”, Advanced Informatics Department Razak Faculty of Technology and Informatics Universiti Teknologi Malaysia Kuala Lumpur, Malaysia (2018)

[57] X. Wang and H. Yu, —How to Break MD5 and Other Hash Functions, EUROCRYPT 2005, LNCS 3494, pp. 19–35, 2005.

[58] T. Xie, F. Liu, and D. Feng, —Fast Collision Attack on MD5, pp. 1–12, 2013.

[59] V. Chiriaco, A. Franzen, and R. Thayil, —Finding Partial Hash Collisions by Brute Force Parallel Programming, in 37th IEEE Sarnoff Symposium 2016, Newark, NJ, September 19-21, 2016, vol. 5, pp. 1–6.

[60] F. Mendel, T. Peyrin, and M. Schl, —Improved Crypanalysis of Reduced RIPEMD-160, Int. Conf. Theory Appl. Cryptol. Inf. Secuerty., pp. 484–503, 2013.

[61] X. Wang, Y. L. Yin, and H. Yu, —Finding Collisions in the Full SHA-1, Int. Assoc. Cryptologic Res. 2005, no. 90304009, pp. 17–36, 2005.

[62] M. Stevens, P. Karpman, and T. Peyrin, —Freestart collision for full SHA-1, EUROCRYPT 2016., vol. 2012, pp. 1–21, 2016.

[63] K. Wu, Y. Li, L. Chen, and Z. Wang, —Research of Integrity and Authentication in OPC UA Communication Using Whirlpool Hash Function, Appl. Sci. ISSN2076-3417, pp. 446–458, 2015.

[64] S. Verma and G. Prajapati, —Robustness and Security Enhancement of SHA with Modified Message Digest and Larger Bit Difference, in 2016 Symposium on Colossal Data Analysis and Networking (CDAN), 2016, pp. 0–4.

[65] G. Meliolla, K. A. Nugroho, and F. I. Hariadi, —Implementation of Hash Function on Embedded- System Platform using Chaotic Tent Map Algorithm, in Electronics and Smart Devices (ISESD), International Symposium on, 2016, pp. 179–183.

[66] N. Kishore and B. Kappor, —Attacks on and Advances in Secure Hash Attacks on and Advances in Secure Hash Algorithms, IAENG Int. J. Comput. Sci., no. September, 2016.

[67] I. Dinur, O. Dunkelman, and A. Shamir, —New attacks on Keccak-224 and Keccak-256, pp. 1–27, 2011.

[68] I. Dinur, O. Dunkelman, and A. Shamir, —Collision Attacks on Up to 5 Rounds of SHA-3 Using Generalized Internal Differentials, no. 827, pp. 1–30, 2013.

[69] D. Kim, D. H. B, J. Lee, and W. Kim, —LSH : A New Fast Secure Hash Function Family, Springer Int. Publ. Switz. 2015, pp. 286–313, 2015.

[70] Dimitri Nitchoun, “Easily understand the difference between Private Blockchain and Public Blockchain!”, Αυγ 2019. Διαθέσιμο [Online]: https://medium.com/@Equisafe/easily-understand-the-difference-between-private-blockchain-and-public-blockchain-2c4f9b2111b

[71] (2020). PoA. [Online]. Available: https://www.poa.network/for-users/ whitepaper/poadao-v1/proof-of-authority

[72] O. Samuel, N. Javaid, M. Awais, Z. Ahmed, M. Imran, and M. Guizani, ‘‘A blockchain model for fair data sharing in deregulated smart grids,’’ in Proc. IEEE Global Commun. Conf. (GLOBECOM), Dec. 2019, pp. 1–7.

[73] S. D. Angelis, L. Aniello, R. Baldoni, F. Lombardi, A. Margheri, and V. Sassone, ‘‘PBFT vs proof-of-authority: Applying the CAP theorem to permissioned blockchain,’’ in Proc. Italian Conf. Cyber Secur., 2018, pp. 1–11.

[74] Andreas M. Antonopoulos, “Mastering Bitcoin”, O’Reilly (2015)

[75] Zelcore, “Solana: An Innovative Crypto That’s Taking Over the Market”, Feb 2022. Διαθέσιμο [Online]: https://medium.zelcore.io/solana-an-innovative-crypto-thats-taking-over-the-market-c745008a11f

[76] Guang Chen, Bing Xu, Manli Lu, Nian-Shing Chen, “Exploring blockchain technology and its potential applications for education”, Smart Learning Environments, 2018

[77] MDN Web Docs, “What is JavaScript?”, 2022. Διαθεσιμο [Online]: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

[78] Nomic Foundation, “Hardhat Overview”. Διαθεσιμο [Online]: https://hardhat.org/getting-started