Πανεπιστήμιο Αιγαίου
Τμήμα Μαθηματικών

Μεταπτυχιακή Εργασία

Σπουδές Στα Μαθηματικά

Ανάλυση Ανωνυμίας Κρυπτονομισμάτων και Μέθοδοι Άρσης της

Συγγραφέας:
Γεώργιος Πετρούδης

Επιβλέπων:
Παναγιώτης Νάστου

11 Σεπτεβρίου 2020

Τριμελής Επιτροπή
Βασίλειος Μεταφτσής, Καθηγητής
Ανδρέας Παπασαλούρος, Επίκουρος Καθηγητής
Παναγιώτης Νάστου, Επίκουρος Καθηγητής

Αφιερωμένη, στην οικογένειά μου και τους φίλους μου.

Γεώργιος Πετρούδης,
Σάμος 2020.

Ευχαριστώ τον επιβλέποντα καθηγητή μου, κύριο Νάστου, για το ενδιαφέρον και την εμπιστοσύνη που μου έδειξε ώστε να φέρω εις πέρας τη παρούσα μεταπτυχιακή εργασία.

Ένα μεγάλο ευχαριστώ στους γονείς μου για την ανιδιοτελή αγάπη και υποστήριξη που μου πρόσφεραν και συνεχίζουν να μου προσφέρουν όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Σχημάτων

Κεφάλαιο 1

Κρυπτονομίσματα

Με την ανάπτυξη της τεχνολογίας, έχουν γίνει πραγματικότητα πράγματα τα οποία στο παρελθόν φάνταζαν αδύνατα να συμβούν. Μέρος αυτής της νέας πραγματικότητας είναι ο χώρος της οικονομίας και των αγορών. Με το πάτημα ενός μόνο κουμπιού, ο καθένας μας πλέον μπορεί να αγοράσει ένα αγαθό που βρίσκεται στην άλλη άκρη της γης. Οι συναλλαγές μέσω του διαδικτύου έχουν γίνει καθημερινότητα. Πώς όμως λειτουργεί μια τέτοια συναλλαγή; Είναι απόλυτα ασφαλές ή υπάρχει κενό ασφαλείας; Στο κεφάλαιο αυτό θα δούμε ποιο είναι το πρόβλημα σε αυτές τις συναλλαγές. Πως αυτό οδήγησε στη δημιουργία κρυπτονομισμάτων και τι είναι τα κρυπτονομίσματα. Πως η τεχνολογία έφερε τη ραγδαία εξέλιξη τους και για ποιο λόγο ένα κρυπτονόμισμα, το bitcoin έφτασε να έχει τόσο μεγάλη αξία κυριαρχώντας στον κόσμο των κρυπτονομισμάτων.

1.1 Εισαγωγή

Οι ηλεκτρονικές αγορές, όπως είπαμε, έχουν εισβάλει στην καθημερινότητά μας. Πως λειτουργεί όμως μια τέτοια αγορά; Αν μπορούσαμε να απαντήσουμε σε αυτό το ερώτημα με μια λέξη τότε αυτή θα ήταν "τράπεζες". Όλο το οικονομικό οικοδόμημα βασίζεται πάνω στο τραπεζικό σύστημα. Ένα μικρό, απλό παράδειγμα για την κατανόηση αυτού του συστήματος είναι το εξής, ας υποθέσουμε ότι δύο άνθρωποι από διαφορετικές περιοχές θέλουν να κάνουν μια συναλλαγή μεταξύ τους. Ο Α θέλει να αγοράσει κάποιο αγαθό από τον Β μέσω του διαδικτύου. Για να γίνει δυνατή μια τέτοια αγορά, ο Α πρέπει να πληροί μια βασική προϋπόθεση, να έχει χρήματα σε κάποιο τραπεζικό λογαριασμό. Αν όντως έχει τότε η διαδικασία είναι απλή. Αυτό που έχει να κάνει είναι να συμπληρώσει μια ηλεκτρονική φόρμα με τα στοιχεία του για να ολοκληρώσει την παραγγελία. Από εκεί και πέρα η τράπεζα είναι αυτή που κάνει τις επόμενες ενέργειες. Βλέπει τη συναλλαγή, την επιτρέπει και μεταφέρει το

χρηματικό ποσό που αναφέρεται στη παραγγελία στον τραπεζικό λογαριασμό του Β.

Στο παραπάνω παράδειγμα, με λίγα λόγια, η τράπεζα έχει τον ρόλο ενός τρίτου εταίρου που εμπλέκεται στη συναλλαγή. Ένας τρίτος εταίρος τον οποίο ο Α και ο Β εμπιστεύονται ώστε να ολοκληρώσει αυτή τη συναλλαγή καθώς τα χρήματα βρίσκονται στα δικά του ″χέρια″.

Σε μια τέτοια διαδικτυακή συναλλαγή, που προϋποθέτει την εμπιστοσύνη σε μια τρίτη οντότητα, υπάρχουν προβλήματα. Για παράδειγμα, αν ο Α ήθελε να μετακινήσει ένα μεγάλο ποσό, δε θα μπορούσε να το κάνει ή ακόμα και στην καλύτερη περίπτωση θα έπρεπε να περιμένει αρκετό καιρό μέχρι να γίνει κάτι τέτοιο εφικτό. Επίσης, για να κάνει κάποιος μια συναλλαγή, όπως είδαμε, χρησιμοποιεί πολλά προσωπικά του στοιχεία τα οποία η τράπεζα κρατάει κρυφά. Σε περίπτωση όμως που το σύστημα ασφαλείας της τράπεζας διαπεραστεί το αποτέλεσμα είναι να είναι εκτεθειμένος σε διαδικτυακές απάτες. Εκτός αυτού, και ο κάθε πωλητής θα πρέπει να έχει μια σχέση εμπιστοσύνης με τον αγοραστή, που θα του εξασφαλίζει ότι θα πάρει τα χρήματα.

Από τα παραπάνω προκύπτει ότι καθώς σε ένα μεγάλο κομμάτι τους οι συναλλαγές φαίνεται να λειτουργούν σωστά, υπάρχει ένα κενό ασφαλείας το οποίο και θεωρείται αποδεκτό. Υπάρχει μεγάλη διαρροή προσωπικών δεδομένων και ανά πάσα στιγμή μπορεί να χαλάσει μια συναλλαγή ή ακόμα και κάποιος να μην έχει πρόσβαση στα χρήματά του καθώς τα έχει εμπιστευθεί σε κάποια τράπεζα.

Αυτή η σχέση αμφιλεγόμενης εμπιστοσύνης σε ένα τρίτο πρόσωπο είναι και ο λόγος ύπαρξης των περισσότερων κρυπτονομισμάτων. Όμως τι είναι ένα κρυπτονόμισμα;

Ορισμός 1.1.1 *Ως κρυπτονόμισμα ορίζουμε κάποιο συνάλλαγμα το οποίο χρησιμοποιεί την κρυπτογραφία ως εργαλείο για την κατωχύρωση ιδιοκτησίας του κρυπτονομίσματος, για την προστασία κατά της απάτης διπλής χρήσης (double-spendng), για την εγγύηση της ανωνυμίας και ιδιωτικότητάς του καθώς και για την δημιουργία νέων κρυπτονομισμάτων [1].*

Ανά καιρούς είχαν προταθεί πολλά κρυπτονομίσματα που θα έλυναν τα προβλήματα εμπιστοσύνης σε τρίτους. Ένα από τα πιο σημαντικά ήταν το b-money. Πάνω σε αυτά τα προβλήματα και στην αρχή του b-money στάθηκε ο Satoshi Nakamoto (ψευδώνυμο) όταν το 2008 με μια δημοσίευση 9 σελίδων με τίτλο *Bitcoin: A Peer-to-Peer Electronic Cash System* [2] μας σύστησε στο bitcoin, το κρυπτονόμισμα με τη μεγαλύτερη αξία αυτή τη στιγμή και με το οποίο και θα ασχοληθούμε. Το bitcoin είναι γνωστό κυρίως για δύο λόγους. Για τον τρόπο με τον οποίο εγγυάται την ανωνυμία του χρήστη καθώς και για τον τρόπο με τον οποίο παράγεται.

Το bitcoin χρησιμοποιεί ασύμμετρο σύστημα κρυπτογράφησης, μέσω του οποίου γίνεται η ψηφιακή υπογραφή με την οποία κατοχυρώνεται η ιδιοκτησία των παραγόμενων κρυπτονομισμάτων. Επίσης, εκμεταλλεύεται την τεχνολογία blockchain ως

μέσω διατήρησης ιστορικού συναλλαγών, αλλά και για την αποφυγή της απάτης της διπλής χρήσης (double-spending). Επιπροσθέτως, το bitcoin προσφέρει στους χρήστες του την ανωνυμία.

Για να καταλάβει κανείς τη λειτουργία του, θα πρέπει να καταλάβει τα εργαλεία που το συμμετέχουν στην παραγωγή και χρήση του. Παρακάτω θα εξετάσουμε αναλυτικά τα εργαλεία που χρησιμοποιεί ένας χρήστης, το δίκτυο του bitcoin, για να σκεπάστει ο "κόσμος" του bitcoin στην ανωνυμία. Τέλος, θα δούμε πως όλα αυτά θα συνδυαστούν για να διαμορφώσουν τα δίκτυα συναλλαγών όπως και τα δίκτυα των χρηστών μέσα από το συνολικό δίκτυο.

1.2 Εργαλεία Κρυπτογράφησης

Σε αυτήν την ενότητα θα μελετήσουμε τα κρυπτογραφικά εργαλεία που χρησιμοποιεί το bitcoin καθώς και τον λόγο που τα χρησιμοποιεί. Θα δούμε τι είναι οι συναρτήσεις κατακερματισμού hash functions, ποιες από αυτές θα χρησιμοποιήσουμε, που μας χρησιμεύουν τα Merkle Trees καθώς και ποια κρυπτοσυστήματα χρησιμοποιούνται από το δίκτυο του bitcoin.

1.2.1 Ασύμμετρα Κρυπτοσυστήματα

Στην κρυπτογραφία, τα κρυπτοσυστήματα χωρίζονται σε δύο μεγάλες κατηγορίες τα συμμετρικά και τα ασύμμετρα συστήματα. Αυτός ο διαχωρισμός έχει να κάνει με το κλειδί της κρυπτογράφησης. Στη περίπτωση των συμμετρικών κρυπτοσυστημάτων, χρησιμοποιείται το ίδιο κλειδί για την κρυπτογράφηση και την αποκρυπτογράφηση ενός κειμένου. Στη περίπτωση του ασύμμετρου κρυπτοσυστήματος, χρησιμοποιείται ένα κλειδί, το οποίο ονομάζεται δημόσιο, για την κρυπτογράφηση και ένα άλλο κλειδί, το ιδιωτικό, για την αποκρυπτογράφηση ενός κειμένου. Για να καταλάβουμε καλύτερα τα κρυπτοσυστήματα όμως πρέπει να δώσουμε τον ορισμό τους.

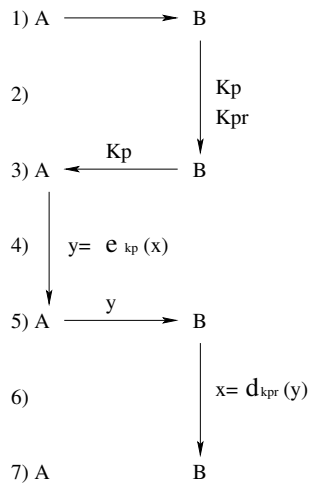
Ορισμός 1.2.1 *Ένα κρυπτοσύστημα ορίζεται ως η διατεταγμένη πεντάδα*

$$(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$$

όπου:

1. *Το $\mathcal{P}$ συμβολίζει το σύνολο όλων των δυνατών απλών κειμένων, δηλαδή το κείμενο που κάποιος θέλει να κρυπτογραφήσει.*

2. *Το $\mathcal{C}$ συμβολίζει το σύνολο όλων των δυνατών κρυπτοκειμένων, δηλαδή το αποτέλεσμα του κειμένου που έχει κρυπτογραφηθεί.*



Σχήμα 1.1: Ασύμμετρη Επικοινωνία

3. *Το $\mathcal{K}$ συμβολίζει το σύνολο όλων των δυνατών κλειδιών που θα χρησιμοποιηθούν για την κρυπτογράφηση.*

4. *Το $\mathcal{E}$ συμβολίζει το σύνολο όλων των δυνατών συναρτήσεων κρυπτογράφησης.*

5. *Το $\mathcal{D}$ συμβολίζει το σύνολο όλων των δυνατών συναρτήσεων αποκρυπτογραφήσεων.*

Για το συμμετρικό έχουμε ένα κλειδί $K \in \mathcal{K}$, ενώ για το ασύμμετρο έχουμε $K = (K_p, K_{pr})$ όπου K_p είναι το δημόσιο κλειδί και K_{pr} το ιδιωτικό.

Τα βήματα του πρωτόκολλου επικοινωνίας μέσω ασύμμετρου κρυπτοσυστήματος δύο οντοτήτων A και B δίνονται στο Σχήμα 1.1. Συγκεκριμένα τα βήματα έχουν ως εξής,

1. Ο Α ενημερώνει τον Β ότι θέλει να επικοινωνήσουν.

2. Ο Β δημιουργεί δύο κλειδιά, το K_p δημόσιο κλειδί και το K_{pr} ιδιωτικό κλειδί, με $K = (K_p, K_{pr}) \in \mathcal{K}$.

3. Ο Β στέλνει το δημόσιο κλειδί K_p στον Α.

4. Ο Α παίρνει το δημόσιο κλειδί και το χρησιμοποιεί για να κρυτογραφήσει ένα απλό κείμενο $x \in \mathcal{P}$ μέσω μιας συνάρτησης $e_{K_p} \in \mathcal{E}$ που θα έχει ως αποτέλεσμα το κρυπτοκείμενο $y \in \mathcal{C}$.

5. Ο Α θα στείλει το κρυπτοκείμενο y στον Β.

6. Ο Β παίρνει το κρυπτοκείμενο και χρησιμοποιεί το ιδιωτικό του κλειδί για να το αποκρυπτογραφήσει μέσω μιας συνάρτησης αποκρυπτογράφησης $d_{K_{pr}} \in \mathcal{D}$ που θα έχει ως αποτέλεσμα το απλό κείμενο x.

7. Ο Α πέρασε με ασφάλεια το μήνυμα του στον Β και τώρα αν ο Β θελήσει να απαντήσει στον Α, η διαδικασία θα πρέπει να γίνει αντιστρόφως, δηλαδή στη θέση του Α θα πρέπει να μπει ο Β και στη θέση του Β ο Α.

Στο επόμενο εδάφιο θα εξετάσουμε τα πιο σημαντικά ασύμμετρα κρυπτοσυστήματα.

Το Κρυπτοσύστημα RSA

Η ασφάλεια του RSA (Rivest,Shamir,Adleman), βασίζεται στη δυσκολία επίλυσης του μαθηματικού προβλήματος της παραγοντοποίησης ενός μεγάλου θετικού ακεραίου ως γινόμενο δύο πρώτων αριθμών ίσου μήκους.

Ορίζεται ως η διατεταγμένη πεντάδα συνόλων

$$(\mathcal{P} = \mathbb{Z}_n, \mathcal{C} = \mathbb{Z}_n, \mathcal{K} = \{(n, p, q, a, b)\}, \mathcal{E}, \mathcal{D})$$

όπου:

- $n = p \cdot q$, με p και q να είναι πρώτοι αριθμοί.
- $a \cdot b = 1(mod \phi(n))$, με $gcd(b, \phi(n)) = 1$ και $\phi(n) = (p-1) \cdot (q-1)$.

Για την κρυπτογράφηση έχουμε ότι:

- Το δημόσιο κλειδί είναι $K_p = (n, b)$.
- Για τη συνάρτηση κρυπτογράφησης $e_{K_p} \in \mathcal{E}$ έχουμε ότι

$$\forall x \in \mathbb{Z}_n \quad \exists y \in \mathbb{Z}_n \quad \text{με} \quad y = e_{K_p}(x) = x^b (mod\ n)$$

Ενώ για την αποκρυπτογράφηση έχουμε ότι:

- Το ιδιοτικό κλειδί είναι $K_{pr} = (p, q, a)$.
- Για τη συνάρτηση αποκρυπτογράφησης $d_{K_{pr}} \in \mathcal{D}$ έχουμε ότι:

$$\forall y \in \mathbb{Z}_n \quad \exists x \in \mathbb{Z}_n \quad \text{με} \quad x = d_{K_{pr}}(y) = y^a (mod\ n)$$

Κρυπτοσύστημα ELGamal στο $\mathbb{Z}_p^\star$

Η ασφάλεια του ELGamal στο $\mathbb{Z}_p^\star$, βασίζεται στη δυσκολία επίλυσης του μαθηματικού προβλήματος του διακριτού λογάριθμου σε μια υποομάδα του $\mathbb{Z}_p^\star$.

Ο ELGamal στο $\mathbb{Z}_p^\star$ είναι η διατεταγμένη πεντάδα

$$(\mathcal{P} = \mathbb{Z}_p^\star, C = \mathbb{Z}_p^\star \times \mathbb{Z}_p^\star, \mathcal{K} = \{(p, a, k, b)\}, \mathcal{E}, \mathcal{D}).$$

όπου:

- Ο p είναι πρώτος αριθμός.
- Το $a \in \mathbb{Z}_p^\star$ είναι ένας γεννήτορας της πολλαπλασιαστικής ομάδας.
- $b = a^k (mod p)$, για κάποιο $b \in < a >$.

Για την κρυπτογράφηση έχουμε ότι:

- Το δημόσιο κλειδί είναι $K_p = (p, a, b)$.
- Για τη συνάρτηση κρυπτογράφησης $e_{K_p} \in \mathcal{E}$ έχουμε ότι

$$\forall x \in \mathbb{Z}_p^\star \quad \exists (y_1, y_2) \in \mathbb{Z}_p^\star \times \mathbb{Z}_p^\star \quad \text{με} \quad (y_1, y_2) = e_{K_p}(x, s), \quad s \in \mathbb{Z}_{p-1}$$
$$\text{και} \quad y_1 = a^s mod p \quad y_2 = x b^s mod p$$

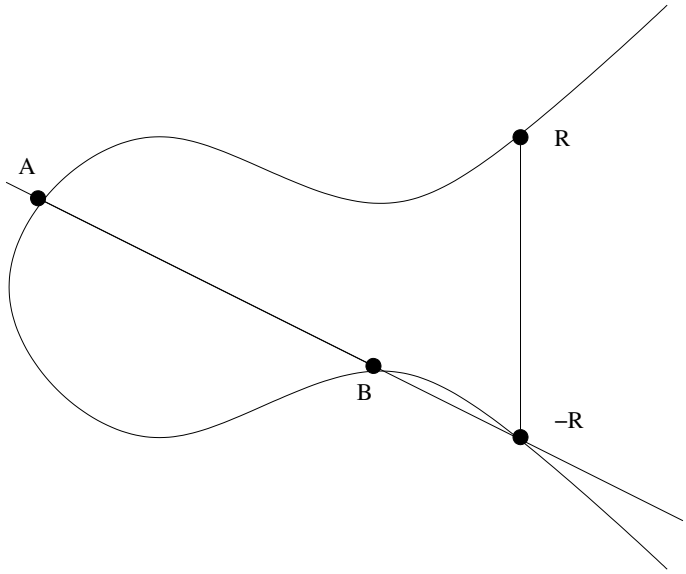
Ενώ για την αποκρυπτογράφηση έχουμε ότι:

- Το ιδιωτικό κλειδί είναι $K_{pr} = k$.
- Για τη συνάρτηση αποκρυπτογράφησης $d_{K_{pr}} \in \mathcal{D}$ έχουμε ότι:

$$\forall (y_1, y_2) \in \mathbb{Z}_p^\star \times \mathbb{Z}_p^\star \quad \exists x \in \mathbb{Z}_p^\star \quad \text{με} \quad x = d_K(y_1, y_2) = y_2 (y_1^k)^{-1} (mod p)$$

Κρυπτοσύστημα ELGamal στις Ελλειπτικές Καμπύλες

Ο ELGamal σε ελλειπτικές καμπύλες είναι ίσως η πιο σημαντική κατηγορία κρυπτογράφησης στο δίκτυο του bitcoin καθώς είναι αυτή που χρησιμοποιείται κυρίως. Αρχικά θα δούμε τις ελλειπτικές καμπύλες και έπειτα θα δούμε τον τρόπο με τον οποίο χρησιμοποιούνται στον ELGamal.



Σχήμα 1.2: Ελλειπτική Καμπύλη

Οι ελλειπτικές καμπύλες που χρησιμοποιούμε στην κρυπτογραφία είναι ορισμένες σε ένα σώμα $\mathbb{Z}_p$, όπου το p είναι πρώτος αριθμός και έχουν τύπο

$$EC : y^2 = x^3 + ax + b \quad \text{με} \quad 4a^3 + 27b^2 \neq 0 (mod\ p)$$

Παίρνουμε ένα σημείο στην καμπύλη, έστω $P \in EC(\mathbb{Z}_p)$ και κατασκευάζουμε την κυκλική υποομάδα $< P >$ ως εξής:

- Σαν πράξη δύο στοιχείων $Q_1, Q_2 \in < P >$ ορίζουμε το $Q_1 + Q_2 = R$, με $-R$ να είναι το συνευθειακό σημείο των Q_1, Q_2 πάνω στην ελλειπτική καμπύλη και R το συμμετρικό του ως προς τον άξονα του x, όπως φαίνεται στο Σχήμα 1.2. Με αυτόν τον τρόπο εξασφαλίζουμε ότι η πράξη θα είναι κλειστή.
- $\forall Q = (x, y) \in < P > \exists -Q = (x, -y) \in < P >$, δηλαδή το αντίθετο σημείο κάθε σημείου είναι το συμμετρικό του ως προς τον άξονα xx'.
- Το ουδέτερο σημείο συμβολίζεται με $O_\infty = Q + (-Q), \quad \forall Q \in < P >$, δηλαδή το σημείο στο άπειρο (δηλαδή το σημείο στο οποίο συναντιούνται όλες οι παράλληλες προς τον άξονα yy' ευθείες).

Βάση του ορισμού της παραπάνω κυκλικής υποομάδας, μπορούμε να ορίσουμε το πρόβλημα του διακριτού λογάριθμου πάνω στην ελλειπτική καμπύλη $EC(\mathbb{Z}_p)$.

Το κρυπτοσύστημα είναι η διατεταγμένη πεντάδα

$$(\mathcal{P} = EC(\mathbb{Z}_p), \mathcal{C} = EC(\mathbb{Z}_p) \times EC(\mathbb{Z}_p), \mathcal{K} = \{(EC(\mathbb{Z}_p), P, n, k, Q)\}, \mathcal{E}, \mathcal{D}),$$

όπου:

- Ο $p > 3$.
- Το $P \in EC(\mathbb{Z}_p)$.
- Η $< P >$ είναι κυκλική υποομάδα τάξης n, και n είναι πρώτος αριθμός.
- Το $Q = kP$.

Για την κρυπτογράφηση έχουμε ότι:

- Το δημόσιο κλειδί είναι $K_\delta = (EC(\mathbb{Z}_p), P, n, Q)$.
- Για τη συνάρτηση κρυπτογράφησης $e_{K_\delta} \in \mathcal{E}$ έχουμε ότι

$$\forall x \in EC(\mathbb{Z}_p) \quad \exists (y_1, y_2) \in EC(\mathbb{Z}_p) \times EC(\mathbb{Z}_p) \quad \text{με} \quad (y_1, y_2) = e_{K_\delta}(x, s), \quad s \in \mathbb{Z}_{n-1}$$

$$\text{και} \quad y_1 = sP \quad y_2 = x + sQ.$$

Ενώ για την αποκρυπτογράφηση έχουμε ότι:

- Το ιδιωτικό κλειδί είναι $K_\iota = k$.
- Για τη συνάρτηση αποκρυπτογράφησης $d_{K_\iota} \in \mathcal{D}$ έχουμε ότι:

$$\forall (y_1, y_2) \in EC(\mathbb{Z}_p) \times EC(\mathbb{Z}_p) \quad \exists x \in EC(\mathbb{Z}_p) \quad \text{με} \quad x = d_{K_\iota}(y_1, y_2) = y_2 + k(-y_1)$$

Το ασύμμετρο κρυπτοσύστημα το χρησιμοποιούμε στο δίκτυο του bitcoin για να κατασκευάζουμε διευθύνσεις μέσω των οποίων γίνεται η ψηφιακή υπογραφή. Η ψηφιακή υπογραφή είναι μια διαδικασία στην οποία η χρήση του ζεύγους των κλειδιών είναι διαφορετική από εκείνη της κρυπτογράφησης/αποκρυπτογράφησης.

Με την ψηφιακή υπογραφή γίνεται γνωστό ότι ο ιδιοκτήτης του κρυπτοκειμένου είναι αυτός που έχει το κλειδί της κρυπτογράφησης K_e που αντιστοιχεί στο K_d με το οποίο έχει υπογραφεί το κρυπτοκείμενο. Έτσι κατωχυρώνεται η ιδιοκτησία ενός αντικειμένου στο δίκτυο του bitcoin. Ο ορισμός της ψηφιακής υπογραφής δίνεται παρακάτω.

Ορισμός 1.2.2 *Στο κρυπτοσύστημα του ELGamal στο $\mathbb{Z}_p^\star$ ορίζουμε του αλγόριθμο της ψηφιακής υπογραφής ως*

$$sig_K(x, w) = (s, d)$$

όπου $s = a^w (mod\ p)$, για $w \in \mathbb{Z}_{p-1}^\star$ και $d = (x - ks)w^{-1} (mod\ (p-1))$. Για $x, s \in \mathbb{Z}_p^\star$ και $d \in \mathbb{Z}_{p-1}^\star$, ορίζουμε του αλγόριθμο επιβεβαίωσης

$$ver_K(x, (s, d)) = true \Leftrightarrow b^s s^d = a^x (mod\ p)$$

Για την καλύτερη κατανόηση παρουσιάζεται το παρακάτω παράδειγμα [15].

Παράδειγμα 1.2.1 *Έστω Α και Β δύο χρήστες ενός δικτύου επικοινωνίας. Ο Α επιθυμεί να υπογράψει το μήνυμα $x = 80$ με την μέθοδο ELGamal και να το στείλει στον χρήστη Β. Για να το κάνει αυτό ο Α θα πρέπει πριν από οποιαδήποτε διαδικασία, να δημιουργήσει:*

1. *Το ιδιωτικό κλειδί, που θα πρέπει να γνωρίζει μόνο ο ίδιος και θα χρησιμοποιηθεί για την υπογραφή του μηνύματος.*

2. *Το δημόσιο κλειδί που θα στείλει ο Α στον Β και θα χρησιμοποιηθεί για την επιβεβαίωση της υπογραφής του μηνύματος, από τον χρήστη Β.*

Τα κλειδιά παράγονται μέσω ενός αλγόριθμου παραγωγής κλειδιών. Αυτός ο αλγόριθμος παράγει ζεύγη από ιδιωτικά και δημόσια κλειδιά για όλους τους χρήστες ενός δικτύου. Έστω ότι ο αλγόριθμος για αυτό το παράδειγμα παρήγαγε την τετράδα $(p, a, k, b) = (263, 5, 25, 240)$, όπου k το ιδιωτικό κλειδί και (p, a, b) το δημόσιο κλειδί του Α. Έστω επίσης ο τυχαίος αριθμός $w = 37$ που έχει αντίστροφο στο $\mathbb{Z}_p$, ο οποίος παράγεται και αυτός από τον παραπάνω αλγόριθμο.

Σύμφωνα με τον ορισμό 1.2.2, ο Α υπογράφει το μήνυμα x χρησιμοποιώντας τον αλγόριθμο υπογραφής sig_K. Έχουμε λοιπόν ότι:

$$sig_K(x, w) = (s, d) \Rightarrow s = a^w \pmod{(p-1)}$$

$$\Rightarrow d = (80 - 25 \cdot 251) \cdot 85 \pmod{262} \Rightarrow d = 45 (mod\ 262)$$

Μετά από τον υπολογισμό των (a, b), ο Α υπέγραψε το μήνυμα x και μπορεί να στείλει το $(x, (s, d))$ στον Β.

Ο Β τώρα από τη δική του μεριά, θα λάβει το ζευγάρι $(x, (s, d))$ και για να επιβεβαιώσει ότι το μήνυμα x έχει υπογραφεί από τον Α, θα πρέπει να χρησιμοποιήσει τον αλγόριθμο επιβεβαίωσης ver_K. Ο ver_K ουσιαστικά επιβεβαιώνει την ισότητα $b^s s^d = a^x (mod p)$. Στο παράδειγμα μας

$$240^{251} \cdot 251^{45} (mod\ 263) = 5^{80} (mod\ 263) \Leftrightarrow 35 = 35 \Rightarrow ver_K(x, y) = true$$

Για να δούμε τον τρόπο με τον οποίον κατασκευάζονται οι διευθύνσεις, θα χρειαστεί να δούμε πρώτα μια άλλη πολύ σημαντική έννοια, αυτή των συναρτήσεων κατακερματισμού ή αλλιώς hash functions, η παρουσίαση των οποίων ακολουθεί.

1.2.2 Συναρτήσεις Κατακερματισμού

Οι συναρτήσεις κατακερματισμού (hash functions) είναι αναπόσπαστο κομμάτι του δικτύου του bitcoin. Χρησιμοποιούνται κατά κόρων και σχεδόν παντού. Στην πραγματικότητα, ολόκληρο το κρυπτονόμισμα είναι μια αλυσίδα από υπολογισμούς συναρτήσεων κατακερματισμού. Ας δούμε όμως για αρχή τι είναι μια συνάρτηση κατακερματισμού.

Ορισμός 1.2.3 *Έστω ένα κρυπτοσύστημα που αποτελείται από τη διατεταγμένη πεντάδα*

$$(\mathcal{P}, C, \mathcal{K}, \mathcal{E}, \mathcal{D})$$

Αν:

- *Το $\mathcal{P}$ είναι το σύνολο όλων των απλών κειμένων.*
- *Το C είναι το σύνολο όλων των κρυπτοκειμένων.*
- *Το $\mathcal{E}$ είναι όλο το σύνολο των συναρτήσεων κρυπτογράφησης.*
- *Το $\mathcal{K} = \mathcal{D} = \emptyset$*

Τότε κάθε συνάρτηση $h \in \mathcal{E}$ ονομάζεται συνάρτηση κατακερματισμού (hash function) και το σύνολο C είναι το σύνολο όλων των δυνατών τιμών κατακερματισμού (hash values). Επίσης έχουμε ότι $\forall x \in \mathcal{P} \quad \exists y \in C \quad$ με $\quad y = h(x)$ με το y να έχει σταθερό πλήθος στοιχείων για οποιαδήποτε είσοδο x της συνάρτησης.

Από τον παραπάνω ορισμό προκύπτουν κάποιες χρήσιμες ιδιότητες για τις συναρτήσεις κατακερματισμού. Οι παρακάτω ιδιότητες αποτελούν και τον λόγο που είναι τόσο σημαντικές για το δίκτυο bitcoin, αλλά θα αναφερθούμε στη χρησιμότητά τους αφού τις αναφέρουμε πρώτα.

Ιδιότητες 1.2.1 *Κάθε συνάρτηση κατακερματισμού ικανοποιεί τα παρακάτω:*

1. ***Είναι μη αντιστρέψιμες συναρτήσεις**, δηλαδή αν κάποιος γνωρίζει την τιμή μιας τέτοιας συνάρτησης είναι υπολογιστικά δύσκολο να βρει την είσοδό της.*
2. *Δεδομένου μιας τιμής εισόδου της συνάρτησης αυτής, είναι υπολογιστικά δύσκολο να βρεθεί μια άλλη τιμή εισόδου με ίδιο αποτέλεσμα.*
3. *Είναι υπολογιστικά δύσκολο να βρεθούν δύο διαφορετικές τιμές που η είσοδός τους στη συνάρτηση θα μας δώσει ίδια τιμή.*

Οι παραπάνω ιδιότητες προκύπτουν από το γεγονός ότι το σύνολο $\mathcal{D} = \emptyset$. Οι συναρτήσεις κατακερματισμού είναι ένας πολύ καλός τρόπος για να βεβαιωθεί κανείς ότι με την αποστολή ενός κειμένου αυτό δεν έχει υποστεί κάποια αλλαγή. Η διαδικασία είναι ότι αν έχουμε ένα απλό κείμενο x το περνάμε από μια συνάρτηση κατακερματισμού h και το αποτέλεσμα $y = h(x)$ το στέλνουμε μαζί με το απλό κείμενο. Οπότε στέλνοντας το ζεύγος (x, y), ο παραλήπτης αυτό που έχει να κάνει είναι να περάσει το απλό κείμενο από την ίδια συνάρτηση. Αν το αποτέλεσμα $y' \neq y$ τότε βάση της τρίτης ιδιότητας, το x έχει υποστεί κάποια αλλαγή και δεν ήταν το κείμενο που αρχικά στάλθηκε. Αν το αποτέλεσμα $y' = y$ τότε το μήνυμα δεν έχει υποστεί καμία αλλαγή και έτσι ο παραλήπτης είναι σίγουρος για την εγκυρότητα του. Ας δώσουμε τώρα τον ορισμό των ψηφιακών υπογραφών.

Ορισμός 1.2.4 *Ένα σχήμα κατασκευής ψηφιακών υπογραφών είναι μια διατεταγμένη πεντάδα*

$$(\mathcal{M}, \mathcal{A}, \mathcal{K}, \mathcal{S}, \mathcal{V})$$

όπου:

1. *Το $\mathcal{M}$ είναι το πεπερασμένο σύνολο όλων των δυνατών μηνυμάτων.*
2. *Το $\mathcal{A}$ είναι το πεπερασμένο σύνολο των ψηφιακών υπογραφών.*
3. *Το $\mathcal{K}$ είναι το πεπερασμένο σύνολο των κλειδιών.*

Για κάθε κλειδί $K \in \mathcal{K}$ υπάρχουν συναρτήσεις υπογραφής $sig_K : \mathcal{M} \mapsto \mathcal{A}$ και επαλήθευσεις $ver_K : \mathcal{M} \times \mathcal{A} \mapsto \{true, false\}$ τέτοιες ώστε να ισχύει:

$$ver_K(x, y) = \begin{cases} true & \text{αν} \quad y = sig_K(x) \\ false & \text{αν} \quad y \neq sig_K(x) \end{cases}$$

Το διατεταγμενο ζεύγος (x, y) με $x \in \mathcal{M}$ και $y \in \mathcal{A}$ ονομάζεται υπογεγραμμένο μήνυμα.

Το δίκτυο του bitcoin χρησιμοποιεί δύο τέτοιες συναρτήσεις, την SHA-256 και την RIPEMD-160. Παρακάτω θα εξετάσουμε τον μηχανισμό με τον οποίο λειτουργούν οι δύο συναρτήσεις αυτές.

Η συνάρτηση SHA-256

Η SHA-256 είναι η πιο σημαντική συνάρτηση κατακερματισμού του δικτύου bitcoin. Χρησιμοποιείται σχεδόν παντού. Η SHA-256 όπως υποδηλώνει και το όνομά της παράγει μια κατακερματισμένη τιμή μήκους των 256-bit και έτσι έχει ένα τεράστιο εύρος πιθανών τιμών. Ας δούμε όμως αρχικά πως λειτουργεί.

Σαν είσοδο της συνάρτησης έχουμε ένα μήνυμα οσοδήποτε μεγάλο (στην πραγματικότητα, το μήνυμα δεν θα πρέπει να ξεπερνάει σε μέγεθος τα 2^{64}-bit). Κατά την

είσοδο του μηνύματος, αυτό αν έχει δοθεί σε ASCII μορφή, θα μετατρέπει σε έναν μεγάλο μη προσημασμένο ακέραιο. Έπειτα ξεκινάει η διαδικασία του **padding**.

- Το μήνυμα αρχικά αποθηκεύεται στη μνήμη με ένα 1 στο τέλος του. Η αποθήκευση γίνεται κατά Big Endian, δηλαδή το πιο σημαντικό ψηφίο είναι αυτό που αποθηκεύεται πρώτο.
- Στο τέλος του μηνύματος (όπως έχει διαμορφωθεί μετά το προηγούμενο βήμα) συμπληρώνονται k μηδενικά, με $k + l + 1 = 448 \ mod \ 512$, όπου $l < 2^64$ το μέγεθος του εισερχόμενου μηνύματος.
- Μετά την προσθήκη των μηδενικών, στα τελευταία 64-bit προστίθεται και το μέγεθος του μηνύματος.

Παρατηρούμε ότι το επαυξημένο μήνυμα θα πρέπει να είναι κάποιο πολλαπλάσιο του 512. Το παρακάτω παράδειγμα θα μας βοηθήσει στη κατανόηση του padding.

Παράδειγμα 1.2.2 *Έστω ότι η είσοδος της SHA-256 είναι η συμβολοσειρά "abc". Καθώς είναι χαρακτήρες, θα γίνει η μετατροπή του σε ακέραιο. Όλοι οι αριθμοί που θα παρουσιαστούν παρακάτω θα είναι σε δεκαεξαδική μορφή.*

1. *Μετατρέπουμε το μήνυμα*
$$"abc" \rightarrow (616263)_{16}$$
2. *Του προσθέτουμε τη μονάδα στο τέλος*
$$61626380$$
3. *Βρίσκουμε το μέγεθος του μηνύματος $l = 24$-bit άρα $l = (18)_{16}$.*
4. *Υπολογίζουμε το k, $k + l + 1 = 448 mod 512 \Rightarrow k + 25 = 448 mod 512 \Rightarrow k = 423 mod 512$. Άρα στο παραπάνω μήνυμα θα προστεθούν 423 μηδενικά και θα έχουμε:*
$$\begin{array}{cccc} 61626380 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & & \end{array}$$
5. *Στα τελευταία 64-bit προστίθεται το μέγεθος του μηνύματος l και το padding έχει τελειώσει. Άρα το επαυξημένο μήνυμα θα είναι:*
$$\begin{array}{cccc} 61626380 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000018 \end{array}$$

Μεταβλητές	Αρχικές Τιμές
a	$6A09E667$
b	$BB67AE85$
c	$3C6EF372$
d	$A54FF53A$
e	$510E527F$
f	$9B05688C$
g	$1F83D9AB$
h	$5BE0CD19$

Πίνακας 1.1: Αρχικές Τιμές της SHA-256

Αφού τελειώσει η διαδικασία του padding, ξεκινάει η διαδικασία της συμπίεσης του μηνύματος ή αλλιώς η διαδικασία υπολογισμού της τιμής κατακερματισμού. Η SHA-256 χρησιμοποιεί κάποιες σταθερές όπως φαίνονται στον Πίνακα 1.2 καθώς και κάποιες μεταβλητές οι οποίες αρχικοποιούνται με τις τιμές του Πίνακα 1.1. Το μήνυμα χωρίζεται σε τμήματα των 512-bit και από εκεί προκύπτουν δεκαέξι 32-bit λέξεις καθώς και άλλες τριανταοχτώ 32-bit λέξεις που προκύπτουν από τις προηγούμενες δεκαέξι χρησιμοποιώντας πάνω σε αυτές τις συναρτήσεις του Πίνακα 1.3, όπου $ROTR^n(w) = (w >> n) \vee (w << (32 - n))$ και $SHR^n(w) = w >> n$.

Αφού έχουν δημιουργηθεί όλες οι λέξεις και έχουν αρχικοποιηθεί οι μεταβλητές, το σύστημα χρησιμοποιεί τις συναρτήσεις του Πίνακα 1.4και κάνει τον υπολογισμό της τιμής κατακερματισμού όπως φαίνεται στο Σχήμα 1.3. Το αποτέλεσμα θα είναι οχτώ 32-bit λέξεις τις οποίες τις συμπιέζουμε σε μία 256-bit λέξη. Αυτή είναι και η τιμή κατακερματισμού της SHA-256.

Η Συνάρτηση RIPEMD-160

Η RIPEMD-160 είναι μια συνάρτηση κατακερματισμού η οποία χρησιμοποιείται μόνο για την κατασκευή διευθύνσεων στο δίκτυο του bitcoin, μια διαδικασία που θα δούμε αργότερα. Αρχικά θα δούμε πως δουλεύει η συγκεκριμένη συνάρτηση.

Το πρώτο κομμάτι αφορά την είσοδο ενός κειμένου στη συνάρτηση. Αρχικά, η συνάρτηση παίρνει ως είσοδο ένα κείμενο οσοδήποτε μεγάλο και οποιουδήποτε τύπου. Αμέσως μετά αρχίζει το **padding**.

- Στο τέλος του μηνύματος προστίθεται η μονάδα.
- Το μήνυμα αποθηκεύεται στη μνήμη κατά Little Endian αρχιτεκτονική, δηλαδή το λιγότερο σημαντικό ψηφίο αποθηκεύεται πρώτο.
- Στο μήνυμα προστίθενται k μηδενικά έτσι ώστε $k + l + 1 = 448 mod\ 512$, όπου με l συμβολίζουμε το μέγεθος του μηνύματος το οποίο είναι $l < 2^{64}$.

428*A*2*F*98	71374491	*B*5*C*0*FBCF*	*E*9*B*5*DBA*5
3956*C*25*B*	59*F*111*F*1	923*F*82*A*4	*AB*1*C*5*ED*5
*D*807*AA*98	12835*B*01	243185*BE*	550*C*7*DC*3
72*BE*5*D*74	80*DEB*1*FE*	9*BDC*06*A*7	*C*19*BF*174
*E*49*B*69*C*1	*EFBE*4786	0*FC*19*DC*6	240*CA*1*CC*
2*DE*92*C*6*F*	4*A*7484*AA*	5*CB*0*A*9*DC*	76*f*988*DA*
983*E*5152	*A*831*C*66*D*	*B*00327*C*8	*BF*597*FC*7
*C*6*E*00*BF*3	*D*5*A*79147	06*CA*6351	14292967
27*B*70*A*85	2*E*1*B*2138	4*D*2*C*6*DFC*	53380*D*13
650*A*7354	766*A*0*ABB*	81*C*2*C*92*E*	92722*C*85
*A*2*BFE*8*A*1	*A*81*A*664*B*	*C*24*B*8*B*70	*C*76*C*51*A*3
*D*192*E*819	*D*6990624	*F*40*E*3585	106*AA*070
19*A*4*C*116	1*E*376*C*08	2748774*C*	34*B*0*BCB*5
391*C*0*CB*3	4*ED*8*AA*4*A*	5*B*9*CCA*4*F*	682*E*6*FF*3
748*F*82*EE*	78*A*5636*F*	84*C*87814	8*CC*70208
90*BEFFFA*	*A*4506*CEB*	*BEF*9*A*3*F*7	*C*67178*F*2

Πίνακας 1.2: Σταθερές της SHA-256

Συναρτήσεις σε Λέξεις w	Τύπος
$SSIG0$	$ROTR^7(w) \oplus ROTR^{18}(w) \oplus SHR^3(w)$
$SSIG1$	$ROTR^{17}(w) \oplus ROTR^{19}(w) \oplus SHR^{10}(w)$

Πίνακας 1.3: Συναρτήσεις για τη Δημιουργία Λέξεων στη SHA-256

- Στα τελευταία 64-bit προστίθεται το μέγεθος του μηνύματος πάλι κατά Little Endian μορφή.
- Το τελικό, επαυξημένο μήνυμα θα έχει μέγεθος πολλαπλάσιο του 512.

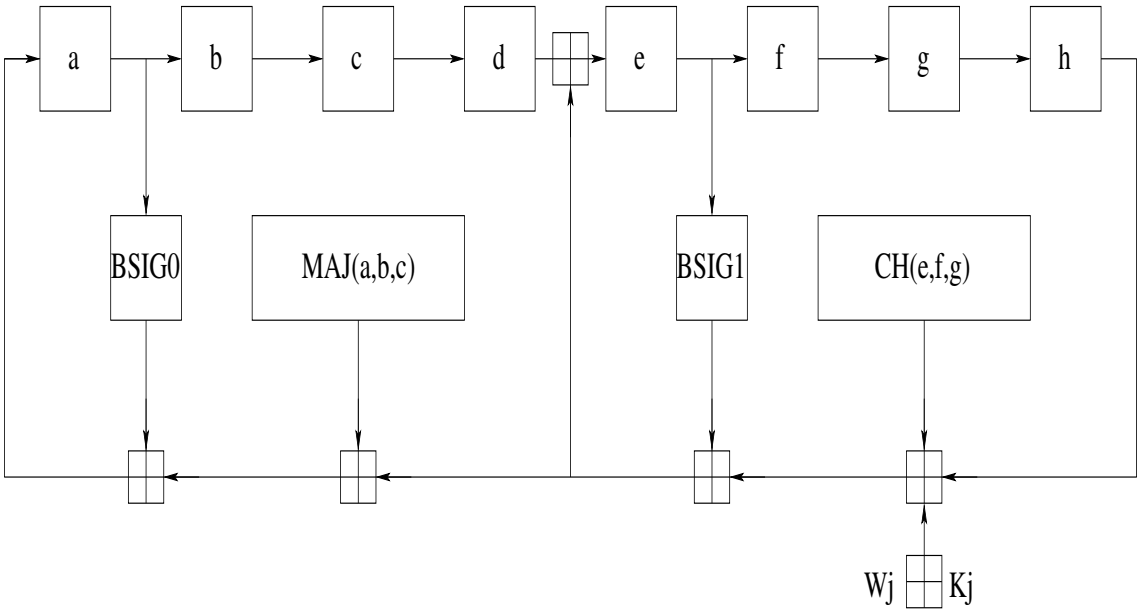
Παράδειγμα 1.2.3 *Το padding γίνεται όπως στο παράδειγμα με το padding της SHA-256, με τη μόνη διαφορά ότι είναι γραμμένο σε Little Endian μορφή. Άρα για το μήνυμα "abc" το επεκταμένο μήνυμα θα είναι:*

80636261 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000018 00000000

Από τη στιγμή που τελειώνει το padding, αρχίζει ο υπολογισμός της τιμής κατακερματισμού. Το σύστημα χωρίζει τις μεταβλητές του σε δεξιές και αριστερές και εκτελεί παράλληλα δύο υπολογισμούς. Κάνει αρχικοποίηση των δεξιών και αριστερών τιμών με την ίδια σταθερά, όπως φαίνεται στον Πίνακα 1.5. Κόβει το μήνυμα ανά

Συναρτήσεις σε Λέξεις w	Τύπος
$BSIG0$	$ROTR^2(w) \oplus ROTR^{13}(w) \oplus ROTR^{22}(w)$
$BSIG1$	$ROTR^6(w) \oplus ROTR^{11}(w) \oplus ROTR^{25}(w)$
MAJ	$(w_1 \wedge w_2) \oplus (w_1 \wedge w_3) \oplus (w_2 \wedge w_3)$
CH	$(w_1 \wedge w_2) \oplus (\neg w_1 \wedge w_3)$

Πίνακας 1.4: Συναρτήσεις της SHA-256



Σχήμα 1.3: Συνάρτηση Συμπίεσης της SHA-256

512-bit σε δεκαέξι 32-bit λέξεις και χρησιμοποιώντας τις σταθερές του Πίνακα 1.6 καθώς και τις συναρτήσεις του Πίνακα 1.7 κάνει τους υπολογισμούς.

Οι υπολογισμοί γίνονται σε πέντε γύρους, τόσοι όσες είναι και οι σταθερές και οι συναρτήσεις που χρησιμοποιούνται. Οι επιλογές τον λέξεων γίνεται σύμφωνα με δύο διαφορετικές συναρτήσεις μετάθεσης. Η πρώτη είναι η

$$\pi(i) = 9i + 5 \ (mod\ 16), \quad i = 0, ..., 15$$

Η δεύτερη συνάρτηση είναι όπως φαίνεται στον Πίνακα 1.8. Τέλος, η επιλογή των λέξεων κάθε φορά γίνεται ανάλογα τον γύρο, όπως φαίνεται στον Πίνακα 1.9.

Αφού τελειώσουν οι υπολογισμοί, όπως φαίνεται στο Σχήμα 1.4 [16] προκείπτουν πέντε 32-bit λέξεις. Καθώς, είπαμε ότι είναι σε Little Endian μορφή, τις μετατρέπουμε σε Big Endian και τις συμπιέζουμε σε μία 160-bit λέξη. Αυτή είναι και η τιμή κατακερματισμού της RIPEMD-160.

Αριστερές Μεταβλητές	Δεξιές Μεταβλητές	Αρχικοποίηση
a	a'	67452301
b	b'	$EFCDAB89$
c	c'	$98BADCFE$
d	d'	10325476
e	e'	$C3D2E1F0$

Πίνακας 1.5: Αρχικές Τιμές της RIPEMD-160

Αριστερές Σταθερές	Τιμές	Δεξιές Σταθερές	Τιμές
K_1	00000000	K'_1	$50A28BE6$
K_2	$5A827999$	K'_2	$5C4DD124$
K_3	$6ED9EBA1$	K'_3	$6D703EF3$
K_4	$8F1BBCDC$	K'_4	$7A6D76E9$
K_5	$A953FD4E$	K'_5	00000000

Πίνακας 1.6: Σταθερές της RIPEMD-160

Το Δένδρο Merkle

Στο δίκτυο του bitcoin έχει βρει εφαρμογή και το Merkle Tree με έναν εφάνταστο τρόπο χρησιμοποιώντας τη ρίζα του. Πριν, όμως, δούμε πως χρησιμοποιείται θα δούμε τι είναι ένα Merkle Tree.

Ορισμός 1.2.5 *Merkle Tree ονομάζεται ένα δένδρο με ρίζα (Merkle Root) του οποίου τα φύλλα έχουν ως ετικέτα τιμές κατακερματισμού, ενώ κάθε εσωτερικός κόμβος είναι μια τιμή κατακερματισμού που προήλθε από μια συνάρτηση κατακερματισμού που είχε σαν είσοδο τις τιμές κατακερματισού των παιδιών του.*

Στην περίπτωση του bitcoin, το δένδρο που κατασκευάζεται είναι δυαδικό. Η διαδικασία κατασκευής του είναι η εξής: Ανά δύο φύλλα, παίρνουμε τις hash τιμές τους και τις χρησιμοποιούμε σαν είσοδο στη συνάρτηση κατακερματισμού. Η καινούργια τιμή κατακερματισμού που προκύπτει είναι ένας ενδιάμεσος κόμβος. Η διαδικασία αυτή συνεχίζεται μέχρις ότου υπολογίσουμε μια τελική τιμή κατακερματισμού. Έτσι έχουμε φτιάξει ένα Merkle Tree και η ρίζα του δέντρου το λεγόμενο Merkle Root είναι η τελική τιμή κατακερματισμού.

Η διαδικασία φαίνεται στο Σχήμα 1.5 [12]. Να σημειωθεί πως η συνάρτηση κατακερματισμού που χρησιμοποιείται είναι η $(SHA-256)^2$, δηλαδή αν υποθέσουμε ότι έχουμε μια τιμή x, τότε το $y = SHA-256(SHA-256(x))$. Περνάμε το απλό κείμενο από τη $SHA-256$ και το αποτέλεσμά της το περνάμε πάλι από τη $SHA-256$ και έτσι παίρνουμε την προαναφερθείσα hash τιμή.

Παρακάτω θα μιλήσουμε για την τεχνολογία που βρήκε τεράστια εφαρμογή στο

Συναρτήσεις	Τιμές
f_1	$x \oplus y \oplus z$
f_2	$(x \wedge y) \vee (\neg x \wedge z)$
f_3	$(x \vee \neg y) \oplus z$
f_4	$(x \wedge z) \vee (y \wedge \neg z)$
f_5	$x \oplus (y \vee \neg z)$

Πίνακας 1.7: Συναρτήσεις της RIPEMD-160

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$p(i)$	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8

Πίνακας 1.8: Η Μετάθεση p της RIPEMD-160

bitcoin, αυτή της blockchain. Θα δούμε τι ακριβώς είναι αυτή η τεχνολογία, πως εφαρμόστηκε και πως όλες οι παραπάνω συναρτήσεις συνδέονται μαζί της.

1.3 BlockChain

Κυρίαρχο στοιχείο σε ένα bitcoin δίκτυο είναι οι συναλλαγές. Τι είναι μια συναλλαγή;

Ορισμός 1.3.1 *Με τον όρο συναλλαγή, εννοούμε τη μεταφορά μιας ποσότητας από bitcoin μέσω διάδοσης στο δίκτυο και τη καταγραφή της σε ένα μπλοκ μιας blockchain.*

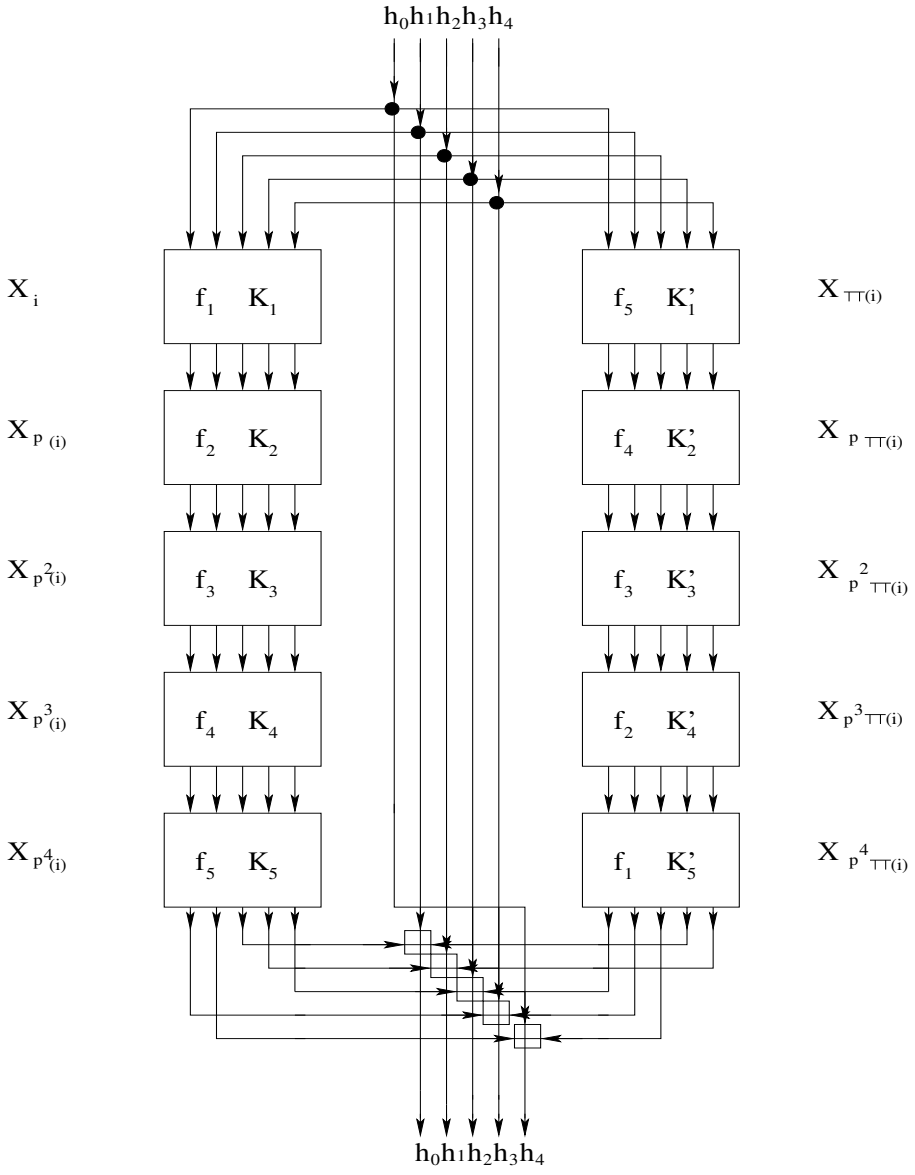
Τι είναι όμως η τεχνολογία blockchain; Το blockchain είναι, όπως λέει και η λέξη, μια αλυσίδα από τμήματα. Η διασύνδεση των τμημάτων γίνεται με τη χρήση μιας συνάρτησης κατακερματισμού.

Ένα τμήμα μιας αλυσίδας blockchain μπορεί κάποιος να το φανταστεί σαν ένα κομμάτι χαρτί στο οποίο μπορεί να εγγραφεί σταθερός αριθμός εγγραφών. Ένα τμήμα συντίθεται από την επικεφαλίδα και το κύριό του μέρος. Στο κύριο μέρος εγγράφονται συναλλαγές που έχουν γίνει από τους χρήστες, ενώ η επικεφαλίδα είναι αυτή που θα συσχετίσει το τμήμα με το προηγούμενο της αλυσίδας blockchain. Από αυτή την αλυσίδα, μπορούμε εύκολα να δημιουργήσουμε ένα δίκτυο, αυτό των συναλλαγών.

Συγκεκριμένα, κάθε συναλλαγή γίνεται είσοδος σε μια συνάρτηση κατακερματισμού και η τιμή της συνάρτησης γίνεται επίσης είσοδο ξανά στην συνάρτηση κατακερματισμού και το τελικό αποτέλεσμα είναι αυτό που αποθηκεύεται στο τμήμα (δηλαδή $(SHA-256)^2$). Στο κύριο μέρος του μπλοκ, βρίσκει εφαρμογή το Merkle Tree. Ως φύλλα του δένδρου έχουμε τις τιμές της συνάρτησης κατακερματισμού για

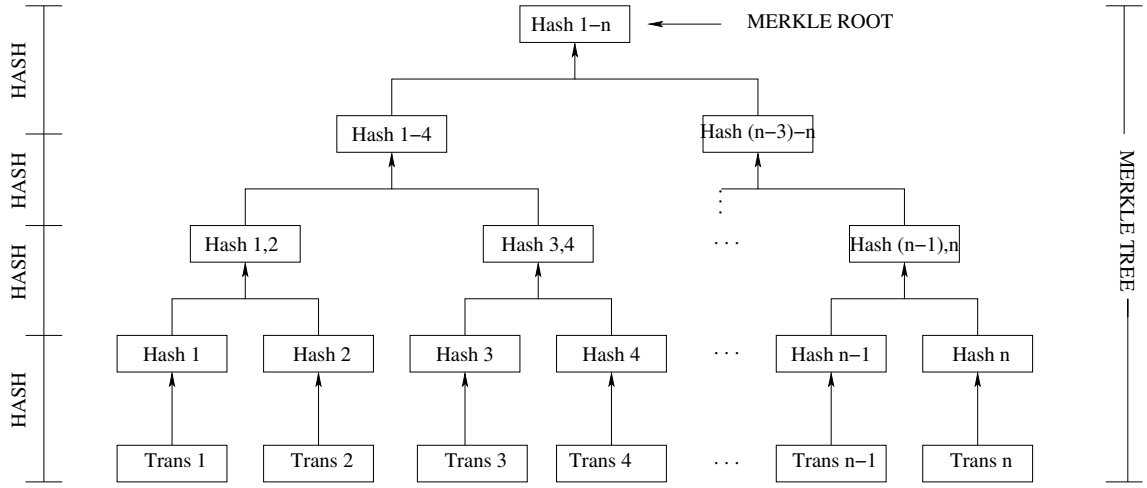
Αριστερή/Δεξιά Λέξη	Γύρος 1	Γύρος 2	Γύρος 3	Γύρος 4	Γύρος 5
X	id	p	p^2	p^3	p^4
X'	π	$p\pi$	$p^2\pi$	$p^3\pi$	$p^4\pi$

Πίνακας 1.9: Οι Λέξεις της RIPEMD-160



Σχήμα 1.4: Υπολογισμός Τιμής Κατακερματισμού της RIPEMD-160

κάθε συναλλαγή. Έπειτα ακολουθείται η διαδικασία κατασκευής του δένδρου όπως είδαμε στο προηγούμενο εδάφιο. Η ρίζα αυτού του δένδρου (Merkle Root) είναι αυτή που θα χρησιμοποιηθεί για τον υπολογισμό της επικεφαλίδας του τμήματος.



Σχήμα 1.5: Merkle Tree-Merkle Root

Η επικεφαλίδα είναι ένας αριθμός μήκους 32-byte, ο οποίος είναι η τιμή της συνάρτησης κατακερματισμού με είσοδο τις εξής ποσότητες:

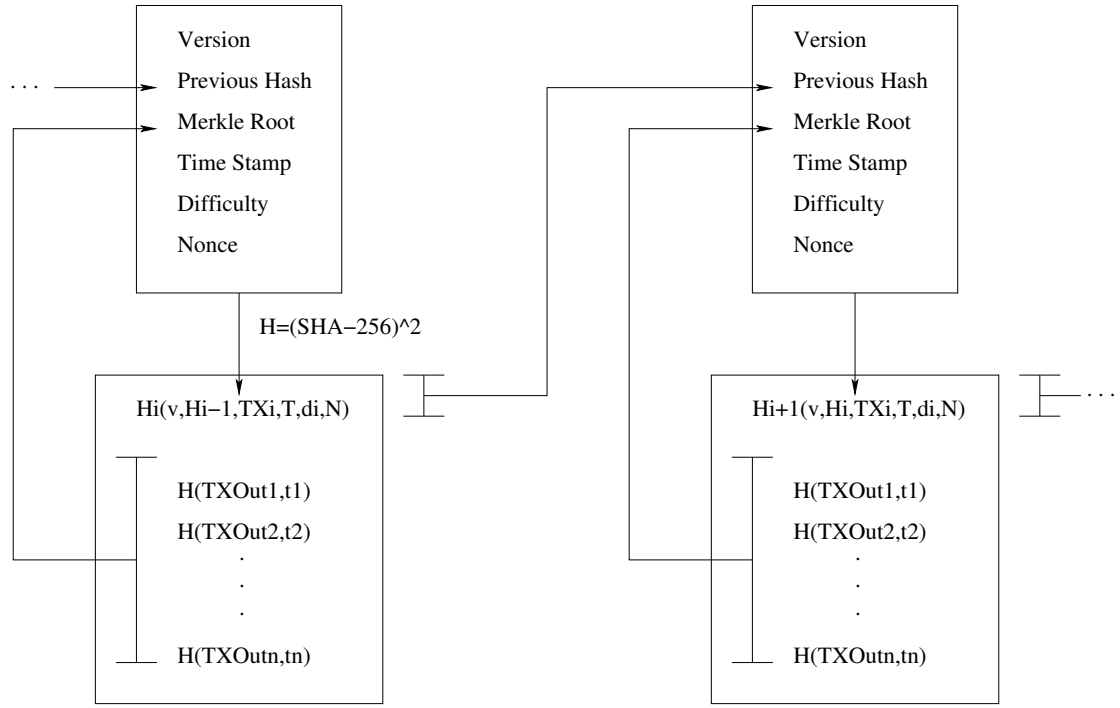
1. v: Η έκδοση του μπλοκ (version) 4-byte.
2. H_{i-1}: Η επικεφαλίδα του προηγούμενου τμήματος μήκους 32-byte.
3. TX_i: Το Merkle Root του δένδρου των συναλλαγών επίσης 32-byte.
4. T: (Time Stamp) έκδοσης του μπλοκ σε δευτερόλεπτα $UNIX$ μήκους 4-byte.
5. d_i: Η δυσκολία του προβλήματος επικύρωσης συναλλαγών (4-byte).
6. N: Ένας τυχαίος αριθμός (nonce) (4-byte).

Άρα η επικεφαλίδα του i-ιοστού μπλοκ είναι [6]:

$$H_i = SHA - 256(SHA - 256(v||H_{i-1}||TX_i||T||d_i||N))$$

Όπως μπορεί να παρατηρήσει κανείς στην επικεφαλίδα έχουμε σαν είσοδο και την επικεφαλίδα του προηγούμενου τμήματος. Κατά αυτόν τον τρόπο δημιουργείται ένας δεσμός ανάμεσα στο τρέχων τμήμα με το προηγούμενο δημιουργώντας μια αλυσίδα (Σχήμα 1.6) της οποίας το αρχικό τμήμα θα μας οδηγήσει στην πρώτη συναλλαγή που έγινε. Με αυτόν τον τρόπο η blockchain λειτουργεί και ως ιστορικό συναλλαγών το οποίο είναι διαθέσιμο στον καθένα.

Παρά το γεγονός ότι η blockchain περιέχει όλη αυτή τη πληροφορία συναλλαγών, το πρόβλημα είναι πως δεν είναι ένα εργαλείο για να επικυρώνει συναλλαγές. Για να επικυρωθεί μια συναλλαγή θα πρέπει τα bitcoin που θα μεταφερθούν από τον



Σχήμα 1.6: Η δομή της blockchain

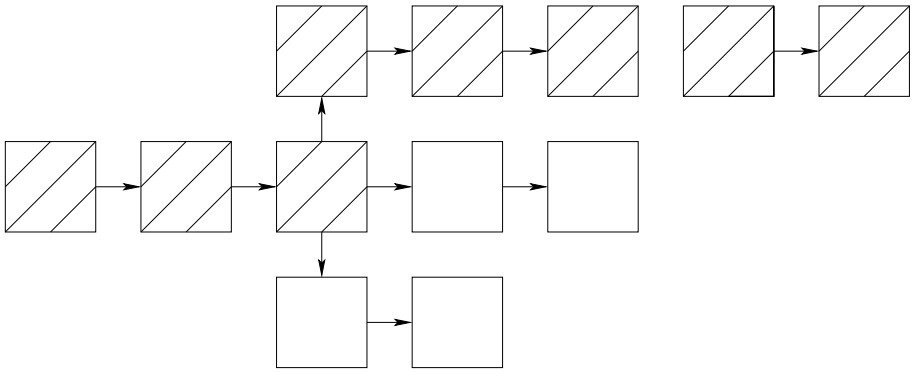
ένα χρήστη στον άλλον, να μπορούν να ″ξοδευτούν″. Εκεί είναι που μπαίνουν άλλοι μηχανισμοί οι οποίοι χρησιμοποιούν την blockchain για να λύσουν το παραπάνω πρόβλημα που θα εξετάσουμε παρακάτω.

1.4 Ηλεκτρονικές Υπηρεσίες με Χρήση Κρυπτονομισμάτων

Το bitcoin δίκτυο μπορεί να χωριστεί σε δύο δίκτυα. Το πρώτο είναι το δίκτυο των συναλλαγών, ενώ το δεύτερο είναι το δίκτυο των χρηστών. Γιατί είναι σημαντικά αυτά τα δίκτυα και γιατί η κατανόηση τους μπορεί να μας οδηγήσει σε συμπεράσματα που δεν περιμέναμε, είναι κάποια από τα ερωτήματα που θα απαντηθούν σε αυτή την ενότητα. Επίσης, θα παρουσιαστούν ο μηχανισμός επικύρωσης συναλλαγής καθώς και τα εργαλεία που χρησιμοποιούνται από τους χρήστες έτσι ώστε οι συναλλαγές τους να είναι ανώνυμες μέσα στο δίκτυο.

1.4.1 Επικύρωση Συναλλαγών

Ο μηχανισμός επικύρωσης συναλλαγών που αναφέρθηκε σε προηγούμενο εδάφιο συντίθεται από ένα σύνολο υπολογιστικών κόμβων (miners) και οι οποίοι αποτελούν



Σχήμα 1.7: Διακλάδωση blockchain

την ″καρδιά″ του bitcoin δικτύου. Είναι αυτοί που προσφέρουν στο σύστημα την επεξεργαστική τους ισχύ για στη δημιουργία καινούριων τμημάτων και την επικύρωση των συναλλαγών που περιέχονται σε αυτά. Αυτό που καλούνται να κάνουν είναι το λεγόμενο Proof-of-Work στο οποίο σημαντική συμμετοχή έχει ο τυχαίος αριθμός (nonce) που χρησιμοποιείται στον υπολογισμό της επικεφαλίδας του τμήματος.

Το nonce είναι ένας αριθμός, τον οποίο πρέπει να προσδιορίσουν οι miners. Αυτός ο προσδιορισμός σχετίζεται με τη δυσκολία d_i του προβλήματος που έχει θέσει το σύστημα και αυτοί καλούνται να λύσουν. Το πρόβλημα που καλούνται οι κόμβοι να επιλύσουν είναι η εύρεση μιας τιμής της συνάρτησης κατακερματισμού $((SHA-256)^2)$ που θα έχει ως είσοδο όλες εκείνες τις τιμές που αποτελούν την επικεφαλίδα H_i του μπλοκ και θα είναι μικρότερη από έναν αριθμό $f(d_i)$ που δίνει το σύστημα. Δηλαδή θα πρέπει να ισχύει ότι:

$$H_i < f(d_i).$$

Το πρόβλημα απαιτεί την εύρεση μιας τιμής κατακερματισμού που θα έχει έναν συγκεκριμένο αριθμό μηδενικών κατ᾽ ελάχιστον. Στον υπολογισμό της επικεφαλίδας ενός τμήματος ο μόνος αριθμός που είναι αυθαίρετος είναι το nonce, και αυτός που θα συμβάλλει στον προσδιορισμό του επθυμητού πλήθους των μηδενικών που θέτει το πρόβλημα. Το πρόβλημα θεωρείται δυσκολότερο όταν ο αριθμός αυτός πρέπει να ξεκινάει με πολλά μηδενικά, ενώ θεωρείται ευκολότερο όταν πρέπει να ξεκινάει με λιγότερα. Αυτό είναι ένα δύσκολο πρόβλημα για να λυθεί και η μόνη προσέγγιση που μπορεί να γίνει είναι αυτή του brute force, δηλαδή να γίνονται διαρκώς τυχαίες δοκιμές μέχρι να βρεθεί αυτός ο αριθμός.

Παρά το γεγονός ότι το Proof-of-Work είναι ένα δύσκολο πρόβλημα για να λυθεί καμιά φορά μπορεί να αποδειχθεί και αδυναμία του συστήματος καθώς το σύστημα δίνει τη δυνατότητα στον καθένα να δημιουργήσει ένα δικό του μπλοκ βάζοντας μέσα ότι συναλλαγές θέλει ο ίδιος και βρίσκοντας τον αριθμό που ικανοποιεί το πρόβλημα που θέτει το σύστημα να δημιουργήσει μια διακλάδωση στην αλυσίδα το λεγόμενο fork [9] (βλέπε Σχήμα 1.7).

Βάση του πρωτοκόλλου του bitcoin δικτύου που θα εξετάσουμε στο εδάφιο 1.4.2, οι miners πάντοτε δέχονται ως σωστό μπλοκ αυτό που είναι το τελευταίο της μεγαλύτερης αλυσίδας. Έτσι αν κάποιος επιχειρήσει να κάνει μια τέτοιου είδους επίθεση στο σύστημα, θα πρέπει διαρκώς να βρίσκει καινούρια μπλοκ για να αυξάνει το μέγεθος της αλυσίδας του. Κάτι τέτοιο είναι απίθανο να συμβεί αν αναλογιστεί κανείς το μέγεθος της επεξεργαστικής ισχύς που απαιτείται.

Ο μόνος τρόπος μια τέτοια επίθεση να πετύχει είναι ο επιτιθέμενος να έχει μεγαλύτερη επεξεργαστική ισχύ από το σύστημα (βλέπε [2]). Παρόλα αυτά, και με τον παράγοντα της τύχης κάποιος μπορεί να βρει ένα τέτοιο τμήμα πιο γρήγορα από τους miners, όμως σίγουρα δε θα μπορέσει να το αναπτύξει γρηγορότερα από αυτούς. Αυτός είναι και ο λόγος που πρέπει κάποιος να περιμένει την εμφάνιση δύο μπορεί και τριών νέων τμημάτων για να θεωρηθεί η συναλλαγή του έγκυρη.

Ας δούμε όμως γιατί μια τέτοια υπόθεση είναι δυνατή. Ας ξεκινήσουμε με την υπόθεση ότι ο επιτιθέμενος θέλει να παράγει μια διακλάδωση στην αλυσίδα και να την αναπτύξει γρηγορότερα από την πραγματική και έγκυρη αλυσίδα. Αρχικά να πούμε πως αυτό δε θα έδινε τον επιτιθέμενο χρήματα. Ο μόνος λόγος που θα επιχειρίσει κάποιος αυτή την επίθεση είναι για να πάρει κάποια χρήματα που έχει ξοδέψει πίσω. Δηλαδή, να αλλάξει κάποια δικιά του συναλλαγή.

Αυτή η διαδικασία ανταγωνισμού μεταξύ της πραγματικής και της ψεύτικης αλυσίδας χαρακτηρίζεται ως Τυχαίος Διωνυμικός Περίπατος. Στην περίπτωση επιτυχίας, δηλαδή η σωστή αλυσίδα να ξεπεράσει τη ψεύτικη, αυξάνει την διαφορά των δύο κατά +1, ενώ σε περίπτωση αποτυχίας, δηλαδή αν ο επιτιθέμενος αυξήσει τη ψεύτικη αλυσίδα πρώτος τότε η διαφορά των δύο πηγαίνει στο -1.

Η πιθανότητα του επιτιθέμενου, να συνεχίζει με ρυθμό ανάπτυξης της αλυσίδας του ίσο με αυτό της πραγματικής αλυσίδας, είναι ανάλογη με αυτή του προβλήματος του Gambler's Ruin (η καταστροφή του τζογαδόρου). Το πρόβλημα είναι το εξής:

Έστω ένας τζογαδόρος με απεριόριστα χρήματα, ξεκινάει να παίζει έχοντας κάποιο έλλειμμα και συνεχίζει να παίζει μέχρι να φτάσει στα λεφτά με τα οποία ξεκίνησε.

Θα επιχειρήσουμε να Υπολογίσουμε την πιθανότητα να φτάσει κάποτε στο αρχικό του ποσό ή στην περίπτωσή μας ο επιτιθέμενος να φτάσει την κανονική αλυσίδα. Έστω ότι:

- p η πιθανότητα ένας miner να βρει το επόμενο τμήμα.
- q η πιθανότητα ο επιτιθέμενος να βρει το επόμενο τμήμα.
- q_z η πιθανότητα ο επιτιθέμενος να φτάσει την πραγματική αλυσίδα όταν βρίσκεται z τμήματα πίσω.

Συνάρτηση Πιθανότητας	Παράμετρος	Μέση Τιμή	Διακύμανση
$(\lambda^k/k!) \cdot e^{-\lambda}$	$\lambda \in \mathbb{R}_+$	λ	λ

Πίνακας 1.10: Κατανομή Poisson

Από τα παραπάνω έχουμε:

$$q_z = \left\{ \begin{array}{ll} 1, & p \leq q \\ (q/p)^z, & p > q \end{array} \right\}.$$

Δεδομένου της υπόθεσης ότι $p > q$, καθώς υποθέσαμε ότι το δίκτυο έχει μεγαλύτερη επεξεργαστική ισχύ από κάθε επιτιθέμενο, η πιθανότητα q_z μειώνεται εκθετικά όσο αυξάνεται το z, δηλαδή το πλήθος των τμημάτων που θα πρέπει να βρει ώστε να φτάσει τη πραγματική αλυσίδα. Έχοντας αυτό κατά νου, προκύπτει το συμπέρασμα ότι αν δεν ξεκινήσει με κάποια τυχερή επιλογή ώστε να βρει το πρώτο μπλοκ γρήγορα, οι πιθανότητες θα στραφούν εναντίον του με ραγδαίο ρυθμό.

Όπως αναφέραμε παραπάνω ο λόγος που γίνεται αυτή η προσπάθεια από τον επιτιθέμενο είναι να κάνει μια συναλλαγή με κάποιον άλλον χρήστη και όταν εκείνος νομίζει ότι έχει πληρωθεί να δημιουργήσει τη διακλάδωση στην αλυσίδα ώστε αλλάζοντας αυτή τη συναλλαγή, να είναι σα να μην έγινε ποτέ και έτσι να έχει τα χρήματά του πίσω. Οπότε, αυτό που πρέπει να αναλογιστεί κανείς είναι το πόσο χρόνο χρειάζεται μια νέα συναλλαγή να επικυρωθεί ώστε να θεωρείται πλέον βέβαιη και έτσι να μη μπορεί ο επιτιθέμενος να την αλλάξει. Αν ο επιτιθέμενος τα καταφέρει, το δίκτυο ενημερώνει τον άλλο χρήστη αλλά πλέον είναι αργά για να αντιδράσει.

Κατά τη διαδικασία αποστολής μιας συναλλαγής, ο δέκτης θα στείλει στον αποστολέα το δημόσιο κλειδί του λίγο πριν αυτός υπογράψει ψηφιακά τη συναλλαγή έτσι ώστε ο αποστολέας να μην έχει χρόνο να ετοιμάσει από πριν καινούριο τμήμα και ξεπεράσει την ήδη υπάρχουσα αλυσίδα. Έτσι ο επιτιθέμενος θα ξεκινήσει αυτή τη διαδικασία, δηλαδή τη δημιουργία ενός τμήματος με μια δικιά του εκδοχή για τη συναλλαγή που μόλις πραγματοποίησε, από τη στιγμή που θα γίνει η συναλλαγή.

Ο δέκτης περιμένει μέχρι η συναλλαγή να προστεθεί στο τμήμα και z τμήματα να προστεθούν μετά από αυτό. Δεν μπορεί να γνωρίζει τι πρόοδο έχει κάνει μέχρι τότε ο επιτιθέμενος, αλλά υποθέτει πως η σωστή αλυσίδα αναπτύσσεται με τον αναμενόμενο χρόνο. Η προοπτική ανάπτυξης της αλυσίδας του επιτιθέμενου θα είναι η κατανομή Poisson (η οποία φαίνεται στον Πίνακα 1.10) με παράμετρο:

$$\lambda = z\frac{q}{p}.$$

Για να βρούμε την πιθανότητα ο επιτιθέμενος να προλάβει την ανάπτυξη της κανονικής αλυσίδας, πολλαπλασιάζουμε την συνάρτηση πυκνότητας πιθανότητας Poisson με κάθε πιθανή ποσότητα προόδου που μπορεί να έκανε ο επιτιθέμενος

με την προϋπόθεση να υπάρχει πιθανότητα από εκείνο το σημείο που βρίσκεται να φτάσει την ανάπτυξη της σωστής αλυσίδας και έχουμε:

$$\mathbb{P}(X < \infty) = \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \cdot \left\{ \begin{array}{cc} (q/p)^{z-k}, & k \leq z \\ 1, & k > z \end{array} \right\} = 1 - \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - (q/p)^{z-k}\right).$$

Αξίζει να σημειωθεί ότι κάθε τμήμα πρέπει να παράγεται ανά 10 με 12 λεπτά. Το σύστημα ελέγχει το χρόνο με τον οποίο παράγονται αυτά τα τμήματα. Αν ο χρόνος είναι πολύ μικρότερος, αυτό σημαίνει πως στο σύστημα έχει προστεθεί επεξεργαστική ισχύ και έτσι αυξάνεται η δυσκολία του προβλήματος. Αν παρατηρηθεί πως ο χρόνος είναι πολύ μεγαλύτερος, αυτό σημαίνει πως από το σύστημα έχουν αποχωρήσει miners, οπότε το πρόβλημα γίνεται ευκολότερο. Αν οι χρόνοι παραμένουν σταθεροί, τότε και το πρόβλημα παραμένει σταθερό.

Με τη παραγωγή του καινούριου τμήματος, έχουμε πάλι το σώμα του τμήματος που αποτελείται από τις νέες συναλλαγές που έχουν γίνει, οι οποίες είναι σε κατακερματισμένη μορφή hash καθώς και η επικεφαλίδα η οποία είναι και αυτή σε μορφή hash. Η επικεφαλίδα θεωρείται η πρώτη συναλλαγή του τμήματος και είναι η αμοιβή των miners για τη συνεισφορά τους στο σύστημα. Το ποσό των bitcoin που παίρνουν είναι σταθερά υποδιπλασιαζόμενο ανά κάποια χρονικά διαστήματα και είναι τα καινούρια κρυπτονομίσματα που παράχθηκαν και θα συνεχίζουν να παράγονται μέχρι να υπάρχουν στο σύνολο 21 εκατομμύρια στο δίκτυο. Επίσης οι miners μετά από κάθε συναλλαγή παίρνουν κάποιο ποσοστό ως προμήθεια.

Όλα τα πάραπάνω εργαλεία είναι χρήσιμα στο σύστημα τόσο για την ασφάλεια του όσο και για την ανωνυμία που παρέχεται μέσα στο δίκτυο. Είναι προφανές ότι τα κρυπτογραφικά εργαλεία προσφέρουν την ασφάλεια. Πως όμως τα παραπάνω προσδίδουν, μες το bitcoin δίκτυο, ανωνυμία στους χρήστες του; Αυτό είναι κάτι που για να απαντηθεί θα πρέπει να μελετηθεί το ίδιο το bitcoin δίκτυο.

1.4.2 Δίκτυα Συναλλαγών

Στο bitcoin δίκτυο το κυρίαρχο στοιχείο είναι οι συναλλαγές. Όπως είδαμε και παραπάνω, οι συναλλαγές αυτές γράφονται σε ένα τμήμα, με μια συγκεκριμένη μορφή, και έπειτα έχουμε μια ολόκληρη αλυσίδα από τέτοια τμήματα, η οποία ονομάζεται blockchain. Από αυτή την αλυσίδα, μπορούμε εύκολα να δημιουργήσουμε ένα δίκτυο συναλλαγών. Πριν όμως δώσουμε τον ορισμό του δικτύου συναλλαγών, ας δούμε το πρωτόκολλο δικτύο όταν γίνεται μια συναλλαγή. Το πρωτόκολλο είναι το εξής [2]:

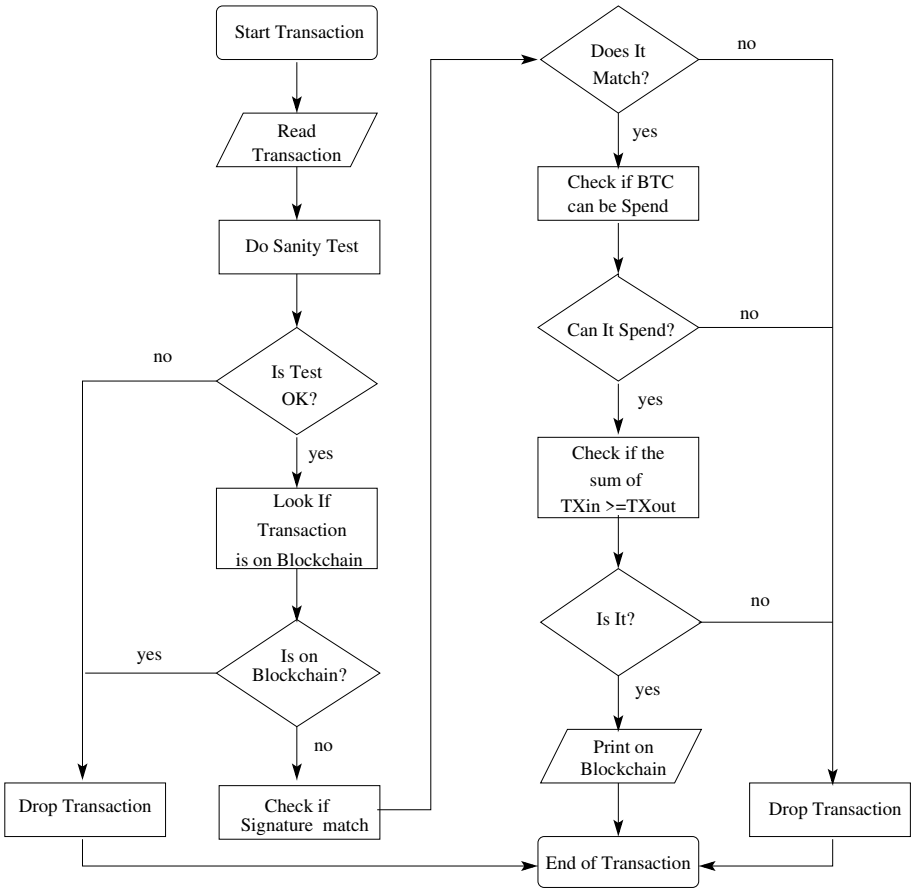
1. Κάθε νέα συναλλαγή διαδίδεται σε όλους του υπολογιστικούς κόμβους (miners).
2. Κάθε κόμβος, συλλέγει κάθε νέα συναλλαγή σε ένα τμήμα.
3. Κάθε κόμβος προσπαθεί να κάνει το Proof-of-Work για το τμήμα του.

4. Όταν ένας κόμβος ολοκληρώσει το Proof-of-Work, το μεταδίδει το τμήμα του στους υπόλοιπους κόμβους.

5. Οι κόμβοι δέχονται το τμήμα μόνο αν όλες οι συναλλαγές είναι έγκυρες και δεν είναι ήδη γραμμένες σε κάποιο άλλο τμήμα, δηλαδή δεν έχουν ξοδευτεί τα χρήματα της συναλλαγής.

6. Οι κόμβοι εκφράζουν την αποδοχή τους σε ένα τμήμα δουλεύοντας για τη δημιουργία του επόμενου τμήματος στην αλυσίδα χρησιμοποιώντας τη κατακερματισμένη τιμή του αποδεκτού τμήματος ως το προηγούμενο τμήμα.

Όπως είπαμε και παραπάνω, οι κόμβοι θεωρούν τη μεγαλύτερη αλυσίδα ως έγκυρη. Σε περίπτωση που δύο ή περισσότερες αλυσίδες έχουν το ίδιο μήκος, τις κρατάνε όλες μέχρι το επόμενο Proof-of-Work, από το οποίο θα ξεχωρίσει ποια αλυσίδα είναι έγκυρη, καθώς θα μεγαλώσει γρηγορότερα από τις άλλες. Έτσι όλοι θα συνεχίσουν να δουλεύουν πάνω στη μεγαλύτερη. Μια τέτοια διαδικασία φαίνεται στο διάγραμμα ροής στο Σχήμα 1.8.



Σχήμα 1.8: Διάγραμμα Ροής Πρωτοκόλλου Συναλλαγής στο Δίκτυο του Bitcoin

Οι συναλλαγές διακρίνονται στις εισερχόμενες συναλλαγές (input transaction) τις οποίες θα συμβολίζουμε με TXin και στις εξερχόμενες συναλλαγές (output transaction) τις οποίες θα συμβολίζουμε με TXout. Η κάθε εισερχόμενη συναλλαγή είναι μια εν δυνάμει εξερχόμενη συναλλαγή. Ο διαχωρισμός τους είναι απλός.

Ορισμός 1.4.1 *Ως εισερχόμενη συναλλαγή ορίζουμε τη συναλλαγή που δέχεται ένας χρήστης του δικτύου, ενώ ως εξερχόμενη συναλλαγή ορίζουμε τη συναλλαγή που στέλνει κάποιος χρήστης του δικτύου.*

Σε κάθε συναλλαγή, αναγράφονται λεπτομέρειες σχετικά με αυτή. Προφανώς μια τέτοια λεπτομέρεια είναι το ποσό bitcoin που μεταβιβάζεται. Εκτός από αυτό όμως, μια συναλλαγή "κουβαλάει" μαζί της και την ημερομηνία όπως και την ώρα που έγινε η συναλλαγή. Φυσικά για την εγκυρότητα της συναλλαγής υπάρχει και η ψηφιακή υπογραφή του αποστολέα. Το θέμα των ψηφιακών υπογραφών θα το αναλύσουμε στη συνέχεια.

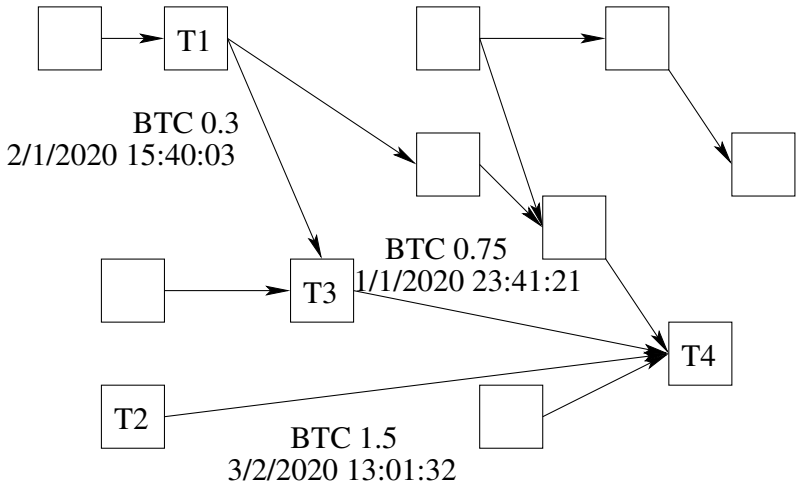
Ορισμός 1.4.2 *Ένα δίκτυο συναλλαγών $\mathcal{T}$ είναι ένας απλός κατευθυνόμενος βεβαρημένος γράφος (δεν έχει βρόγχους ούτε περισσότερες από μια ακμές ανάμεσα σε δύο κορυφές). Κάθε κορυφή αναπαριστά μια συναλλαγή ενώ κάθε κατευθυνόμενη ακμή από μια κορυφή σε κάποια άλλη αναπαριστά μια εξερχόμενη συναλλαγή της πρώτης κορυφής που αντιστοιχεί σε μια εισερχόμενη συναλλαγή της δεύτερης κορυφής. Το βάρος των ακμών είναι η τιμή των bitcion που μεταφέρονται καθώς και η ημερομηνία και η ώρα αποστολής (Time Stamp).*

Μια παρατήρηση είναι ότι σε ένα δίκτυο συναλλαγών κάθε συναλλαγή (κάθε κόμβος) δεν έχει μέγιστο όριο εισερχόμενων συναλλαγών (δηλαδή ο εισερχόμενος βαθμός δεν είναι φραγμένος), αλλά μπορεί να έχει μέχρι και δύο εξερχόμενες συναλλαγές (εξερχόμενος βαθμός κορυφής το πολύ 2). Η μία εξερχόμενη ακμή αφορά την πληρωμή κάποιου χρήστη και η δεύτερη τα ρέστα (αν υπάρχουν) που επιστρέφονται σε αυτόν που πληρώνει. Δηλαδή μια συναλλαγή (ένας κόμβος) δια των εξερχόμενων ακμών της δύναται να δημιουργήσει δύο άλλες συναλλαγές (κόμβους) μια για την πληρωμή ενός χρήστη και μια για τα ρέστα που πρέπει να επιστραφούν.

Για τον συμβολισμό δίνεται ο παρακάτω ορισμός

Ορισμός 1.4.3 *Ο βαθμός εισερχόμενης συναλλαγής για μια συναλλαγή t στο δίκτυο συναλλαγών, θα συμβολίζεται με $d^+_{txin}(t)$ και θα είναι το πλήθος των εισερχόμενων συναλλαγών για τη συναλλαγή t. Ο βαθμός εξερχόμενης συναλλαγής για μια συναλλαγή t στο δίκτυο συναλλαγών, θα συμβολίζεται με $d^+_{txout}(t)$ και θα είναι το πλήθος των εξερχόμενων συναλλαγών για τη συναλλαγή t, με $d^+_{txout}(t) \leq 2$.*

Αξίζει να σημειωθεί πως σε κάθε συναλλαγή, ένα ποσό πηγαίνει στους κόμβους (miners) ως προμήθεια (transaction fee). Η δεύτερη εξερχόμενη συναλλαγή, δηλαδή



Σχήμα 1.9: Δύκτιο Συναλλαγών

αυτή που επιστρέφει ρέστα, είναι απαραίτητη γιατί ο κάθε χρήστης μπορεί να ξοδέψει τα ποσά που έχει από εισερχόμενες συναλλαγές. Αυτά τα ποσά μπορεί να είναι μεγαλύτερα από αυτά που θέλει ο ίδιος να ξοδέψει και το σύστημα δε δίνει τη δυνατότητα σε κάποιον να διασπάσει ένα ποσό σε μικρότερο. Το μόνο που επιτρέπεται είναι να μετατρέψει ως μια έξοδο το άθροισμα πολλών εισερχόμενων συναλλαγών.

Όπως προκύπτει από τον ορισμό, ένα δίκτυο συναλλαγών $\mathcal{T}$ αναπαριστά τη ροή μεταφοράς bitcoin μεταξύ συναλλαγών σε βάθος χρόνου. Ένα παράδειγμα για την κατανόηση του δικτύου είναι το παρακάτω.

Παράδειγμα 1.4.1 *Όπως βλέπουμε και στο Σχήμα 1.9, μια συναλλαγή μπορεί να έχει περισσότερες από μία εισερχόμενες συναλλαγές. Στη περίπτωση του παραδείγματος τη T_4 έχει τέσσερις εισερχόμενες συναλλαγές. Επίσης, βλέπουμε ότι κάθε συναλλαγή έχει το πολύ δύο εξερχόμενες συναλλαγές. Στη περίπτωση μας, η T_1 έχει δύο εξερχόμενες συναλλαγές. Η μία πηγαίνει στη κορυφή με ετικέτα T_3 ενώ η άλλη είναι για τα ρέστα της πρώτης συναλλαγής.*

Όταν μια κορυφή δεν έχει εξερχόμενη ακμή, όπως είναι η T_4, σημαίνει ότι όλες αυτές οι εισερχόμενες συναλλαγές μπορούν να ξοδευτούν. Τέλος, βλέπουμε πως τα βάρη των ακμών είναι η το πλήθος των bitcoin που μεταφέρονται καθώς και η ώρα και η ημερομηνία αυτής της συναλλαγής. Στην περίπτωσή μας, η συναλλαγή από το T_1 στο T_3 έγινε στις $2/1/2020$ και ώρα $15:40:03$ και μεταφέρθηκαν 0.3 bitcoin.

1.4.3 Δίκτυα Χρηστών

Σε αυτό το εδάφιο θα εξεταστεί το δίκτυο χρηστών του bitcoin δικτύου. Θα μπορούσαμε να ισχυριστούμε πως ίσως είναι το ποιο σημαντικό δίκτυο από τα δύο, κα-

θώς μέσω του οποίου μπορούμε να βγάλουμε πολλά συμπεράσματα για τη συμπεριφορά ενός χρήστη. Αρχικά όμως, θα εξετάσουμε κάποιες ακόμα υπηρεσίες που είναι απαραίτητες στο δίκτυο αυτό.

Ψηφιακό Πορτοφόλι

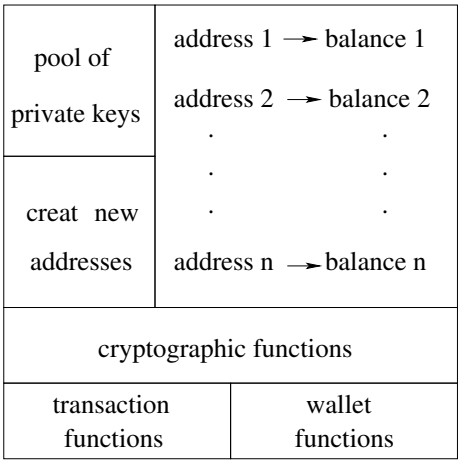
Στο bitcoin δίκτυο κάθε χρήστης γίνεται γνωστός με μια διεύθυνση. Με αυτόν τον τρόπο διασφαλίζεται και η ανωνυμία στο δίκτυο, δηλαδή κατά την συναλλαγή αυτό που φαίνεται στη blockchain είναι μια διεύθυνση να στέλνει ένα ποσό μια συγκεκριμένη χρονική στιγμή σε κάποια άλλη διεύθυνση. Τα φυσικά πρόσωπα πίσω από τις διευθύνσεις δεν μπορούν να γίνουν γνωστά, όπως επίσης ούτε και τα αγαθά που αγοράστηκαν μέσω αυτής της συναλλαγής.

Ας δούμε όμως πως δημειουργείται μια τέτοια διεύθυνση και πως είναι δυνατών να γίνονται συναλλαγές μέσω αυτών των διευθύνσεων. Το εργαλείο που χρησιμοποιείται σε αυτές τις περιπτώσεις είναι το πορτοφόλι (Σχήμα 1.10). Ένα ψηφιακό πορτοφόλι έχει αρκετές δυνατότητες. Αρχικά, ένα πορτοφόλι έχει συνήθως τη δυνατότητα να δημιουργεί διευθύνσεις. Μια διεύθυνση μπορεί κάποιος να τη φανταστεί σαν έναν τραπεζικό λογαριασμό. Οπότε βγαίνει και εύκολα το συμπέρασμα ότι το πορτοφόλι γνωρίζει το σύνολο των διαθέσιμων bitcoin που βρίσκονται σε κάθε διεύθυνση. Έπειτα, σε κάθε συναλλαγή, αποδέχεται εισερχόμενες συναλλαγές όπως επίσης και υπογράφει εξερχόμενες συναλλαγές.

Υπάρχουν δύο μεγάλες κατηγορίες πορτοφολιών: αυτά που έχουν πρόσβαση στο διαδίκτυο (hot wallets) και αυτά που δεν έχουν (cold wallets). Καταλαβαίνει κανείς ότι ανάλογα με τη μορφή που έχει ένα πορτοφόλι αυξάνονται ή μειώνονται οι λειτουργίες του. Για παράδειγμα, ένα πορτοφόλι που δεν είναι συνδεδεμένο στο διαδίκτυο δεν μπορεί να δεχτεί συναλλαγές.

Τα πορτοφόλια που έχουν σύνδεση στο διαδίκτυο είναι τις περισσότερες φορές λογισμικά τα οποία βρίσκονται είτε στο διαδίκτυο είτε στην εκάστοτε συσκευή που χρησιμοποιεί ο καθένας (ηλεκτρονικός υπολογιστής, κινητό τηλέφωνο και άλλα). Τα πορτοφόλια που δεν έχουν σύνδεση στο διαδίκτυο μπορεί να είναι επίσης λογισμικά σε συσκευές, αλλά μπορεί να έχουν και φυσική μορφή. Στην πιο απλή τους μορφή είναι απλά ένα κομμάτι χαρτί.

Κάθε είδος έχει τα δικά του πλεονεκτήματα και μειονεκτήματα. Για παράδειγμα, με τα πορτοφόλια που έχουν σύνδεση στο διαδίκτυο, η διαδικασία της συναλλαγής είναι πολύ πιο γρήγορη και άμεση καθώς και υπάρχει μικρή πιθανότητα σφάλματος. Ένα πιθανό σφάλμα είναι να δοθεί λάθος η διεύθυνση αποστολής. Αν σταλούν χρήματα σε λάθος διεύθυνση, δεν υπάρχει τρόπος να γυρίσουν πίσω στον κάτοχό τους και έτσι αυτό το ποσό πλέον θεωρείται χαμένο και δε χρησιμοποιείται ποτέ ξανά στο δίκτυο. Το μειονέκτημα αυτών των πορτοφολιών είναι ότι υπάρχει μεγαλύτερη πιθανότητα υποκλοπής των διευθύνσεων που διαχειρίζεται μέσω διαδικτυακών επι-



Σχήμα 1.10: Ψηφιακό Πορτοφόλι

θέσεων.

Από την άλλη πλευρά, τα πορτοφόλια χωρίς πρόσβαση στο διαδίκτυο δεν προσφέρουν μεγάλη αμεσότητα σε μια συναλλαγή, αλλά είναι πιο ασφαλή σε καταστάσεις διαδικτυακής απειλής. Αν πάρουμε σαν παράδειγμα την πιο απλή μορφή, το χαρτί, τότε τα μειονεκτήματα είναι εμφανή καθώς κάποιος θα πρέπει να διαχειρίζεται όλες τις συναλλαγές του διά χειρός, αν και θα μπορεί να είναι βέβαιος για την ασφάλεια των χρημάτων του, εκτός βέβαια από την περίπτωση της φυσικής κλοπής στο χώρο που έχει φυλαγμένο το πορτοφόλι του.

Το σημαντικότερο κομμάτι μιας συναλλαγής είναι η διεύθυνση καθώς αυτή είναι που δίνει ανωνυμία στο συστήματος και είναι το μέσο με το οποίο γίνονται οι συναλλαγές καθώς χρησιμοποιείται για την υπογραφή των συναλλαγών. Πως δημιουργείται όμως μια διεύθυνση;

Σε ένα πορτοφόλι πρέπει να υπάρχουν κρυπτογραφικές συναρτήσεις μέ τη βοήθεια των οποίων κατασκευάζεται μια διεύθυνση. Το πορτοφόλι χρησιμοποιεί ένα ασύμμετρο κρυπτοσύστημα για την κατασκευή ιδιωτικού και δημόσιου κλειδιού. Το ιδιωτικό κλειδί πρέπει να παραμείνει μυστικό. Από τη στιγμή του δημιουργηθεί αυτό το ζεύγος, ακολουθείται ο παρακάτω αλγόριθμος για τη κατασκευή της διεύθυνσης με τη βοήθεια των συναρτήσεων κατακερματισμού SHA-256 και RIPEMD-160:

- $index =< OneByteNumber >$
- $keyHash = RIPEMD - 160(SHA - 256(publicKey))$
- $dataHash = SHA - 256(SHA - 256(index||Hash))$
- $checkSum = GetFirst4Bytes(dataHash)$
- $address = Base58Encode(index||keyHash||checkSum)$

Η μεταβλητή *index* δέχεται τιμές μήκους 1-Byte η οποία είναι η έκδοση του δικτύου. Στη συνέχεια υπολογίζεται η κατακερματισμένη τιμή του δημόσιου κλειδιού δίνοντας το ως είσοδο στη SHA-256 και το αποτέλεσμά της στην RIPEMD-160. Με τον συμβολισμό || εννοούμε τη συγχώνευση δύο αριθμών σε έναν. Το αποτέλεσμα της προηγούμενης διαδικασίας τοποθετείται αμέσως μετά το *index* και το αποτελεσμα της συγχώνευσης αυτής δίνεται ως είσοδος στη $(SHA-256)^2$. Από την ποσότητα που υπολογίστηκε παίρνουμε τα τέσσερα πιο σημαντικά *Byte* και τα τοποθετούμε αμέσως μετά το *index* και τη τιμή κατακερματισμού του δημόσιου κλειδιού. Τέλος, η ποσότητα που προέκυψε από την παραπάνω συγχώνευση κωδικοποιείται στο πενηνταοχταδικό σύστημα το οποίο αποτελείται από όλους τους αριθμούς από το 1 μέχρι το 9, τα κεφαλαία A μέχρι Z και τα πεζά a μέχρι z. Οι μόνοι χαρακτήρες που λείπουν είναι τα O (κεφαλαίο o), το I (κεφαλαίο i) και το l (πεζό L). Τα παραπάνω γράμματα λείπουν για να μην υπάρχει παρανόηση.

Από αυτή τη διαδικασία μπορεί να προκύψει διεύθυνση από 27 έως και 34 χαρακτήρες. Να σημειωθεί πως οι διευθύνσεις που χρησιμοποιούνται στο δίκτυο του bitcoin ξεκινάνε με το 1.

Η διεύθυνση έχει μείζουσα σημασία στο σύστημα καθώς χρησιμοποιείται στην κατοχύρωση της ιδιοκτησίας bitcoin. Τα bitcoin αλλάζουν ιδιοκτήτη όταν η κατακερματισμένη μορφή του συμπιεστεί με τη διεύθυνση του δέκτη και περάσει από μια συνάρτηση κατακερματισμού. Έτσι, οποιαδήποτε συναλλαγή υπογραφεί από τον κάτοχο των συγκεκριμένων bitcoin θα μπορεί να γίνει καθώς η παραπάνω διαδικασία επιβεβαιώνει πως είναι δικά του.

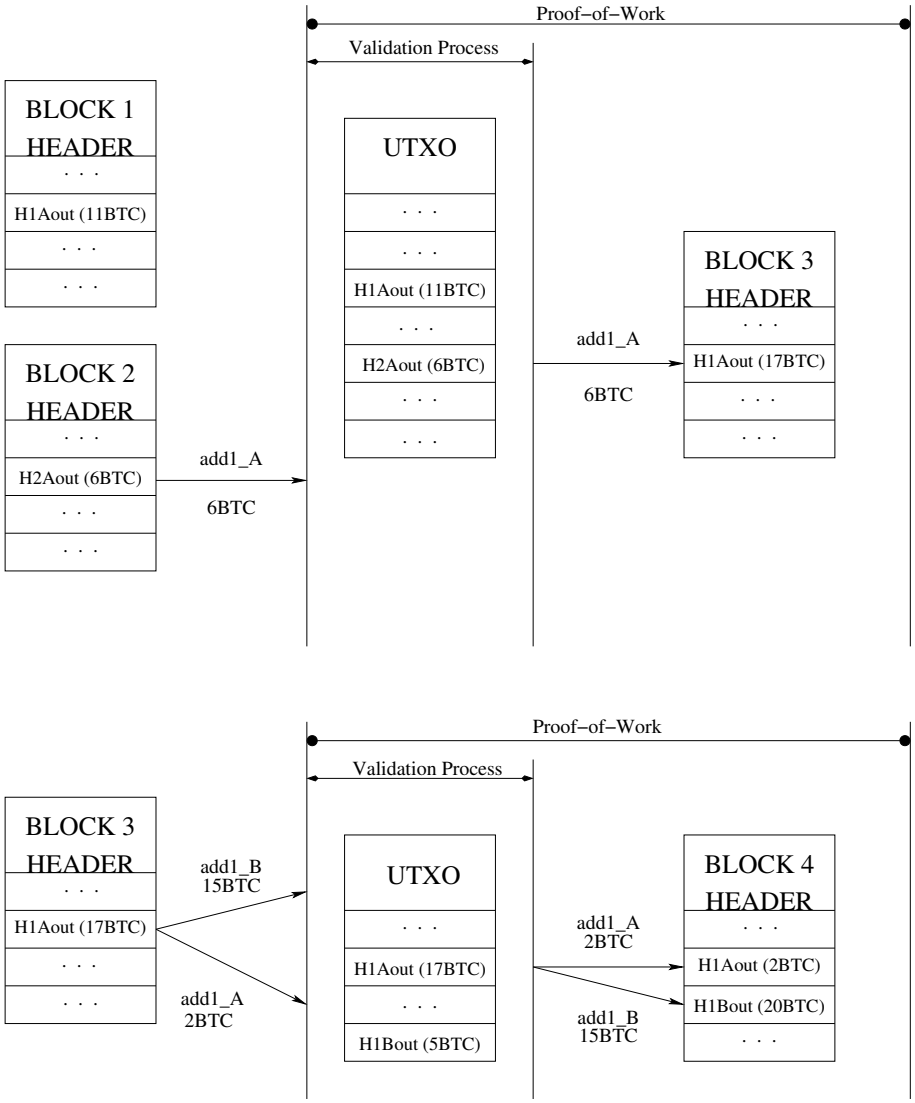
Το παραπάνω μας δίνει τον πραγματικό ορισμό ενός τέτοιου κρυπτονομίσματος, δηλαδή ότι είναι μια αλυσίδα από ψηφιακές υπογραφές που αν τη ξετυλίξει κανείς μπορεί να φτάσει στον πρώτο κάτοχο του.

Διαδικασία Επιβεβαίωσης Διαθεσιμότητας Ποσού bitcoin

Κάτι που αναφέρθηκε για το πορτοφόλι είναι ότι γνωρίζει το ποσό διαθέσιμων bitcoin σε κάθε διεύθυνση. Για να μπορέσει να γίνει κάτι τέτοιο θα πρέπει να έχει πρόσβαση σε κάποια λίστα που θα καταγράφονται όλα τα bitcoin που μπορούν να ξοδευτούν.

Αν αναλογιστεί κάποιος το πότε κατασκευάστηκε το δίκτυο του bitcoin θα καταλάβει πως έχουν γίνει από τότε εκατομμύρια συναλλαγών. Αυτό έχει ως αποτέλεσμα η blockchain να είναι αρκετά μεγάλη και δύσκολα διαχειρίσιμη. Για τον λόγο αυτό το σύστημα έχει ένα τρόπο ώστε η διαδικασία επιβεβαίωσης του αν ένα ποσό μπορεί να ξοδευτεί να είναι πολύ πιο γρήγορη.

Αρχικά διαβάστηκε όλη η blockchain και βρέθηκαν όλα τα διαθέσιμα - ″ξοδέψιμα″ bitcoin. Έπειτα, τα αποτελέσματα καταγράφηκαν σε μία μνήμη, πολύ μικρότερη της blockchain, η οποία ονομάζεται UTXO [12]. Κάθε φορά, όταν είναι να δημιουργηθεί ένα νέο τμήμα συναλλαγών, ελέγχεται αν τα bitcoin που μεταφέρονται υπάρχουν στη



Σχήμα 1.11: Συναλλαγές με UTXO

λίστα. Αν υπάρχουν, σβήνονται και καταχωρείται το ποσό με τον καινούριο ιδιοκτήτη (ψηφιακή υπογραφή) ώστε να έχει πλέον ο νέος του ιδιοκτήτης το δικαίωμα να το ξοδέψει. Αν δεν υπάρχει στη λίστα, τότε το δίκτυο αγνοεί την συναλλαγή και αυτή δεν ολοκληρώνεται.

Στο Σχήμα 1.11 φαίνεται η διαδικασία στην οποία ένας χρήστης του δικτύου, ο A, στέλνει μια ποσότητα κρυπτονομισμάτων στον χρήστη B. Για τον συμβολισμό έχουμε για λόγους συντομίας, με $H1Aout$ συμβολίζουμε την κατακερματισμένη τιμή των κρυπτονομισμάτων του χρήστη Α που βρίσκονται στη διεύθυνση $add1_A$. Όμοια, με $H2Aout$ συμβολίζουμε την κατακερματισμένη τιμή των κρυπτονομισμάτων του χρήστη Α που βρίσκονται στη διεύθυνση $add2_A$. Αυτές οι τιμές είναι καταγεγραμμένες σε μπλοκ της *BlockChain* (δεν είναι απαραίτητο και οι δύο να βρίσκονται στο

ίδιο τμήμα). Μέσα σε παρένθεση αναγράφεται η ποσότητα των *bitcoin* στην οποία αναφέρεται η εκάστοτε κατακερματισμένη τιμή. Πάνω στα βέλη αναγράφεται το ποσό μεταφοράς καθώς και ο προορισμός της συναλλαγής. Έτσι, αν υποθέσουμε ότι ο χρήστης Α θέλει να στείλει μια ποσότητα $15BTC$ στον χρήστη Β, η διαδικασία θα χωριστεί σε δύο βήματα. Στο πρώτο βήμα (πάνω μέρος του σχήματος) ο Α από τη στιγμή που δεν έχει μια διεύθυνση η οποία περιέχει ένα ποσό μεγαλύτερο ή ίσο των $15BTC$ μπορεί είτε να στείλει δύο συναλλαγές από δύο διαφορετικές διευθύνσεις για να συμπληρωθεί το ποσό ή στην περίπτωσή μας, θα στείλει από τη μία του διεύθυνση ($H2Aout$) στην άλλη ($H1Aout$) έτσι ώστε να έχει μια διεύθυνση που θα περιέχει ένα ποσό μεγαλύτερο ή ίσο των $15BTC$. Όπως φαίνεται στο σχήμα, στέλνει $6BTC$ από τη διεύθυνση $add2_A$ στην διεύθυνση $add1_A$. Κατά τη διαδικασία του Proof-of-Work, γίνεται ο έλεγχος στην $UTXO$, δηλαδή ελέγχεται αν είναι αποθηκευμένη η τιμή $H2Aout$ σε αυτή τη μνήμη. Πράγματι, είναι καταχωρημένη στη μνήμη και αυτό σημαίνει ότι αυτή η ποσότητα των *bitcoin* μπορεί να ξοδευτεί. Έτσι, συνεχίζεται η διαδικασία του Proof-of-Work για τη δημιουργία του νέου τμήματος κατά την οποία θα γραφτεί η συναλλαγή αυτή. Την ίδια στιγμή θα σβηστεί από την μνήμη της $UTXO$ η $H2Aout$ καθώς και η τιμή της $H1Aout$ θα αλλάξει. Η αλλαγή αυτή φαίνεται στο κάτω μέρος του σχήματος όπου είναι και το δεύτερο βήμα της συναλλαγής. Αφού, λοιπόν, έχει μαζέψει το ποσό που χρειάζεται ο Α σε μία διεύθυνση, θα στείλει το ποσό των $15BTC$ στη διεύθυνση του Β ($add1_B$). Καθώς όμως, το ποσό της διεύθυνσης από το οποίο θα στείλει αυτή τη συναλλαγή είναι μεγαλύτερο των $15BTC$ θα πρέπει να κάνει δύο συναλλαγές, μία με αποδέκτη την διεύθυνση $add1_B$ με το ποσό των $15BTC$ και μία με αποδέκτη τη δική του διεύθυνση $add1_A$ κατά την οποία θα του επιστραφούν τα ρέστα $2BTC$ όπως αναγράφεται πάνω στα βέλη. Και πάλι, για το Proof-of-Work, θα γίνει ο έλεγχος πάνω στην $UTXO$, αν υπάρχουν δηλαδή αυτές οι ποσότητες καταχωρημένες στη μνήμη. Από τη στιγμή που υπάρχουν, συνεχίζεται η διαδικασία με τη δημιουργία ενός νέου μπλοκ όπου καταγράφονται και οι συναλλαγές αυτές. Τέλος, σβήνονται οι προηγούμενες ποσότητες από την $UTXO$ και καταγράφονται οι νέες.

Γνωρίζοντας πλέον τι είναι η διεύθυνση ενός χρήστη, τι είναι το πορτοφόλι και με ποιον τρόπο αλλάζουν τα χρήματα ιδιοκτήτη μπορούμε να δούμε πως λειτουργεί το δίκτυο των χρηστών. Ας δώσουμε, όμως πρώτα τον ορισμό αυτού του δικτύου.

Ορισμός 1.4.4 *Δίκτυο χρηστών $\mathcal{U}$ είναι ένας απλός κατευθυνόμενος βεβαρημένος γράφος (δεν έχει βρόγχους ούτε περισσότερες από μια ακμές ανάμεσα σε δύο κορυφές). Κάθε κορυφή αναπαριστά έναν χρήστη του δικτύου ενώ κάθε κατευθυνόμενη ακμή από μια κορυφή σε κάποια άλλη αναπαριστά ένα ζεύγος εισερχόμενης - εξερχόμενης συναλλαγής όπου το δημόσιο κλειδί της εισερχόμενης συναλλαγής ανήκει στο χρήστη της κορυφής από την οποία ξεκινάει η κατευθυνόμενη ακμή, ενώ το δημόσιο κλειδί της εξερχόμενης συναλλαγής στο χρήστη της κορυφής που καταλήγει η κατευθυνόμενη ακμή. Το βάρος των ακμών είναι τα bitcoin που μεταφέρονται καθώς και η ημερομηνία και η ώρα αποστολής (Time Stamp).*

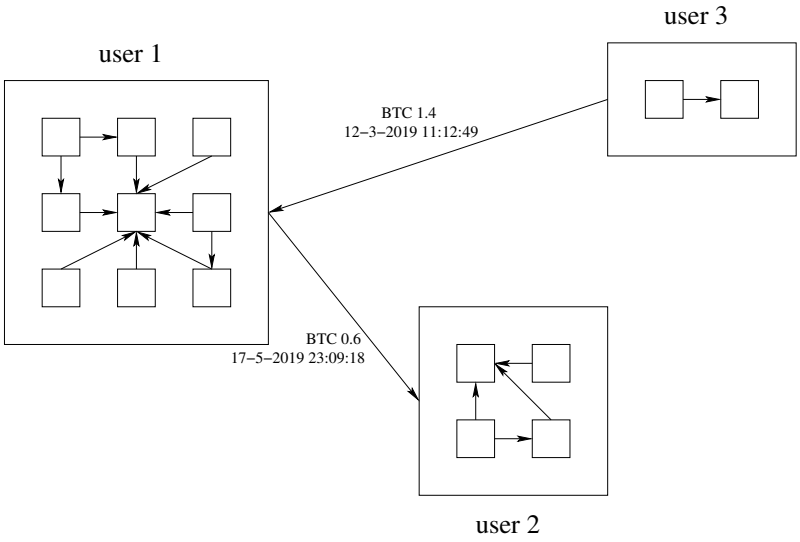
Όπως μπορεί να καταλάβει κάποιος, το δίκτυο των χρηστών αναπαριστά τη ροή συναλλαγών ανάμεσα σε χρήστες του δικτύου σε βάθος χρόνου. Για να βρει κάποιος ένα δίκτυο χρηστών μέσω της blockchain δεν είναι εύκολο. Αυτό συμβαίνει γιατί η μόνη πληροφορία σχετικά με τις συναλλαγές που παίρνουμε είναι διευθύνσεις και ποσά. Κάθε χρήστης, όπως είδαμε μπορεί να κατασκευάσει και να διαχειρίζεται όσες διευθύνσεις θέλει. Για να φτιαχτεί ένα τέτοιο δίκτυο θα χρειαστεί να αντιστοιχηθούν οι χρήστες του δικτύου με όλες τους τις διευθύνσεις. Μπορούμε να καταλήξουμε, όμως σε συμπεράσματα χωρίς να γνωρίζουμε στην πραγματικότητα τον γράφο. Για παράδειγμα μπορούμε να φανταστούμε ότι ο γράφος θα έχει υπογράφους κλίκες καθώς αν ο χρήστης διαχειρίζεται πολλές διευθύνσεις, θα μπορεί να κάνει αγοραπωλησίες από αυτές αλλά συνήθως τα χρήματα αυτά συλλέγονται σε μια διεύθυνση. Άρα σε εκείνη την κορυφή θα υπάρχουν πολλές εισερχόμενες συναλλαγές από τον ίδιο χρήστη. Με αυτή τη λογική, θα μπορούσε κανείς να πει ότι στον γράφο ορίζουμε ως κορυφές αυτές τις κλίκες υπογράφους. Ας δούμε όμως ένα παράδειγμα για την καλύτερη κατανόηση του προβλήματος.

Παράδειγμα 1.4.2 *Όπως παρατηρούμε στο Σχήμα 1.12 έχουμε αναγνωρίσει τρεις χρήστες του δικτύου οι οποίοι είναι κάτοχοι περισσότερων από μια διεύθυνση. Θέτουμε ως κορυφή του γράφου να είναι ολόκληρη η κλίκα συναλλαγών μεταξύ των διευθύνσεων που διαχειρίζεται ο ίδιος και ως συναλλαγή μεταξύ των χρηστών δεχόμαστε αυτή μεταξύ των μεγάλων κορυφών. Να σημειωθεί πως οποιαδήποτε εξερχόμενη ή εισερχόμενη συναλλαγή προς τη κορυφή που περιέχει μια ολόκληρη κλίκα μπορεί να είναι μια συναλλαγή από ή προς οποιαδήποτε από τις παρεχόμενες κορυφές τις κλίκας, δηλαδή τις διευθύνσεις που διαχειρίζεται ο χρήστης. Στο παράδειγμα ο χρήστης user 1 είναι κάτοχος εννιά διαφορετικών διευθύνσεων, ενώ ο χρήστης user 2 είναι κάτοχος τεσσάρων και ο χρήστης user 3 δύο. Παρατηρούμε, επίσης ότι μεταξύ των διευθύνσεων ενός χρήστη μπορούν να γίνονται πολλές συναλλαγές, αλλά θεωρούμε ως συναλλαγή στο δίκτυο αυτή μεταξύ διαφορετικών χρηστών όπως αυτή μεταξύ του user 3 και κάποιου άλλου χρήστη. Και πάλι βλέπουμε ότι στην ακμή αυτού του γράφου υπάρχει ένα Time Stamp της συναλλαγής, δηλαδή πότε έγινε αυτή η συναλλαγή, καθώς και το ποσό της συναλλαγής.*

Ένα τέτοιο δίκτυο δεν είναι εύκολο να δημιουργηθεί. Αυτό που προσφέρει το δίκτυο στους χρήστες του σχετικά με την ανωνυμία είναι η δυνατότητα που έχει ο καθένας να φτιάξει όσες διευθύνσεις θέλει. Αυτό έχει ως συνέπεια τη δημιουργία των λεγόμενων διευθύνσεων μιας χρήσης [4].

Προτού δούμε όμως τον ορισμό μιας τέτοιας διεύθυνσης, θα παρουσιαστεί ο συμβολισμός που θα χρησιμοποιηθεί παρακάτω. Για τα δίκτυα των χρηστών έχουμε τους παρακάτω ορισμούς.

Ορισμός 1.4.5 *Με $d^{+}_{addr}(pk)$ για ένα δημόσιο κλειδί pk συμβολίζουμε το πλήθος των εμφανίσεων του pk ως έξοδο μιας συναλλαγής. Με $d^{-}_{addr}(pk)$ για ένα δημόσιο κλειδί pk συμβολίζουμε το πλήθος των εμφανίσεων του pk ως είσοδο μιας συναλλαγής.*



Σχήμα 1.12: Δίκτυο Χρηστών

Ορισμός 1.4.6 *Έστω μια ακμή t του δικτύου χρηστών. Με $outputs(t)$ θα συμβολίζουμε το σύνολο των διευθύνσεων που τα bitcoin της συναλλαγής καταλήγουν, ενώ με $inputs(t)$ θα συμβολίζουμε το σύνολο των διευθύνσεων από όπου τα bitcoin της συναλλαγής ξεκινούν.*

Τώρα είμαστε έτοιμη να δώσουμε τον ορισμό των διευθύνσεων μιας χρήσης.

Ορισμός 1.4.7 *Ένα δημόσιο κλειδί pk ονομάζεται διεύθυνση μιας χρήσης αν για μια συναλλαγή t ισχύουν τα παρακάτω:*

1. *Το $d^+_{addr}(pk) = 1$, δηλαδή είναι η πρώτη εμφάνιση του κλειδιού pk.*
2. *Τα bitcoin της συναλλαγής δεν είναι τα πρώτα που έχει παράξει το δίκτυο.*
3. *Δεν υπάρχει $pk' \in outputs(t)$ τέτοιο ώστε $pk' \in inputs(t)$, δηλαδή δεν είναι μια διεύθυνση που αλλάζει μόνη της.*
4. *Δεν υπάρχει $pk' \in outputs(t)$ τέτοιο ώστε $pk' \neq pk$, αλλά το $d^+_{addr}(pk') = 1$, δηλαδή από όλες τις διευθύνσεις αποδέκτες το (1) ισχύει μόνο για το pk.*

Είναι φανερό πως χρησιμοποιώντας μια τέτοια διεύθυνση για συναλλαγή είναι σχεδόν απίθανο να γίνει σύνδεση μεταξύ της διεύθυνσης και του χρήστη πίσω από αυτήν καθώς δεν υπάρχει καμία πληροφορία που θα μπορούσε κάποιος να βασιστεί ώστε να κάνει τη σύνδεση. Στο bitcoin δίκτυο η χρήση τέτοιων διευθύνσεων είναι πολύ συχνή. Σχεδόν το ένα τέταρτο των συναλλαγών κάθε μήνα είναι με αυτόν τον τρόπο. Γενικά, για λόγους ασφαλείας και ανωνυμίας, προωθείται αυτός ο τρόπος συναλλαγών.

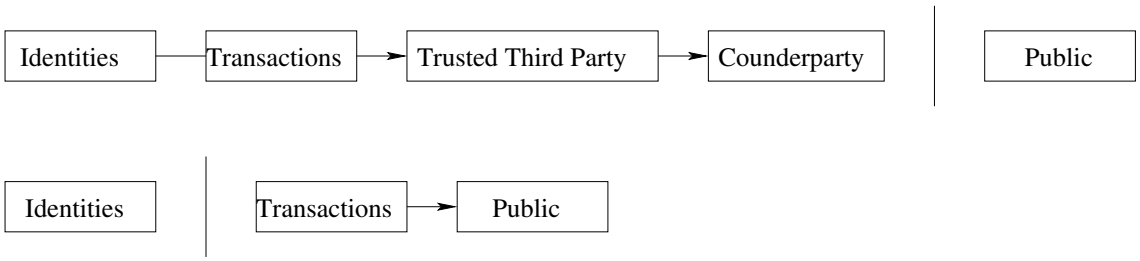
Είδαμε τεχνικές με τις οποίες το bitcoin δίκτυο του προσφέρει ανωνυμία στους χρήστες του. Είναι όμως κάτι τέτοιο αρκετό για να διατηρηθεί κρυφή η ταυτότητα ενός χρήστη; Παρακάτω θα δούμε την Peer-2-Peer (P2P) σύνδεση στο δίκτυο που χρησιμοποιείται και μεθόδους σύνδεσης σε αυτό για την μεγαλύτερη ασφάλεια της ανωνυμίας ενός χρήστη.

Είναι εύκολο να φανταστεί κανείς ότι ένα τέτοιο δίκτυο με τόσο δυνατό σύστημα ανωνυμίας θα προσέλκυε αρκετές παράνομες διαδικασίες. Πράγματι, το bitcoin απέκτησε μεγάλη αξία κυρίως για αυτόν τον λόγο. Μήπως όμως αυτό που όλοι πιστεύουν πως είναι τόσο ασφαλές στην πραγματικότητα δεν είναι; Αυτό το ερώτημα θα προσπαθήσουμε να απαντήσουμε στο επόμενο κεφάλαιο.

Κεφάλαιο 2

Μέθοδοι Άρσης Ανωνυμίας των Χρηστών

Το τραπεζικό σύστημα έχει καταφέρει να αποκτήσει ένα επίπεδο ανωνυμίας παρέχοντας πολύ λίγες πληροφορίες μεταξύ των αναφλεγόμενων οντοτήτων (Σχήμα 2.1 πρώτο μέρος). Η αναγκαιότητα για την ύπαρξη δημόσιας λίστας συναλλαγών έρχεται σε ρήξη με την παραπάνω μέθοδο ανωνυμίας. Υπάρχει όμως ένας ακόμα τρόπος για να επιτευχθεί η ανωνυμία, ο τρόπος αυτός είναι διατηρώντας τα δημόσια κλειδιά ως μυστικό. Ένα τέτοιο μοντέλο ανωνυμίας είναι όπως αυτό του χρηματιστηρίου, δηλαδή οι συναλλαγές είναι δημόσιες, όμως κανείς δεν μπορεί να δει ποιος έκανε αυτές τις συναλλαγές. Η μόνη πληροφορία που διαρρέει είναι ότι κάποιος στέλνει ένα ποσό σε κάποιον άλλον. Οπότε, σε ένα τέτοιο σύστημα, ένα επιπλέον τοίχος προστασίας είναι ένα νέο ζεύγος κλειδιών που χρησιμοποιείται για τις συναλλαγές (Σχήμα 2.1 κάτω μέρος) [2]. Τα κλειδιά αυτά θα πρέπει να παραμένουν κρυφά ειδάλλως θα υπάρχει σύνδεση της οντότητας της συναλλαγής και της φυσικής οντότητας.



Σχήμα 2.1: Μοντέλα Ανωνυμίας

Στο κεφάλαιο αυτό θα δείξουμε ότι η παραπάνω παραδοχή πως bitcoin δίκτυο παρέχει ένα ισχυρό μοντέλο ανωνυμίας δεν είναι σωστή. Επίσης, θα δούμε με ποια μέσα μπορούμε να βγάλουμε συμπεράσματα για τους χρήστες πίσω από τις διευ-

θύνσεις που χρησιμοποιούν, όσο και για την τοποθεσία που βρίσκονται. Τέλος, θα δούμε αναλύσεις ανωνυμίας που μας οδηγούν στην εύρεση χρηστών.

2.1 Το Πρόβλημα της Ανωνυμίας στο Bitcoin Δίκτυο

Ο Nakamoto αναφέρει [2] ότι ορισμένες διαρροές είναι αναπόφευχτες σε συναλλαγές που συμμετέχουν με πολλές ροές εισόδου. Και αυτό γιατί κάτι τέτοιο αποκαλύπτει πως όλες αυτές οι διευθύνσεις που αντιστοιχούν σε αυτές τις ροές ανήκουν στον ίδιο χρήστη. Αν το δημόσιο κλειδί ενός χρήστη αποκαλυφθεί τότε είναι εύκολο να βρεθούν και όλες οι συναλλαγές στις οποίες μπορεί να είχε εμπλακεί.

Σε ένα δίκτυο κρυπτονομισμάτων αυτό που επιδιώκεται είναι να πληρεί δύο βασικές ιδιότητες, αυτές της ανωνυμίας και της μικρής καθυστέρησης (low latency).

Ορισμός 2.1.1 *Με του όρο μικρή καθυστέρηση εννοούμαι ότι ο μέγιστος χρόνος που απαιτείται για ένα μήνυμα να φτάσει σε όλους τους υπολογιστικούς κόμβους είναι μικρός και φραγμένος.*

Η μικρή καθυστέρηση είναι απαραίτητη για ένα τέτοιο δίκτυο. Αν το δίκτυο δεν καταφέρει να μεταφέρει σε όλους τους κόμβους το μήνυμα σε έναν προβλεπόμενο χρόνο, θεωρείται ασυνεπές.

Ορισμός 2.1.2 *Με του όρο ανωνυμία εννοούμε ότι ο επιτιθέμενος δεν μπορεί να κάνει μια σύνδεση μεταξύ της διεύθυνσης μιας συναλλαγής και της IP διεύθυνσης της συσκευής που έκανε τη συναλλαγή η οποία συνδέεται με ένα φυσικό πρόσωπο.*

Υπενθυμίζουμε ότι κάθε συναλλαγή που γίνεται από κάποιο δημόσιο κλειδί βρίσκεται σε κάποιο τμήμα της blockchain. Έτσι, αν ένα δημόσιο κλειδί συνδεθεί με τη διεύθυνση της IP, τότε μπορεί να γίνει συνδεση μεταξύ όλων των συναλλαγών που έχουν γίνει από αυτό το κλειδί και σε ορισμένες περιπτώσεις αυτό μπορεί να οδηγήσει στη σύνδεση των παραπάνω με το φυσικό πρόσωπο. Για τον λόγο αυτό οι επιθέσεις ανωνυμοποίησης βασίζονται στην εύρεση του ιστορικού συναλλαγών ενός χρήστη. Ακόμα και αν ο χρήστης δημιουργεί πολλές διευθύνσεις για τις συναλλαγές του, αν γίνει μια σύνδεση μεταξύ αυτών και της διεύθυνσης της IP του, τότε μπορεί να βρεθούν όλες οι συναλλαγές του.

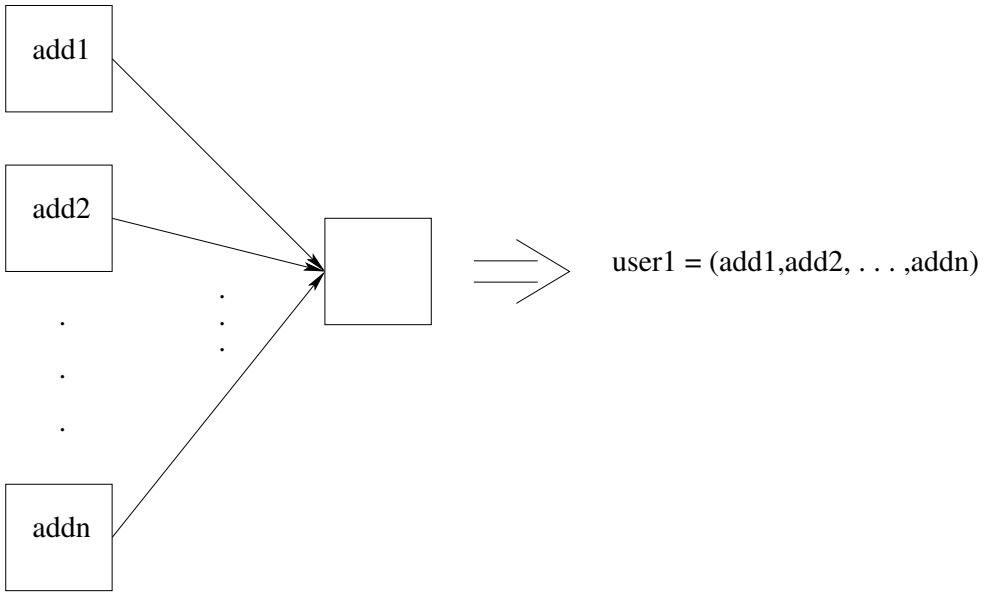
Με την πάροδο του χρόνου, έχουν παρουσιαστεί πολλές τεχνικές άρσης της ανωνυμίας σε ένα P2P δίκτυο του bitcoin. Συνήθως χρησιμοποιείται ένας υπερκόμβος που θα συνδεθεί στο bitcoin δίκτυο και θα καταγράφει κάθε συναλλαγή από τους κόμβους του δικτύου. Καθώς οι κόμβοι μεταδίδουν τις συναλλαγές συμμετρικά στο δίκτυο, είναι δυνατόν να γίνει σύνδεση μεταξύ των διευθύνσεων και των δημόσιων

κλειδιών του χρήστη με ακρίβεια μεγαλύτερη του 30%. Το ποσοστό αυτό βασίζεται σε μικρή χρήση της γνώσης του δικτύου και είναι πιθανό να μεγαλώσει με χρήση ειδικών εργαλείων.

Το παραπάνω μας δείχνει ότι τελικά το δίκτυο δεν είναι όσο ανώνυμο πιστεύαμε ότι είναι, αλλά αντιθέτως έχει πολλά κενά ως προς την ανωνυμία του και καθώς πολλά άλλα κρυπτονομίσματα χρησιμοποιούν τον κώδικα του bitcoin αυτούσιο, το πρόβλημα επάγεται σε όλα τα κρυπτονομίσματα.

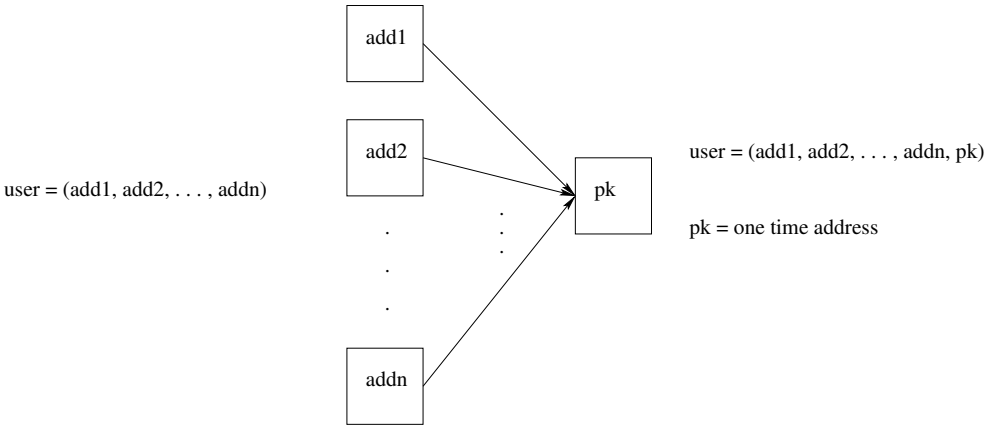
Το δίκτυο των χρηστών, όπως παρουσιάστηκε στο προηγούμενο κεφάλαιο, αποτελείται από δένδρα που αντιπροσωπεύουν τη ροή των bitcoin μεταξύ διευθύνσεων μιας χρήσης οι οποίες δεν συνδέονται με άλλες διευθύνσεις. Επίσης, παρατηρούμε ότι το δίκτυο έχει και δομές κυκλωμάτων. Έχοντας υπόψιν μας την παραπάνω δομή θα παρουσιάσουμε δύο υποθέσεις στις οποίες θα βασιστούμε παρακάτω για την άρση της ανωνυμίας των χρηστών του δικτύου [4].

1. Αν δύο ή περισσότερες διευθύνσεις εμπλέκονται σε μια συναλλαγή, τότε αυτές ανήκουν στον ίδιο χρήστη. Δηλαδή, για κάθε συναλλαγή t, όλα τα $pk \in input(t)$ ανήκουν στον ίδιο χρήστη.



Σχήμα 2.2: Υπόθεση 1

2. Οι διευθύνσεις μιας χρήσης που δέχονται bitcoin από άλλες διευθύνσεις, ανήκουν στον ίδιο χρήστη που ανήκουν και οι διευθύνσεις που εμπλέκονται στην ροή των bitcoin. Δηλαδή, για κάθε συναλλαγή t, ο χρήστης των διευθύνσεων που ανήκουν στο σύνολο $inputs(t)$ είναι και ο χρήστης των διευθύνσεων μιας χρήσης $pk \in outputs(t)$.



Σχήμα 2.3: Υπόθεση 2

Χρησιμοποιώντας όλα τα παραπάνω καθώς και παράγοντες εκτός δικτύου, μπορούμε να φτάσουμε σε άρση ανωνυμίας των χρηστών. Στο επόμενο εδάφιο θα δούμε το πως γίνεται να βρούμε τη φυσική τοποθεσία ενός χρήστη.

2.2 Εντοπισμός της Τοποθεσίας Χρήστη

Στο bitcoin δίκτυο δεν υπάρχει κάποιος φάκελος που να περιέχει πληροφορίες για κάθε χρήστη. Στο εδάφιο αυτό θα προσπαθήσουμε να φτιάξουμε κάτι τέτοιο, δηλαδή να βρούμε έναν τρόπο χρησιμοποιώντας στοιχεία εκτός δικτύου, που θα χαρακτηρίζουν έναν χρήστη. Πολλές φορές, ένα από τα χαρακτηριστικά θα είναι μέχρι και η φυσική του τοποθεσία. Όπως μπορεί να καταλάβει ο καθένας μια τέτοια επίθεση θα δημιουργήσει πολύ μεγάλο πλήγμα στην ανωνυμία που προσφέρει το σύστημα.

Το πρώτο βήμα θα ήταν κάποιος να κατεβάσει τη blockchain και χρησιμοποιώντας τις παραπάνω υποθέσεις να ομαδοποιήσει διευθύνσεις με χρήστες ώστε να βγουν τα πρώτα συμπεράσματα. Γνωρίζοντας πλέον το αριθμό (κατά προσέγγιση) των χρηστών στο δίκτυο, θα προσπαθήσουμε να βρούμε πληροφορίες σχετικά με τη φυσική τοποθεσία των χρηστών.

Υπάρχουν μεγάλοι οργανισμοί και υπηρεσίες οι οποίες δέχονται πληρωμές με bitcoin. Υπηρεσίες όπως αυτές της μετατροπής χρημάτων σε bitcoin (ξέπλυμα μαύρου χρήματος) προϋποθέτουν τη δήλωση στοιχείων του χρήστη, στοιχεία όπως διεύθυνση ηλεκτρονική, φυσική, ακόμα και τραπεζικούς λογαριασμούς. Αν γίνει μια τέτοια πληροφορία γνωστή τότε η ανωνυμία του χρήστη δε θα υφίσταται πλέον. Ένας τρόπος με τον οποίο μπορούν να παρθούν τέτοιες πληροφορίες είναι τα cookies [8].

Πληροφορίες σχετικά με τις διευθύνσεις χρηστών μπορούμε να πάρουμε και από άλλες πηγές. Για παράδειγμα, φιλανθρωπικές οργανώσεις που δέχονται bitcoins πρέπει να έχουν τη διεύθυνσή τους δημόσια ώστε να μπορεί κανείς να κάνει κατά-

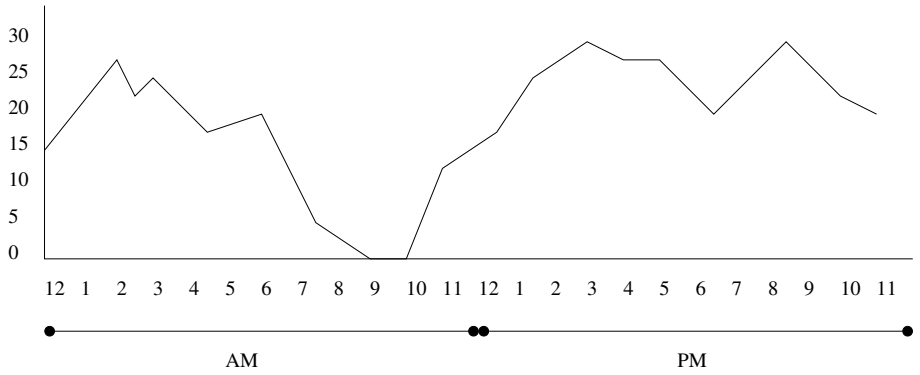
θεση. Με αυτόν τον τρόπο μπορούμε να κάνουμε τη σύνδεση μεταξύ διευθύνσεων μεγάλων τέτοιων οργανισμών και της τοποθεσίας τους.

Επίσης, πολλές φορές σε forums, χρήστες αποκαλύπτουν κάποια ή κάποιες διευθύνσεις τους. Έτσι μπορεί να γίνει σύνδεση μεταξύ ενός φυσικού προσώπου και μιας διεύθυνσης. Σε σελίδες όπως το Bitcoin Faucet χρήστες κάνουν δωρεές σε άλλους χρήστες ποσά από bitcoins (συνήθως μικρά ποσά). Για να μη γίνεται κατάχρηση αυτής της δυνατότητας, διατηρείται ιστορικό με τις δωρεές αυτές. Η έννοια της κατάχρησης σημαίνει ότι για παράδειγμα θα μπορούσε κάποιος να δημιουργήσει διάφορες διευθύνσεις και μέσω αυτής της σελίδας να στέλνει χρήματα ώστε να μην είναι εύκολη η σύνδεση μεταξύ των προηγούμενων διευθύνσεών του και των νέων.

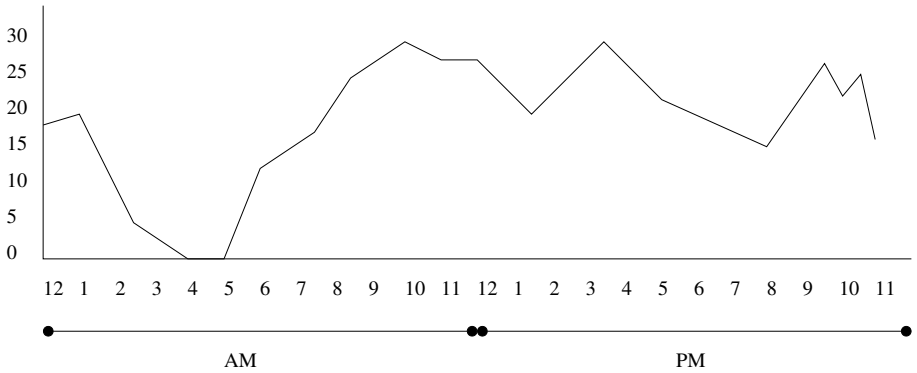
Το ιστορικό με τις δωρεές περιέχει το ποσό που δόθηκε και τη διεύθυνση στην οποία στάλθηκε αυτό το ποσό. Με αυτό το ιστορικό είναι πιθανό να γίνει σύνδεση μεταξύ της διεύθυνσης του δέκτη και του δημόσιού του κλειδιού. Με επαγωγή, μπορούμε να φτάσουμε και σε άλλες διευθύνσεις που ανήκουν στο ίδιο δημόσιο κλειδί και έτσι να βρούμε όλες τις συναλλαγές που σχετίζονται με την αρχική διεύθυνση.

Ας δούμε όμως πως μπορούμε να βρούμε την τοποθεσία στην οποία βρίσκεται κάποιος χρήστης. Παίρνοντας ως παράδειγμα χρήστες οι οποίοι οικειοθελώς δημοσίευσαν τις διευθύνσεις τους, έγινε δυνατή η σύνδεση μεταξύ των διευθύνσεών τους και της φυσικής τους τοποθεσίας. Για τη διερεύνηση, θα χρησημοποιηθούν οι ζώνες ώρας *UTC* καθώς και το Time Stamp, δηλαδή η ώρα που καταγράφηκε κάποια συναλλαγή.

Η μεθοδολογία [5] είναι η εξής, αρχικά λαμβάνουμε υπόψη μας μόνο τις εξερχόμενες συναλλαγές ενός χρήστη καθώς είναι οι μόνες που χρειάζονται την παρέμβαση του ίδιου (τις εισερχόμενες συναλλαγές τις δέχεται αυτόματα). Έπειτα, κοιτάζουμε τη χρονική στιγμή (Time Stamp) που έγινε κάθε συναλλαγή. Στην blockchain η χρονική στιγμή που αναγράφεται στη συναλλαγή είναι σε $UTC-0$ (για να μη γίνει γνωστή η περιοχή που βρίσκεται ο χρήστης και να υπάρχει μια κοινή ώρα για όλο το δίκτυο).



Σχήμα 2.4: Ιστόγραμμα Συναλλαγών/Ώρας UTC-0



Σχήμα 2.5: Ιστόγραμμα Συναλλαγών/Ώρας UTC-5

Τέλος, χρησιμοποιούμε δύο τεστ για τον προσδιορισμό της τοποθεσίας ενός χρήστη.

1. Την ανάλυση της μεθόδου του μεσημεριού, βάση της οποίας οι περισσότεροι χρήστες κάνουν περισσότερες συναλλαγές μεταξύ του χρονικού διαστήματος $12:00-00:00$ απ' ότι στο χρονικό διάστημα $00:00-12:00$.

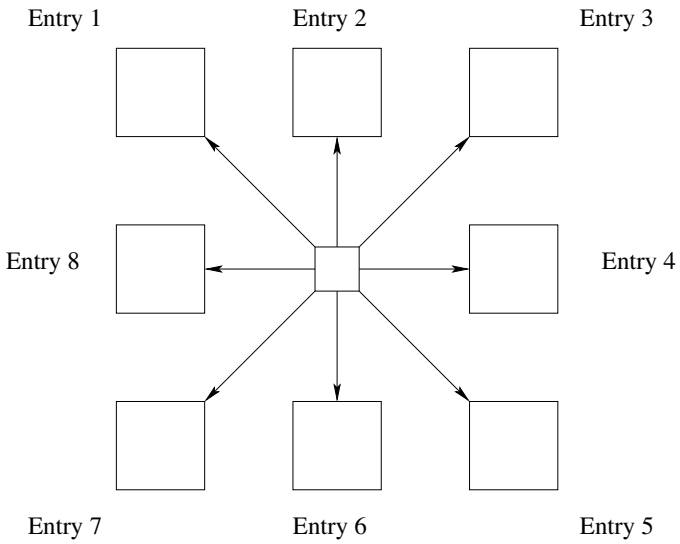
2. Την ανάλυση της μεθόδου του πρωινού, βάση της οποίας οι περισσότεροι άνθρωποι δεν κάνουν συναλλαγές τις πολύ πρωινές ώρες.

Παράδειγμα 2.2.1 *Με τις παραπάνω παραδοχές για την εύρεση της φυσικής περιοχής ενός προσώπου αρχικά φτιάχνουμε ένα ιστόγραμμα με την αναλογία πλήθους συναλλαγών ανά ώρα (Σχήμα 2.4). Έπειτα, ψάχνουμε να βρούμε το διάστημα μιας ώρας με τις λιγότερες συναλλαγές. Αν περισσότερα από ένα διάστηματα έχουν το ίδιο μικρό πλήθος συναλλαγών τότε δημιουργούμε ένα σύνολο με όλες αυτές τις ώρες. Βάση των παραπάνω παραδοχών, η ώρα με το μικρότερο πλήθος συναλλαγών αντιστοιχίζεται στην ώρα 4:00 - 5:00 το πρωί τοπική ώρα του χρήστη. Έτσι για παράδειγμα, αν αυτή η ώρα με το μικρότερο αριθμό συναλλαγών έχει καταγραφεί στο ιστόγραμμα ως 9:00 - 10:00 το πρωί $UTC-0$ (ώρα που αναγράφεται στην blockchain) τότε μπορεί να γίνει η παραδοχή πως ο χρήστης βρίσκεται στη ζώνη χρόνου $UTC-5$ (Σχήμα 2.5).*

Αξίζει να σημειωθεί, πως στη μέθοδο αυτή μπορεί να υπάρχει σφάλμα για τον λόγο ότι βασίζεται σε συνηθισμένες συμπεριφορές των ανθρώπων. Αν για παράδειγμα, ένας χρήστης εργάζεται ως νυχτοφύλακας, τότε θα άλλαζαν όλα τα δεδομένα καθώς η παραδοχή ότι τα ξημερώματα δεν κάνει πολλές συναλλαγές δε θα ευσταθούσε.

2.3 Bitcoin P2P Δίκτυο

Στο bitcoin δίκτυο η σύνδεση ενός χρήστη γίνεται Peer-2-Peer, δηλαδή γίνεται μια απευθείας σύνδεση μεταξύ δύο οντοτήτων χωρίς την ύπαρξη ενδιάμεσου κόμβου.



Σχήμα 2.6: Σύνδεση client (τετράγωνο στη μέση) σε servers

Ένας τέτοιος κόμβος θα μπορούσε να είναι ένας εξυπηρετητής. Για παράδειγμα, όταν κάποιος θέλει να στείλει ένα e-mail σε κάποιον άλλον η συνομιλία γίνεται μέσω των εξυπηρετητών της εκάστοτε εταιρίας. Η IP του αποστολέα στέλνει το πακέτο πληροφοριών στον εξυπηρετητή του πάροχου και από εκεί προωθείται στην IP του προορισμού. Σε αντίθετη περίπτωση, μια P2P σύνδεση έχει αποτέλεσμα, ένα πακέτο πληροφορίας από την IP του αποστολέα στέλνεται απευθείας στην IP του παραλήπτη χωρίς τη παρέμβει άλλο σύστημα.

Μια τέτοια επικοινωνία όμως έχει μεγάλα μειονεκτήματα. Αρχικά, δε μπορεί να σταλθεί μεγάλο μέγεθος πληροφορίας λόγω περιορισμού του διαδικτύου και έτσι η κάθε IP στέλνει διαρκώς, πολλά και μικρά πακέτα. Από αυτό και μόνο μπορεί να καταλάβει κανείς ότι είναι πολύ εύκολο να χαθεί η πληροφορία αρκεί να χαθεί ένα τέτοιο μικρό πακέτο. Για τον λόγο αυτό, η σύνδεση στο bitcoin δίκτυο αν και P2P γίνεται μέσω ενός ακρυπτογράφητου καναλιού TCP.

Ένα κανάλι TCP (Transmission Control Protocol) χρησιμοποιείται μαζί με την IP για τη βέβαιη μεταφορά πακέτων. Τα πλεονεκτήματα αυτού του καναλιού είναι ότι δημιουργούνται συνδέσεις μεταξύ δύο συσκευών και έτσι όταν η πρώτη συσκευή στείλει πακέτα πληροφοριών στον άλλον, ο δεύτερος ανταποκρίνεται επιβεβαιώνοντας ότι έλαβε το πακέτο, έτσι σιγουρευόμαστε ότι όλη η πληροφορία έχει σταλθεί. Τέλος, όταν κάποιος θέλει να σταματήσει τη μεταξύ τους συνδέση απλά μπορεί να κλείσει το κανάλι επικοινωνίας που έχουν δημιουργήσει (Σχήμα 2.6).

Στο δίκτυο δεν υπάρχει κάποιο σύστημα αυθεντικοποίησης και έτσι κάθε κόμβος του συστήματος αποθηκεύει μια λίστα από διαφορετικές IP που συνδέονται σε αυτόν. Για να αποφευχθούν επιθέσεις του τύπου denial-of-service το πρωτόκολλο έχει μειώσει το μέγεθος πληροφοριών που δέχεται από τους χρήστες. Υπάρχει επίσης

σύστημα τιμωρίας για κάθε χρήστη που προσπαθήσει να διαπράξει απάτη. Κάθε τέτοια προσπάθεια θα του δώσει μια ποινή κάποιων πόντων. Όταν οι πόντοι φτάσουν τους εκατό, τότε η IP δεν μπορεί να συνδεθεί στο δίκτυο για μία μέρα.

Στο bitcoin δίκτυο γίνεται διαχωρισμός μεταξύ των servers και των clients. Server ονομάζεται μια οντότητα του συστήματος που μπορεί να δεχτεί εισερχόμενες συνδέσεις. Καθένας από αυτούς πρέπει να διατηρεί 8 εξερχόμενες ανοιχτές συνδέσεις. Κάθε server μπορεί να δεχτεί μέχρι και 117 εισερχόμενες συνδέσεις (συνολικά 125). Αν κάποια από τις 8 χαθεί, το σύστημα προσπαθεί να την αντικαταστήσει, ενώ αν διατηρηθούν και οι 8 τότε τα πράγματα θα παραμείνουν έτσι μέχρι ο server να κάνει επανεκκίνηση. Στην περίπτωση του client, οι 8 κόμβοι με τους οποίους διατηρεί εξωτερικές συνδέσεις, ονομάζονται κόμβοι εισόδου. Τέλος, κάθε IP μπορεί να έχει οσοδήποτε μεγάλο αριθμό συνδέσεων αρκεί να μη παραβιάζεται κάτι από τα παραπάνω.

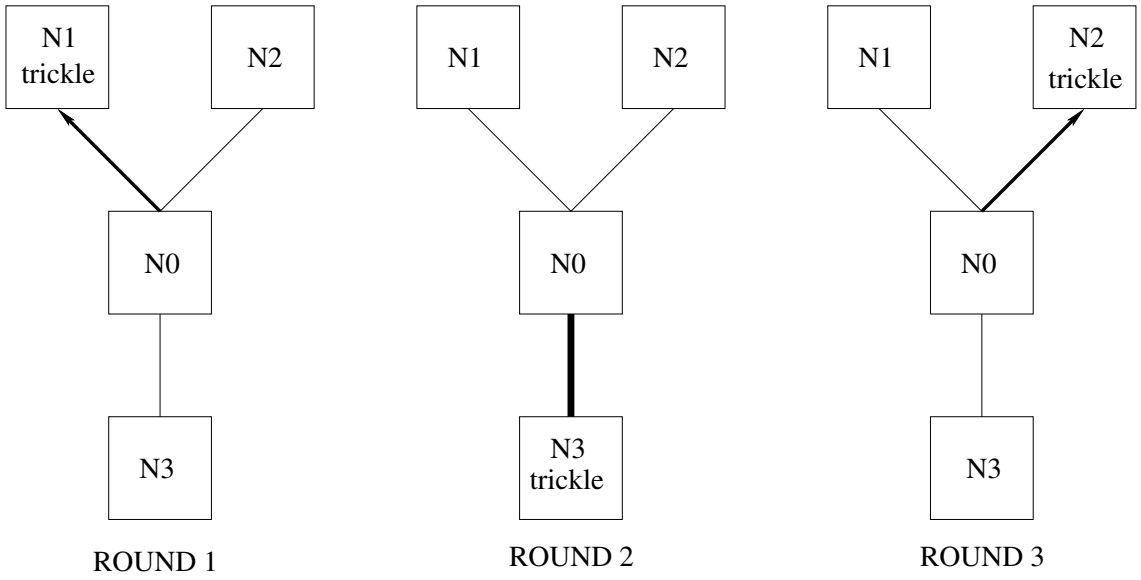
2.3.1 Διάδοση Διεύθυνσης

Το πρωτόκολλο του bitcoin δικτύου έχει αναπτύξει έναν μηχανισμό διάδοσης μιας διεύθυνσης η οποία ψάχνει να βρει κάποιον server για να συνδεθεί. Κάθε server ή client διατηρεί μια λίστα στην οποία αναγράφονται διευθύνσεις άλλων servers ή clients καθώς και το Time Stamp τους που δηλώνει το πότε συνδέθηκαν.

Οι servers ή clients μπορούν να ζητήσουν τη λίστα αυτή ο ένας από τον άλλον. Κάθε φορά που ένας κόμβος δέχεται ένα τέτοιο μήνυμα, αποφασίζει σε ποια διεύθυνση από αυτές της λίστας θα προωθεί στους γείτονές της. Αρχικά ελέγχει αν ο συνολικός αριθμός των διευθύνσεων του μηνύματος δεν ξεπερνά τους 10 και δεύτερον αν το Time Stamp της διεύθυνσης δεν ξεπερνάει τα 10 λεπτά. Αν τίποτα από τα δύο δεν ισχύει, τότε η διεύθυνση δεν προωθείται, διαφορετικά η διεύθυνση θα σταλεί σε δύο γείτονες του κόμβου στην περίπτωση που θεωρείται προσβάσιμη, αλλιώς τη προωθεί σε έναν.

Ορισμός 2.3.1 *Μια διεύθυνση θεωρείται προσβάσιμη από κάποιου κόμβο αν ο κόμβος έχει δικτυακή διεπαφή με διεύθυνση από την ίδια οικογένεια διευθύνσεων (IPv4, IPv6, OnionCat). Σε διαφορετική περίπτωση, η διεύθυνση ονομάζεται μη προσβάσιμη.*

Για να διαλέξει ο κόμβος που θα προωθήσει την διεύθυνση κάνει το εξής, για κάθε γείτονά του υπολογίζει μια τιμή κατακερματισμού συνθέτοντας τη διεύθυνση που θα προωθηθεί, ένα κρυφό κλειδί (secret salt), τη τρέχουσα μέρα, και τη διεύθυνση μνήμης στη βάση δεδομένων που περιγράφει τον γείτονα. Αξίζει να σημειωθεί ότι η παραπάνω τιμή παραμένει σταθερή για μια μέρα. Στη συνέχεια, ο server ή client ταξινομεί τη λίστα σύμφωνα με τις τιμές κατακερματισμού και επιλέγει τις δύο πρώτες ή τη πρώτη ανάλογα με τη προσβασιμότητα.



Σχήμα 2.7: Διαδικασία Trickling

Η διαδικασία προώθησης δε γίνεται απευθείας, αλλά μετά από κάποιο χρονικό διάστημα μπαίνει σε ουρά αναμονής και επιλέγεται τυχαία ένας κόμβος για να προωθηθεί η διεύθυνση. Ο κόμβος αυτός ονομάζεται trickle κόμβος και η διαδικασία trickling. Για να καταλάβουμε τη διαδικασία του trickling θα δούμε το παράδειγμα παρακάτω.

Παράδειγμα 2.3.1 *Έστω η διαδικασία του trickling όπως περιγράφεται στο Σχήμα 2.7. Έστω ότι ο κόμβος N_0 δέχεται ένα μήνυμα με τη διεύθυνση A_0 από τον κόμβο N_3 και έστω ότι αποφασίζει να το προωθήσει στους κόμβους N_1 και N_2. Οι κόμβοι αυτοί θα ονομάζονται υπεύθυνοι για την A_0. Στον πρώτο γύρο ο κόμβος N_1 επιλέγεται ως trickle κόμβος και η διεύθυνση προωθείται σε αυτόν, ενώ η προώθηση προς τον κόμβο N_2 βρίσκεται σε αναμονή. Μετά από ένα σταθερό χρονικό διάστημα (100ms) στον δεύτερο γύρο, ο κόμβος N_3 επιλέγεται ως trickle κόμβος και έτσι δεν γίνεται καμία προώθηση. Μετά από 100ms ως trickle κόμβος επιλέγεται ο N_2 και η διεύθυνση A_0 προωθείται και σε αυτόν καθιστώντας του και αυτόν ως υπεύθυνο κόμβο για την A_0. Φαίνεται πως η τυχαία επιλογή κόμβου μπορεί να δημιουργήσει καθυστερήσεις ανάμεσα στους γύρους.*

Κάπως έτσι κάθε σύνδεση στο bitcoin δίκτυο είναι καταγεγραμμένη ως προωθημένη. Πριν προωθηθεί οποιαδήποτε σύνδεση, γίνεται έλεγχος αν η ίδια διεύθυνση έχει προωθηθεί ξανά. Αυτή η μνήμη διαγράφεται κάθε μέρα, ενώ ένας server ή client μπορεί να αποθηκεύσει μέχρι και 20480 διαφορετικές διευθύνσεις (μια IP μπορεί να συνδεθεί με πολλές διαφορετικές διευθύνσεις).

2.3.2 Ανακάλυψη Διεύθυνσης

Κάθε server ή client με την εκκίνηση του παίρνει διευθύνσεις IP οι οποίες περιλαμβάνουν τις δικτυακές διεπαφές του καθώς και τις διευθύνσεις IP όπως αυτές εμφανίζονται στο διαδίκτυο. Για το λόγο αυτό ακολουθεί μια διαδικασία ανακάλυψης διευθύνσεων και για κάθε μια που ανακαλύπτει της δίνει κάποιο βαθμό. Στις τοπικές διεπαφές δίνει 1, και στις εξωτερικές IP δίνει 4, και στην περίπτωση που συμίπτουν αθροίζει και δίνει 5. Έπειτα, αρχίζει να κάνει εξωτερικές συνδέσεις προς κάποιον server. Αρχικά ανταλλάζουν μηνύματα και ο client προβάλει τις διευθύνσεις του με φθίνουσα σειρά ως προς τους βαθμούς. Ο server ακολουθεί τη διαδικασία του trickling. Ο client ακολουθεί τη διαδικασία και συνδέεται με άλλους 7 εξωτερικούς servers.

2.3.3 Διάδοση Συναλλαγής

Τέλος, υπάρχει και η διαδικασία της προώθησης μιας συναλλαγής. Για να γίνει κάτι τέτοιο, θα πρέπει να ακολουθηθούν κάποια βήματα. Αρχικά, ο αποστολέας μεταδίδει τον κατάλογο των τιμών κατακερματισμού των συναλλαγών. Ο παραλήπτης επιχειρεί κάποιους ελέγχους στη συναλλαγή και αν είναι πετύχουνζητάει το περιεχόμενο της συναλλαγής. Όταν το λάβει, το μεταδίδει και στους υπόλοιπους server.

Όταν ένας client κάνει μια συναλλαγή, την προωθεί σε όλους τους γείτονές του. Έπειτα, υπολογίζει τη τιμή κατακερματισμού της ποσότητας που προκύπτει από τη συνένωση της συναλλαγής με ένα μυστικό κλειδί. Αν τα δύο τελευταία ψηφία της τιμής αυτής είναι μηδέν, τότε προωθείται στους 8 κόμβους. Σε διαφορετική περίπτωση θα περιμένει σε μια ουρά αναμονής μέχρι να επιλεγεί για εξωτερική σύνδεση όταν κάποιος γείτονάς του γίνει trickle κόμβος.

Όταν γίνει η λήψη της συναλλαγής, τότε αυτή θα προωθηθεί σε όλους τους γείτονες του server, με τον τρόπο που περιγράφτηκε παραπάνω. Κάθε server καταγράφει όλες τις συναλλαγές που έχει προωθήσει σε κάθε σύνδεση. Αν κάποια συναλλαγή έχει ήδη σταλεί μέσω μιας σύνδεσης, τότε δε θα σταλεί ξανά. Επίσης, κάθε server κρατάει σε μια μνήμη και όλες τις συναλλαγές που δέχτηκε. Αν δεχτεί μια συναλλαγή με την ίδια τιμή κατακερματισμού με αυτή που έχει στη μνήμη ή με αυτές στην blockchain τότε αρνείται την προώθηση.

Όπως γίνεται αντιληπτό, η διαδικασία συναλλαγής είναι αρκετά ασφαλής και ανώνυμη καθώς υπάρχουν πολλές δικλίδες ασφαλείας. Ένας τρόπος όμως που μπορεί να υπάρξει άμεση κατάρρευση του συστήματος ανωνυμίας είναι η IP. Καθώς είπαμε ότι η σύνδεση δεν είναι κρυπτογραφημένη, και πως κάθε σύνδεση καταγράφεται, είναι εύκολο κάποιος να βρει την IP του άλλου. Για τον λόγο αυτό, η είσοδος στο bitcoin δίκτυο συνίσταται να γίνεται μέσω Tor καθώς και με χρήση VPN, έτσι ώστε η IP που θα εμφανίζεται στο σύστημα να μην είναι η πραγματική

και να είναι δύσκολο να γίνει η σύνδεση μεταξύ μιας διεύθυνσης και ενός φυσικού προσώπου.

2.4 Επίθεσεις στην Τοπολογία Δικτύου

Στο εδάφιο αυτό θα φτιάξουμε μια τοπολογία και θα δούμε επιθέσεις για την εύρεση διευθύνσεων χρηστών του δικτύου που δεν χρησιμοποιούν Tor και VPN για την ασφαλή πλοήγησή τους μέσα στο bitcoin δίκτυο. Επίσης, οι στόχοι αυτής της επίθεσης στην Peer-2-Peer σύνδεση είναι στους clients και όχι στους servers του δικτύου, δηλαδή στους κόμβους που δεν δέχονται συνδέσεις, αντ' αυτού όμως συνδέονται οι ίδιοι σε 8 κόμβους εισόδου (entry nodes - servers).

Γνωρίζουμε πως αν ένας client συνδεθεί στο δίκτυο μέσω ενός κόμβου εισόδου, το πρώτο πράγμα που κάνει, όπως είδαμε παραπάνω, είναι να προωθήσει τη διεύθυνση του. Έστω ότι ο client είναι ο C και η διεύθυνσή του είναι η C_a. Η διεύθυνσή IP C_a θα καταγραφεί όπως εμφανίζεται στο διαδίκτυο (από τη στιγμή που δεν χρησιμοποιεί *Tor*).

Ο επιτιθέμενος σε αυτή τη περίπτωση, αν είναι συνδεδεμένος σε κάποιον κόμβο εισόδου λαμβάνει την διεύθυνση C_a με κάποια πιθανότητα, η οποία εξαρτάται από το πλήθος των συνδέσεων που έχει κάνει. Τα παραπάνω συνοψίζονται στην ακόλουθη στρατηγική.

1. Συνδέεται σε W bitcoin servers, με W να είναι ένα πλήθος πολύ κοντά στο συνολικό πλήθος των servers στο δίκτυο.
2. Για κάθε διεύθυνση C_a που προωθείται, καταγράφει το σύνολο E'_{C_a} των servers οι οποίοι προώθησαν τη διεύθυνση C_a στις συσκευές του επιτιθέμενου.

Αυτή είναι μια πολύ καλή μέθοδος για να πάρει ο επιτιθέμενος τη διεύθυνση του C, όμως έχει κάποια βασικά προβλήματα.

1. Ο κόμβος εισόδου μπορεί να στείλει τη διεύθυνση C_a και σε κόμβους που δεν είναι συνδεδεμένος ο επιτιθέμενος.
2. Κάθε client που συνδέεται στο δίκτυο δε συνδέεται και με τους 8 κόμβους εισόδου ταυτόχρονα (όπως είδαμε και παραπάνω), αλλά υπάρχει ένα χρονικό κενό μεταξύ των συνδέσεών του.

Στην περίπτωση που η διεύθυνση C_a φτάσει στον επιτιθέμενο μέσω κάποιου άλλου κόμβου πέραν των κόμβων εισόδου, τότε η διαδικασία θεωρείται ότι έχει "θόρυβο" και το σύνολο E'_{C_a} έχει λάθος στοιχεία. Για να διορθωθεί το πρόβλημα του θορύβου, ακολουθούμε μια τεχνική (Noise Reduction Technique).

Αρχικά, μπορεί να θεωρηθεί πως η διεύθυνση IP του client έχει χρησιμοποιηθεί ξανά στο δίκτυο και δεν είναι αυτή η πρώτη φορά. Αν συμβαίνει κάτι τέτοιο, είδαμε ότι το δίκτυο κρατάει ιστορικό των συνδέσεων που έχουν γίνει και έτσι μπορεί από εκεί να την πάρει ο επιτιθέμενος. Στην περίπτωση που αυτός γνωρίζει τη C_a, μπορεί να εμποδίσει τη περαιτέρω μετάδοσή της στο δίκτυο. Αυτό συμβαίνει καθώς αν η C_a έχει σταλεί από έναν client A σε έναν άλλον B, τότε δε θα προωθηθεί ξανά. Με άλλα λόγια, αυτό που θα κάνει είναι να μεταδώσει τη διεύθυνση σε όλους τους κόμβους που είναι συνδεμένος. Η διαδικασία αυτή επαναλαμβάνεται κάθε 10 λεπτά. Ο επιτιθέμενος προσδοκά αν ένας client επανασυνδεθεί στο δίκτυο η διεύθυνση C_a θα προωθηθεί από τους servers εισόδου στον επιτιθέμενο. Ακόμα και να μη προωθηθεί όμως, η προώθηση προς αυτόν από κάποιον άλλον κόμβο πέρα των κόμβων εισόδου θα σταματήσει.

Έτσι, σε κάποιο χρονικό διάστημα, ο επιτιθέμενος θα καταλήξει να έχει συνδεθεί σε ένα ποσοστό κόμβων εισόδου από το συνολικό πλήθος αυτών που είναι συνδεμένος ο C. Το κλάσμα αυτό συμβολίζεται με p_{addr} και εξαρτάται από το πλήθος των συνδέσεων που έχει κάνει ο επιτιθέμενος. Για παράδειγμα, αν ο επιτιθέμενος έχει κάνει 35 συνδέσεις σε κόμβους εισόδου όπου στον καθένα έχουν συνδεθεί 90 clients, τότε μπορεί να είναι σίγουρος ότι έχει συνδεθεί με 4 από τους 8 κόμβους εισόδου που είναι συνδεμένος ο C, άρα το $p_{addr} = 0.5$. Ο υπολογισμός του p_{addr} θα δοθεί παρακάτω.

Οπότε το αποτέλεσμα αυτής της επίθεσης είναι ότι, καθώς ο επιτιθέμενος προωθεί τη C_a κάθε server στο δίκτυο διαλέγει δύο κόμβους υπεύθυνους για την διάδοση της διεύθυνσης. Ο επιτιθέμενος έχει συνδεθεί σε ένα πλήθος W από servers, αριθμός πολύ κοντά στο συνολικό πλήθος, και περιμένει ένας από τους κόμβους αυτούς να γίνει υπεύθυνος για τη μετάδοση της διεύθυνσης. Όταν ο C συνδεθεί σε κάποιον κόμβο εισόδου e_1 θα προωθήσει τη διεύθυνσή του. Αν ένας από τους κόμβους έχει αντικαταστήσει έναν από τους υπεύθυνους κόμβους για τη διάδοση της διεύθυνσης τότε ο επιτιθέμενος θα μάθει πως ο C έχει συνδεθεί στον κόμβο e_1 όπως επίσης δε θα προωθήσει τη διεύθυνση στο δίκτυο. Αν όμως δεν συμβεί αυτό τότε η επίθεση αυτή αποτυγχάνει.

2.4.1 Άρση Ανωνυμίας

Σε αυτό το εδάφιο θα παρουσιαστεί η επίθεση για την άρση της ανωνυμίας η οποία βασίζεται στην επίθεση στην τοπολογία του δικτύου. Η άρση της ανωνυμίας γίνεται σε τέσσερα βήματα:

1. Παίρνουμε τη λίστα S των servers. Αυτό το σύνολο ανανεώνεται τακτικά καθώς όπως είδαμε μπορεί να συνδεθεί ή να αποσυνδεθεί από το δίκτυο κάποιος server.

2. Κατασκευάζουμε ένα σύνολο C από clients που θέλουμε να κάνουμε άρση της

ανωνυμίας τους.

3. Εντοπίζουμε τους κόμβους εισόδου των clients του συνόλου C.

4. Παρακολουθούμε τους servers του συνόλου S και γίνεται σύνδεση των συναλλαγών με τους κόμβους εισόδου και στη συνέχεια με τους clients.

Κάνοντας τα παραπάνω βήματα καταλήγουμε στη δημιουργία ενός συνόλου

$$I = \{(IP, Id, PK)\}$$

όπου IP είναι η διεύθυνση IP του client ή του παρόχου του, Id ξεχωρίζει τους διαφορετικούς client που χρησιμοποιούν την ίδια IP και PK είναι η διεύθυνση που χρησιμοποιείται στη συναλλαγή, δηλαδή η κατακερματισμένη τιμή του δημόσιου κλειδιού. Ας δούμε πως εφαρμόζονται όμως αυτά τα βήματα.

Βήμα 1. Ο επιτιθέμενος παίρνει τη λίστα με όλους τους servers. Αρχικά έχουμε $S = \emptyset$. Ο επιτιθέμενος στέλνει μήνυμα ανάκτησης της διεύθυνσης γνωστών servers και η κάθε διεύθυνση P ελέγχεται αν είναι συνδεμένη τη δεδομένη στιγμή. Αν είναι τότε $S = S \cup \{P\}$. Ο επιτιθέμενος μπορεί να ξεκινήσει με ένα μικρό σύνολο και στη συνέχεια να συμπληρώνει με κάθε καινούρια IP που δέχεται. Ο επιτιθέμενος κάνει περίπου $m = 50$ συνδέσεις σε κάθε server.

Βήμα 2. Ο επιτιθέμενος διαλέγει ένα σύνολο C από clients που θέλει να κάνει άρση της ανωνυμίας τους (μέσω του διαδικτύου, του ιστορικού του δικτύου του bitcoin, ακόμα και του συνόλου του βήματος 1).

Βήμα 3. Ο επιτιθέμενος ανακαλύπτει του κόμβους εισόδου των clients όπως είδαμε στο προηγούμενο υποεδάφιο. Για έναν client P συμβολίζουμε το σύνολο των κόμβων εισόδου του με E_P. Είναι πιθανόν το $E_{P_1} \neq E_{P_2}$ ακόμα και αν οι P_1 και P_2 χρησιμοποιούν την ίδια IP. Για κάθε P που προωθεί τη διεύθυνσή του, ο επιτιθέμενος φτιάχνει το σύνολο $E'_P \subset E_P$. Καθώς έχουμε ότι περίπου $8 \cdot 10^3$ είναι οι πιθανοί κόμβοι εισόδου από τους 10^5 συνολικά servers και clients, προκύπτει ότι με τρεις κόμβους εισόδου μπορούμε να αναγνωρίσουμε έναν χρήστη αφού

$$\frac{10^5 \cdot 10^5}{(8 \cdot 10^3)^3} << 1.$$

Βήμα 4. Το βήμα αυτό τρέχει παράλληλα με τα παραπάνω. Ο επιτιθέμενος προσπαθεί να συνδυάσει τις συναλλαγές που εμφανίζονται στο δίκτυο με τα σύνολα E'_P. Παρακολουθεί τις συναλλαγές που γίνονται στο δίκτυο σε όλες τις συνδέσεις που έχει κάνει και για κάθε συναλλαγή T συλλέγει στο σύνολο R_T τις πρώτες q (περίπου $q = 10$) διευθύνσεις των servers που του τις προώθησε. Έπειτα, συγκρίνει τα E'_P και R_T και δημιουργεί ζευγάρια (P, T). Η σύγκριση των συνόλων γίνεται ως εξής:

- Ο επιτιθέμενος επιλέγει όλες τις δυνατές τριάδες από το E'_p και ψάχνει να βρει ποιες από αυτές εμφανίζονται στο R_T. Αν υπάρχει κάποια εμφάνιση, τότε παίρνει το ζευγάρι (P, T).
- Αν δεν υπάρχει εμφάνιση, τότε παίρνει δυάδες κόμβων ή απλούς κόμβους. Μπορεί να προκύψουν αρκετά ζευγάρια $\{(P_i, T)\}$ από τα οποία κάποια θα απορριφθούν με τις επόμενες συναλλαγές.

Παρατήρηση 2.4.1 *Το βήμα 4 της επίθεσης βασίζεται στο γεγονός ότι κάποιοι κόμβοι εσόδου ενός client είναι μέσα στους πρώτους που προωθούν μια συναλλαγή. Όταν ένας κόμβος λαμβάνει μια συναλλαγή αρχικά τρέχει κάποια τεστ και μετά τη προωθεί. Ο χρόνος που χρειάζεται για να προωθηθεί μια συναλλαγή στον επιτιθέμενο μέσω ενός κόμβου εισόδου είναι το άθροισμα του χρόνου μετάδοσης τεσσάρων μηνυμάτων, του χρόνο που χρειάζεται για να τρέξει 16 τεστ και το χρόνο που χρειάζεται για ένα πιθανό trickling. Ο χρόνος που χρειάζεται για να προωθηθεί μια συναλλαγή στον επιτιθέμενο μέσω ενός κόμβου που δεν είναι κόμβος εισόδου είναι περισσότερο από διπλάσιος. Έτσι ο πρώτος που θα διαδώσει ένα μήνυμα συναλλαγής θα είναι ένας κόμβος εισόδου του δικτύου.*

Παρατήρηση 2.4.2 *Η παραπάνω επίθεση προϋποθέτει από τον επιτιθέμενο να έχει ένα πολύ μεγάλο αριθμό συνδέσεων χωρίς να στέλνει μεγάλο όγκο δεδομένων. Αν θέλει κάποιος η επίθεση αυτή να μην είναι ανιχνεύσιμη, αρκεί να χρησιμοποιήσει κάποιο VPN ώστε να αλλάξει την IP του.*

Παρατήρηση 2.4.3 *Η τεχνική αυτή μας δίνει την ταυτότητα ενός χρήστη όσο είναι συνδεμένος και κάνει συναλλαγές. Ως εκ τούτου, όσο περισσότερες συναλλαγές κάνει τόσο μεγαλύτερη και η πιθανότητα της άρσης της ανωνυμίας του. Αξίζει να σημειωθεί ότι η άρση της ανωνυμίας γίνεται ακόμα και αν ο χρήστης χρησιμοποιεί διαφορετικά δημόσια κλειδιά και δεν υπάρχει σύνδεση μεταξύ τους στο δίκτυο συναλλαγών.*

Στο επόμενο εδάφιο θα παρουσιαστεί μια ανάλυση αυτών των επιθέσεων με τη χρήση μαθηματικών εργαλείων για πιο αυστηρά αποτελέσματα.

2.4.2 Ανάλυση των Επιθέσεων

Το ποσοστό επιτυχίας των παραπάνω επιθέσεων βασίζεται όπως είδαμε σε πολλούς παράγοντες. Η πιο σημαντική, που χρησιμοποιείται στην επίθεση στην τοπολογία του δικτύου, τα αποτελέσματα της οποίας χρησιμοποιούνται για την άρση της ανωνυμίας, είναι το ποσοστό συνδέσεων του επιτιθέμενου στο σύνολο των συνδέσεων όλων των κόμβων εισόδου ενός client στο δίκτυο το οποίο συμβολίσαμε με p_{addr}. Θα προσπαθήσουμε αρχικά να βρούμε τον αριθμό των συνδέσεων που χρειάζονται.

Αν υποθέσουμε ότι κάποιος κόμβος εισόδου έχει n συνδέσεις και ο επιτιθέμενος του προσθέσει άλλες m τότε το σύνολο των συνδέσεων θα είναι $N = n + m$. Η πιθανότητα να λάβει τη διεύθυνση με τη πρώτη φάση του trickling είναι $p_{addr}(n, N) = 1 - \frac{n}{N} \cdot \frac{n-1}{N-1}$. Η πιθανότητα ένας κόμβος του επιτιθέμενου να επιλεχθεί ως κόμβος υπεύθυνος για την προώθηση μιας συναλλαγής που δεν έχει προωθηθεί ακόμα είναι $p_{tx} = \frac{m}{N}$. Για $n = 90$ και $m = 35$ όπως το παράδειγμα που δώσαμε παραπάνω, έχουμε ότι $p_{addr} = 0.49$ και $p_{tx} = 0.28$. Υπενθυμίζουμε ότι ένας server μπορεί να έχει μέχρι 125 συνδέσεις (ο μεγαλύτερος αριθμός συνδέσεων που μπορεί να κάνει ένας επιτιθέμενος).

Υποθέτουμε ότι κάποια βήματα της επίθεσης αποτυγχάνουν. Τότε οι πρώτοι 10 servers ή clients που θα προωθήσουν τη συναλλαγή στον επιτιθέμενο θα είναι τυχαίοι. Αν στους 10 δεν υπάρχει υποσύνολο τριων στοχείων που να συμπίπτει με ένα σύνολο κόμβων εισόδου τότε η συναλλαγή δεν αναγνωρίζεται από τον επιτιθέμενο. Η πιθανότητα κάποιοι κόμβοι τυχαία να είναι συμπίπτουν με κόμβους εισόδου είναι

$$p_c = \begin{pmatrix} 10 \\ 3 \end{pmatrix} \cdot \begin{pmatrix} 10 \\ 3 \end{pmatrix} \cdot \left(\frac{1}{N}\right)^3$$

όπου N είναι το πλήθος των server στο δίκτυο.

Τώρα θα μετρήσουμε τη πιθανότητα ο επιτιθέμενος να προσθέσει λάθος κόμβο στο σύνολο των κόμβων εισόδου ενός client. Για να γίνει κάτι τέτοιο θα πρέπει ένας ή περισσότεροι κόμβοι εισόδου να προώθησαν τη C_a σε κάποιον άλλον κόμβο (όχι του επιτιθέμενου) όπου τουλάχιστον ένας από τους κόμβους που είναι υπεύθυνος για την προώθηση της C_a άλλαξε μετά τη προώθησή της από τον επιτιθέμενο. Έστω ότι ο επιτιθέμενος προώθησε τη C_a τη χρονική στιγμή t_0 και έστω $t_1 = t_0 + \Delta t$ η χρονική στιγμή που άλλαξε ο υπεύθυνος κόμβος. Η πιθανότητα να έγινε κάτι τέτοιο εξαρτάται από το πλήθος των κόμβων που αποσυνδέθηκαν τη χρονική στιγμή t_1 και συνδέθηκαν σε αυτόν τον κόμβο εισόδου τη χρονική στιγμή t_0. Τα αποτελέσματα μετά από πειράματα [6] δείχνουν πως όσο περισσότερες συνδέσεις έχει ένας κόμβος τόσο λιγότερο πιθανό είναι να αλλάξει ο κόμβος υπεύθυνος για τη μετάδοση σε χρονικό διάστημα Δt. Επίσης, η πιθανότητα εξαρτάται και από το πόσο μεγάλο είναι αυτό το διάστημα.

Τέλος, θα μελετήσουμε το ποσοστό επιτυχίας P_c αυτής της επίθεσης. Το ποσοστό αυτό εξαρτάται από το πλήθος των χαρακτηριστικών του διαδικτύου. Για τον υπολογισμό του θεωρούμε ότι ο επιτιθέμενος συνδέεται με όλους τους πιθανούς servers. Δίνοντας διάφορες τιμές στις παραμέτρους n, N παίρνουμε διάφορες τιμές για την p_{addr} και παίρνουμε το μέσο όρο αυτών και το συμβολίζουμε p_{addr}^{avg}. Για περίπου 50 συνδέσεις έχουμε $p_{addr}^{avg} = 0.34$ περίπου. Χρησιμοποιώντας το p_{addr}^{avg} και την κατανομή των 10 πρώτων κόμβων που προωθούν το μήνυμα συναλλαγών των clients στον επιτιθέμενο.

Αρχικά, έχουμε δύο τύπους. Υποθέτουμε ότι έχουμε N μπάλες. Αν κάθε μπάλα είναι κόκκινη με πιθανότητα p_a και πράσινη με πιθανότητα $1 - p_a$ τότε η πιθανότητα

Κόμβοι	Πιθανότητα
1	0.15
2	0.27
3	0.28
4	0.18
5	0.07
6	0.02
7	0.002
8	0.0002

Πίνακας 2.1: Η Πιθανότητα $\mathbb{P}_1(R;8)$.

R μπάλες να είναι κόκκινες είναι

$$\mathbb{P}_1(R;N) = \left(\begin{array}{c} N \\ R \end{array} \right) p_a^R (1-p_a)^{(N-R)}.$$

Τώρα υποθέτουμε ότι έχουμε R κόκκινες μπάλες και $N-R$ πράσινες. Υποθέτουμε ότι επιλέγουμε L τυχαίες μπάλες από τις N. Η πιθανότητα να υπάρχουν ακριβώς q κόκκινες μπάλες από τις L είναι

$$\mathbb{P}_2(q;L,R,N) = \frac{\left(\begin{array}{c} R \\ q \end{array} \right) \left(\begin{array}{c} N-R \\ L-q \end{array} \right)}{\left(\begin{array}{c} N \\ L \end{array} \right)}.$$

Χρησιμοποιώντας τις παραπάνω πιθανότητες, αν βρεθεί ένας κόμβος εισόδου με πιθανότητα $p_{addr}^{avg} = 0.34$ τότε μπορούμε να βρούμε R κόμβους εισόδου από τους 8 με πιθανότητα $\mathbb{P}_1(R;8)$ όπως φαίνεται στον Πίνακα 2.1 [6].

Τώρα θα βρούμε τη πιθανότητα να έχουμε L κόμβους εισόδου ανάμεσα στους 10 πρώτους που προωθούν το μήνυμα στον επιτιθέμενο. Υποθέτουμε ότι τα γεγονότα είναι ανεξάρτητα. Η πιθανότητα ο επιτιθέμενος να έχει ανακαλύψει τουλάχιστον M από τους L κόμβους εισόδου είναι:

$$\mathbb{P}_{suc}(M) = \sum_{q \geq M} \sum_{L \leq 8} \sum_{R \leq 8} \mathbb{P}_2(q;L,R,8) \cdot \mathbb{P}_1(R;8) \cdot \mathbb{P}_3(L).$$

Η πιθανότητα L από τους 8 κόμβους εισόδου να εμφανιστούν στους 10 πρώτους είναι $\mathbb{P}_3(L)$ όπως φαίνεται στον Πίνακα 2.2 [6]. Η πιθανότητα ο επιτιθέμενος να έχει βρει και τους L είναι όπως φαίνονται στον Πίνακα 2.3 [6]. Η πιθανότητα ο επιτιθέμενος να έχει βρει τους M από τους L είναι όπως φαίνονται στον Πίνακα 2.4 [6].

Κόμβοι	Πιθανότητα
1	0.02
2	0.055
3	0.1225
4	0.245
5	0.2125
6	0.2125
7	0.0925
8	0

Πίνακας 2.2: Η Πιθανότητα $\mathbb{P}_3(L)$.

L	Πιθανότητα
1	0.366
2	0.243
3	0.9
4	0.02
5	0.002

Πίνακας 2.3: Η Πιθανότητα $\mathbb{P}_{suc}(L)$.

2.5 Μέθοδοι Ανάλυσης της Ανωνυμίας

Στο εδάφιο αυτό θα δούμε μεθόδους ανάλυσης που χρησιμοποιούνται για την άρση της ανωνυμίας των χρηστών. Στις αναλύσεις αυτές χρησιμοποιήθηκαν εργαλεία για την απεικόνιση των αποτελεσμάτων, να σημειωθεί ότι στην παρουσίαση που θα γίνει αυτών των αναλύσεων θα χρησιμοποιηθούν τα διαγράμματα όπως δόθηκαν στη δημοσίευση των Fergal Reid, Martin Harrigan [6] λόγω της έλλειψης των συγκεκριμένων εργαλείων.

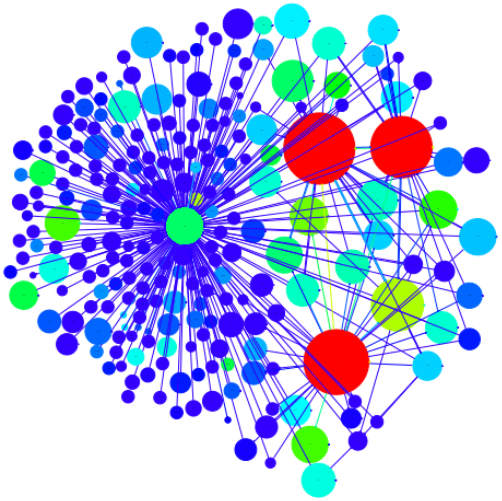
2.5.1 Εγωκετρική Ανάλυση

Όπως είδαμε στο προηγούμενο κεφάλαιο, από τη blockchain μπορούμε να δημιουργήσουμε δύο πολύ σημαντικά δίκτυα αυτά των συναλλαγών και των χρηστών. Εφαρμόζοντας τη δύσκολη διαδικασία της εύρεσης του δικτύου των χρηστών μπορούμε να το χρησιμοποιήσουμε για την περαιτέρω ανακάλυψη λεπτομερειών για έναν συγκεκριμένο χρήστη για τον οποίο θέλουμε να γίνει η άρση της ανωνυμίας του. Για παράδειγμα, μπαρούμε να ανακαλύψουμε το σύνολο των bitcoin που έχει σε κάποια διεύθυνσή του (δημόσιο κλειδί), αλλά ακόμα και το σύνολο των bitcoin που έχει σε όλες τις άλλες διευθύνσεις που διαχειρίζεται.

Γνωρίζοντας το σύνολο των bitcoin που έχει και από τη στιγμή που γνωρίζουμε τις

M	Πιθανότητα
1	0.721
2	0.355
3	0.112
4	0.022
5	0.002

Πίνακας 2.4: Η Πιθανότητα $\mathbb{P}_{suc}(M)$.



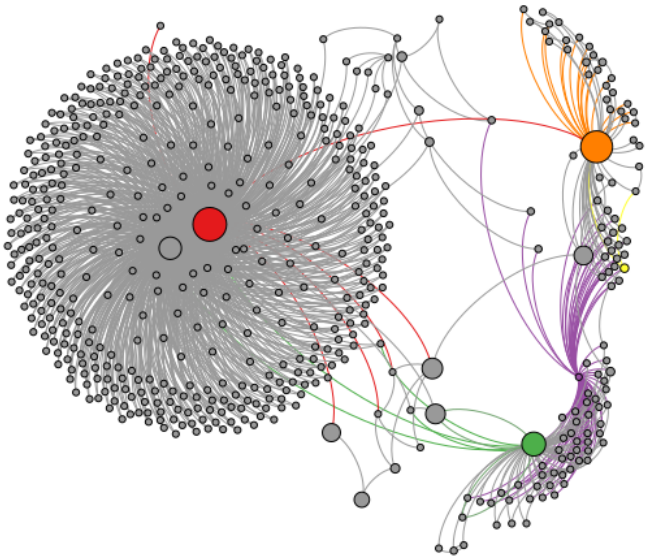
Σχήμα 2.8: Το Εγωκεντρικό Δίκτυο

διευθύνσεις του μπορούμε να βρούμε στην blockchain το σύνολο όλων τον συναλλαγών προς και από αυτές. Έτσι μπορούμε να δημιουργήσουμε (και να απεικονίσουμε) ένα νέο δίκτυο, το εγωκεντρικό.

Ορισμός 2.5.1 *Εγωκεντρικό δίκτυο ονομάζεται ένας μη κατευθυνόμενος υπογράφος του δικτύου χρηστών όπου αποτελείται από μια κεντρική κορυφή που συμβολίζει τον χρήστη του οποίο μελετάμε και όλες οι υπόλοιπες κορυφές συμβολίζουν όλους τους διαφορετικούς χρήστες με τους οποίους συναλλάσσεται. Οι ακμές του γράφου αποτελούν τις συναλλαγές μεταξύ αυτών των κορυφών.*

Αξίζει να σημειωθεί πως σε ένα εγωκεντρικό δίκτυο χρησιμοποιούνται χρώματα και διαφορετικά μεγέθη για τις κορυφές. Το μέγεθος μιας κορυφής αντιπροσωπεύει το βαθμό της κορυφής (το σύνολο των εισερχόμενων και εξερχόμενων ακμών) στο συνολικό δίκτυο των χρηστών. Το χρώμα μιας κορυφής υποδηλώνει το πλήθος των bitcoin που εισέρχονται ή εξέρχονται προς και από την κορυφή. Όσο πιο έντονο είναι το χρώμα τόσο μεγαλύτερα ποσά ρέουν από και προς την κορυφή.

Χρησιμοποιώντας τα εργαλεία απεικόνισης, το εγωκεντρικό δίκτυο και επιθέσεις όπως είναι αυτή της εύρεσης της φυσικής τοποθεσίας ενός χρήστη (επιθέσεις που δε



Σχήμα 2.9: Η Εγωκεντρική Απεικόνιση μιας Κλοπής

βασίζονται στο bitcoin δίκτυο) μπορούμε να χρησιμοποιήσουμε τις διευθύνσεις των χρηστών που μας ενδιαφέρουν να κάνουμε άρση της ανωνυμίας τους έτσι ώστε να ανακαλύψουμε μεγαλύτερες λεπτομέρειες για αυτούς, λεπτομέρειες για τα ποσά που διαχειρίζονται. Επίσης, μια ακόμα σημαντική πληροφορία που μπορούμε να αντλήσουμε από τέτοιου είδους διαγράμματα είναι η εύρεση του ελάχιστου μονοπατιού μεταξύ δύο χρηστών στο δίκτυο των χρηστών. Παρακάτω δίνεται ένα παράδειγμα για το πως χρησιμοποιούνται όλες αυτές οι τεχνικές.

Παράδειγμα 2.5.1 *Σε αυτό το παράδειγμα θα δούμε την ανάλυση μιας κλοπής* 25000 *bitcoin. Η κλοπή έχει αναφερθεί στο Bitcoin Forum. Όπως φαίνεται στο Σχήμα 2.9 θα ασχοληθούμε κυρίως με πέντε διαφορετικές κορυφές.*

1. *Η κορυφή με το κόκκινο χρώμα είναι η*

   $$p_r = 1KPTdMb6p7H3YCwsyFqrEmKGmsHqe1Q3jg$$

   Η p_r *ανήκει στον κλέφτη.*

2. *Η κορυφή με το μοβ χρώμα είναι η*

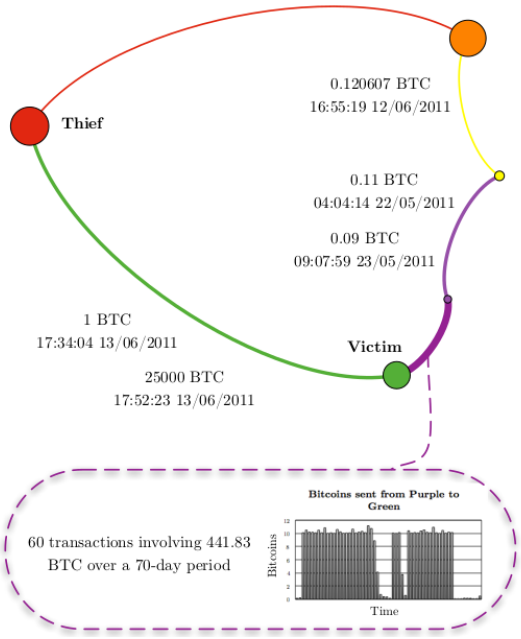
   $$p_o = 15iUDqk6nLmav3B1xUHPQivDpfMruVsu9f$$

   Η p_o *είναι η διεύθυνση με την οποία έκανε mining το θύμα. Περιέχει* 60 *διαφορετικές συναλλαγές (*441.83 *bitcoin σε διάρκεια* 70 *ημερών).*

3. *Η κορυφή με το πράσινο χρώμα είναι η*

   $$p_g = 1J18yk7D353z3gRVcdbS7PV5Q8h5w6oWWG$$

   Η p_g *ανήκει στο θύμα.*



Σχήμα 2.10: Το Υποδίκτυο του Εγωκετρικού Δικτύου της Κλοπής

4. *Η κορυφή με το κίτρινο χρώμα περιέχει πέντε διαφορετικές διευθύνσεις*

$$1MUpbAY7rjWxvLtUwLkARViqSdzypMgVW4$$

$$13tst9ukW294Q7f6zRJr3VmLq6zp1C68EK$$

$$1DcQvXMD87MaYcFZqHzDZyH3sAv8R5hMZe$$

$$1AEW9ToWWwKoLFYSsLkPqDyHeS2feDVsVZ$$

$$1EWASKF9DLUCgEFqfgrNaHzp3q4oEgjTsF$$

5. *Η κορυφή με το πορτοκαλί χρώμα ανήκει σε μια ομάδα hackers τους LulzSec.*

Το θύμα δήλωσε πως ένα μεγάλο ποσό μεταφέρθηκε στη p_r μετά από αλλαγή της διεύθυνσης πληρωμής του p_o. Αυτή είναι μια χαρακτηριστική περίπτωση που ο χρήστης με διεύθυνση p_r επιχειρεί να διατηρήσει την ανωνυμία του.

Δημιουργούμε ένα εγωκεντρικό δίκτυο γύρω από τον κλέφτη χρησιμοποιώντας όλες τις προσβάσιμες από αυτόν κορυφές, όπως είδαμε στη TCP/IP σύνδεση. Αφαιρώντας τους βρόχους και τις πολλαπλές ακμές από μια κορυφή καταλήγουμε στο Σχήμα 2.9. Στη συνέχεια απομονώνοντας τις κορυφές που μας ενδιαφέρουν καταλήγουμε στο Σχήμα 2.10. Η πράσινη ακμή αποτελεί την κλοπή. Το ενδιαφέρον είναι πως οι δύο αυτοί χρήστες συνδέονται και με άλλες ακμές πέραν τις πράσινης, έτσι ο γράφος που καταλήξαμε είναι ένα κύκλωμα. Παρατηρούμε ότι πριν γίνει η κλοπή του μεγάλου αυτού ποσού, είχε γίνει προηγούμενως μια ακόμα κλοπή ενός πολύ μικρότερου ποσού.

Η σύνδεση μέσω της κόκκινης και της πράσινης κορυφής εκτός από την πράσινη ακμή γίνεται μέσω της κίτρινης και πορτοκαλής ακμής. Φαίνεται πως μετά την κλοπή, ο κλέφτης έστειλε ένα ποσό στους hackers (πάνω σε αυτό βασίστηκαν υποψίες πως αυτή η ομάδα έκανε τη κλοπή) ενώ από το mining pool (η κορυφή με το μοβ χρώμα) συνδέεται με την κίτρινη, πιθανόν ο χρήστης με το κίτρινο χρώμα συμμετέχει στην ίδια ομάδα mining με τον πράσινο και φαίνεται πως και αυτός έχει στείλει ένα χρηματικό ποσό στην παραπάνω ομάδα. Η αποστολή αυτή έγινε πριν την κλοπή.

Όπως φαίνεται και με το παραπάνω παράδειγμα μια τέτοια απεικόνιση βοηθάει πολύ στην εξαγωγή συμπερασμάτων και στη συμπλήρωση προτέρων λεπτομερειών στο προφίλ που έχει σχηματιστεί για κάποιον χρήστη. Βέβαια, υπάρχουν και άλλες αναλύσεις τις οποίες θα δούμε παρακάτω.

2.5.2 Άλλες Μορφές Ανάλυσης

Στο προηγούμενο εδάφιο είδαμε πως χρησιμοποιούνται τα εγωκεντρικά δίκτυα για την εύρεση λεπτομερειών που αφορούν το προφίλ που έχει χτιστεί για κάποιον χρήστη. Σε αυτό το υποεδάφιο θα δούμε άλλες αναλύσεις που κάποιες από αυτές χρησιμοποιούν τις παραπάνω αναλύσεις.

- Η ανάλυση που βασίζεται στα ποσά που συναλλάσσεται ένας χρήστης μέσα σε μια χρονική περίοδο. Σε αυτή την ανάλυση, θεωρείται ενδιαφέρουσα μια συναλλαγή αν κάποιος λάβει ένα μεγάλο ποσό, σε σχέση με το πλήθος των bitcoin που έχει ο ίδιος, και μετά από λίγο μετακινήσει το παραπάνω ποσό. Στη προκειμένη ανάλυση χρησιμοποιείται και πάλι η εγωκεντρική ανάλυση δημιουργώντας εγωκεντρικά δίκτυα με την επιπλέων ιδιότητα όμως, πως οι ακμές που μένουν είναι αυτές με μεγάλο ποσό σαν εισερχόμενη συναλλαγή και μετά από μικρό χρονικό διάστημα, το παραπάνω ποσό εμφανίζεται ως εξερχόμενη συναλλαγή.

- Η ανάλυση που βασίζεται στα ρέστα μιας συναλλαγής. Σε αυτή την ανάλυση, έχουμε ότι κάθε συναλλαγή έχει το πολύ δύο εξερχόμενες συναλλαγές. Η μία πηγαίνει στον παραλήπτη, ενώ η άλλη πηγαίνει σε μια διεύθυνση του χρήστη και είναι τα ρέστα της συναλλαγής. Κάτω από προϋποθέσεις, μπορεί κάποιος να μπει και να δει τον πηγαίο κώδικα του client που χρησιμοποιήθηκε και να βρει ποια από τις δύο εξόδους είναι για τα ρέστα. Με αυτή τη διαδικασία μπορούμε να συνδυάσουμε μια διεύθυνση με έναν χρήστη.

- Η ανάλυση των αγορών μεταξύ άλλων νομισμάτων. Σε αυτή την ανάλυση, βασιζόμαστε στην συναλλακτική αξία του bitcoin σε σχέση με κάποιο άλλο νόμισμα. Υπάρχουν υπηρεσίες στις οποίες γίνεται μετατροπή των κρυπτονομισμάτων με μεγάλη ακρίβεια στην συναλλαγματική αξία, πολλές φορές πάνω από 8

δεκαδικά ψηφία. Μια τόσο συγκεκριμένη συναλλαγή θα είναι ευκολότερο να ανιχνευτεί.

- Η ανάλυση της εξαπάτησης. Σε αυτή την ανάλυση, ο επιτιθέμενος έχει μια πιο ενεργή ανάμειξη με το δίκτυο. Μπορεί να εξαπατήσει το θύμα του είτε σημαδεύοντας κρυπτονομίσματα είτε προσφέροντάς του υπηρεσίες που είναι αναγκαίο κάποιος να δώσει προσωπικά στοιχεία, όπως αυτές του ξεπλύματος μαύρου χρήματος.

Με όλα αυτά βλέπουμε ότι το bitcoin δίκτυο έχει πολλές δυσκολίες στο να διατηρήσει την ανωνυμία των χρηστών του. Στο επόμενο κεφάλαιο θα παρουσιαστεί ένας τρόπος για την ισχυροποίηση της ανωνυμίας.

Κεφάλαιο 3

Μοντέλο Δικτύου για Ισχυρή Ανωνυμία σε Bitcoin Δίκτυο

Στο προηγούμενο κεφάλαιο, είδαμε τις ιδιότητες που θα πρέπει να έχει ένα δίκτυο κρυπτονομισμάτων και που το bitcoin δίκτυο αποτυγχάνει. Σε αυτό το κεφάλαιο, θα προσπαθήσουμε να παρουσιάσουμε τη δομή ενός δικτύου όπου οι ιδιότητες της ανωνυμίας και της μικρής καθυστέρησης διάδοσης ενός μηνύματος να είναι ισχυρές. Με λίγο λόγια θα γίνει μια μοντελοποίηση του δικτύου έτσι ώστε να δίνει λύσεις στις πραγματικές ανάγκες του δικτύου.

Ιδανικά, το δίκτυο που θα κατασκευάσουμε θα ακολουθεί ένα πρωτόκολλο το οποίο θα είναι ″ελαφρύ″ και θα εγγυάται την ανωνυμία σε επιθέσεις όπως αυτές που αναφέρθηκαν στο προηγούμενο κεφάλαιο.

Τα αντικείμενα μελέτης του κεφαλαίου θα είναι:

1. **Φράγματα της Ανωνυμίας.** Εδώ θα μελετηθούν οι έννοιες της *Ακρίβειας* και τις *Ανάκλησης*, θα δοθεί ο αυστηρώς μαθηματικός συμβολισμός των ποσοτήτων καθώς και ο αυστηρός ορισμός της έννοιας της ανωνυμίας στο δίκτυο.

2. **Αλγόριθμοι Βελτιστοποίησης.** Εδώ, θα δοθεί το πρωτόκολλο που θα πρέπει να ακολουθεί το δίκτυο καθώς και ο αλγόριθμος για τη διάδοση της πληροφορίας στο δίκτυο.

3. **Υλοποίηση του Προβλήματος.** Εδώ, θα γίνει η κατασκευή του νέου δικτύου που θα ακολουθεί το νέο πρωτόκολλο.

Στο επόμενο εδάφιο θα μελετηθεί το πρώτο αντικείμενο.

3.1 Μετρική της Ανωνυμίας

Στο εδάφιο αυτό, όπως είπαμε, θα μελετηθεί η ανωνυμία. Θα τη θεωρήσουμε σαν μια φραγμένη μετρική και θα προσπαθήσουμε να προσδιορίσουμε το φράγμα της. Επίσης, θα μελετηθούν έννοιες όπως η ακρίβεια και η ανάκληση οι οποίες θα είναι και αυτές μετρικές του δικτύου.

3.1.1 Συμβολισμός

Στο εδάφιο 2.3 είδαμε τη TCP/IP σύνδεση στο P2P bitcoin δίκτυο. Τώρα θα δώσουμε τον συμβολισμό όλων εκείνων των εννοιών τον οποίο και θα χρησιμοποιούμε από εδώ και στο εξής.

Ορισμός 3.1.1 *Ένα P2P δίκτυο είναι ένας κατευθυνόμενος γράφος $G(V, E)$, όπου V είναι το σύνολο των κόμβων του δικτύου και E το σύνολο όλων των συνδέσεων μεταξύ τους.*

Ορισμός 3.1.2 *Σε ένα γράφο $G(V, E)$, για κάθε κόμβο $v \in V$ ορίζουμε το $\Gamma(v)$ να είναι το σύνολο των γειτονικών κόμβων του v στο G. Γενικότερα, για $U \subseteq V$, ορίζουμε το $\Gamma(U)$ να είναι το σύνολο των γειτονικών κόμβων όλων των $v \in U$.*

Άρα με τους παραπάνω ορισμούς έχουμε ότι για κάθε έναν τέτοιο γράφο $G(V, E)$, οι clients (χρήστες) αντιστοιχούν σε κορυφές $v \in V$ με $outdeg(v) \leq 8$ και $indeg(v) = 0$. Από την άλλη πλευρά, οι servers (διακομιστές) είναι οι κόμβοι $v \in V$ με $outdeg(v) \leq 8$ και $indeg(v) \leq 117$.

Στο υποεδάφιο 2.4 είδαμε ότι κάθε κόμβος έχει στη μνήμη του μια λίστα από όλες τις διευθύνσεις που έχουν κάνει σύνδεση στο δίκτυο και ότι όταν κάνει επανεκκίνηση η λίστα αυτή διαγράφεται. Επίσης, είδαμε ότι ένας επιτιθέμενος παίρνει το ρόλο ενός κόμβου για τη προώθηση της πληροφορίας μέσα στο δίκτυο. Άρα, οι κόμβοι θα χωρίζονται σε δύο κατηγορίες.

Ορισμός 3.1.3 *Για κάθε server που δεν ελέγχει ο επιτιθέμενος $v \in V$ ορίζουμε το X_v να είναι το μήνυμα συναλλαγής που έστειλε ο v και με X συμβολίζουμε ένα διάνυσμα που περιέχει τις αντιστοιχίες των $v \mapsto X_v$.*

Στο εδάφιο 2.3 είδαμε πως λειτουργεί το πρωτόκολλο διάδοσης μιας συναλλαγής στο δίκτυο χρησιμοποιώντας τους κόμβους υπεύθυνους για τη διάδοση και πως υπάρχει μια καθυστέρηση στη μεταφορά της πληροφορίας.

Στην περίπτωση μιας επίθεσης κατά την οποία ο επιτιθέμενος προσπαθεί να συσχετίσει διάφορες διευθύνσεις με κάποιον χρήστη, οι λύσεις που έχουν δοθεί έχουν

να κάνουν με την αλλαγή της συμπεριφοράς του χρήστη (καθώς χρησιμοποιούνται και εξωδικτυακοί παράγοντες όπως έχουμε δει). Η λύση του προβλήματος σε αυτό το κεφάλαιο δεν αναγκάζει τον χρήστη να κάνει κάτι τέτοιο.

Ορισμός 3.1.4 *Σε έναν γράφο $G(V,E)$ θα συμβολίζουμε με $V_H \subseteq V$ το σύνολο των κόμβων που ελέγχονται από το δίκτυο (Honest nodes, ενώ με $V_A \subseteq V$ το σύνολο των κόμβων που ελέγχονται από τον επιτιθέμενο (Adversary nodes).*

Αν θεωρήσουμε ότι στο δίκτυο υπάρχουν $|V| = n$ κόμβοι, τότε υπάρχει $p \in \mathbb{Q}$ τέτοιο ώστε $|V_A| = pn \in \mathbb{Z}$ και συνεπώς, καθώς $|V| = |V_A| + |V_H| \Rightarrow |V_H| = n(1-p) = \hat{n}$.

Ορισμός 3.1.5 *Σε έναν γράφο $G(V,E)$, για κάθε $v \in V_H$ ορίζουμε το S_v να είναι ένα σύνολο από τριάδες*

$$(x, u, T_u(x))$$

όπου με x συμβολίζουμε μια συναλλαγή που προωθήθηκε από τον v σε κάποιον επιτιθέμενο $u \in V_A$ τη χρονική στιγμή $T_u(x)$.

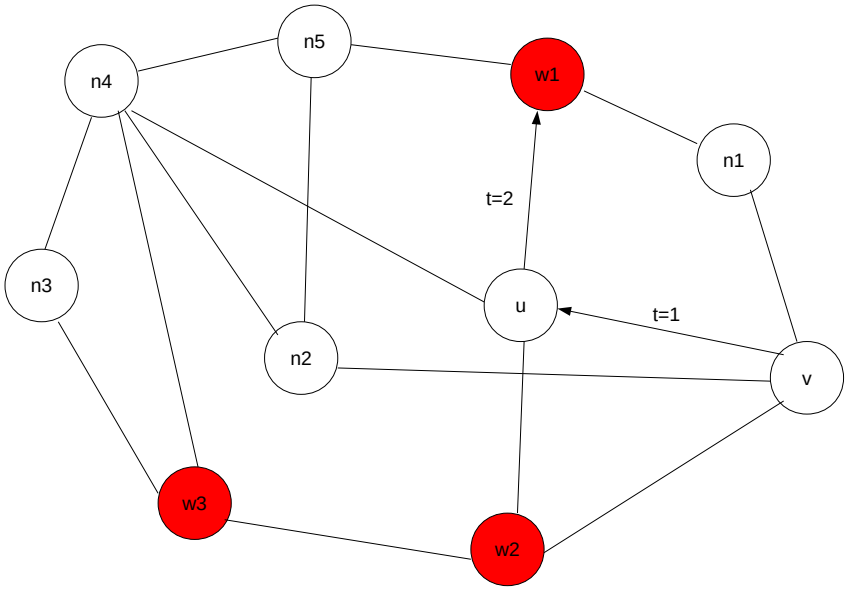
$$S = \{S_v | v \in V_H\}$$

Τέλος, όπως είδαμε στην Υποενότητα 2.4, ο επιτιθέμενος είναι θέμα χρόνου να ανακαλύψει ολόκληρη την τοπολογία του δικτύου. Αυτό, βέβαια, εξαρτάται από το πόσο συχνά αλλάζει η τοπολογία. Αν ο γράφος αλλάζει ανά μικρά χρονικά διαστήματα, τότε ο επιτιθέμενος θα γνωρίζει μόνο τη γειτονιά του $\Gamma(V_A)$. Για ευκολία θα συμβολίζουμε $\Gamma := \Gamma(V_A)$. Για αυτή τη γνώση, ο επιτιθέμενος μαζεύει τα Time Stamps των συναλλαγών και φτιάχνει ένα μονοπάτι ώστε να βρει τη πηγή της συναλλαγής. Η παρατήρηση αυτή συμβολίζεται με $O = (S, \Gamma)$.

Ορισμός 3.1.6 *Σε έναν γράφο $G(V,E)$ με $V = V_H \cup V_A$, για κάθε $v \in V_H$ που ξεκινάει μια συναλλαγή ορίζουμε το $M(X_v)$ να είναι ο server που έχει εμπλακεί στην συναλλαγή X_v και υπάρχει στο μονοπάτι που σχεδιάζει ο επιτιθέμενος για την εύρεση του κόμβου από τον οποίο ξεκίνησε η συναλλαγή.*

Για την καλύτερη κατανόηση της παραπάνω διαδικασίας ακολουθεί ένα παράδειγμα.

Παράδειγμα 3.1.1 *Έστω ένα δίκτυο Peer-2-Peer $G(V,E)$ με $|V| = 10$ και $|V_A| = 3$. Από εδώ συμπεραίνουμε ότι $p = \frac{3}{10} \in \mathbb{Q}$ και $\hat{n} = |V_H| = n(1-p) = 7$. Έτσω τώρα ένας κόμβος $v \in V_H$ στέλνει ένα μήνυμα X_v μες το δίκτυο το οποίο φτάνει σε ένα κόμβο του επιτιθέμενου $w \in V_A$ μέσω ενός κόμβου $u \in V_H$, άρα $M(X_v) = u \in V_H$, τη χρονική στιγμή $T_w(X_v) = 2$. Άρα έχουμε ότι $S_u = \{(X_v, w, 2), ...\}$. Με αυτή την πληροφορία, ο επιτιθέμενος γνωρίζει ότι ο κόμβος που έστειλε τη συναλλαγή αυτή είναι γείτονας του u καθώς χριάστηκε μόλις χρόνο $T_u(X_v) = 1$ για να φτάσει σε αυτόν*



Σχήμα 3.1: Εύρεση του Κόμβου όπου Ξεκίνησε μια Συναλλαγή.

και συνεπώς $S_v = \{(X_v, u, 1), ...\}$. *Έτσι, οι υποψήφιοι έχουν περιοριστεί σε μόλις* $|\Gamma(u) \setminus (\Gamma(u) \cap V_A)| = |\{v, n_4\}| = 2$ *κόμβους (Σχήμα 3.1 οι σκιαγραφημένοι κόμβοι ελέγχονται από τον επιτιθέμενο). Τέλος, κοιτάζοντας πότε έφτασε το μήνυμα στους άλλους κόμβους που διαχειρίζεται, μπορεί να βρει ποιος από τους δύο αυτούς κόμβους που απέμειναν είναι αυτός που έστειλε το μήνυμα.*

Για την επίλυση του προβλήματός μας θα χρησιμοποιήσουμε τους παραπάνω συμβολισμούς καθώς και ορισμένες παραδοχές που θα πρέπει να κάνουμε.

1. Θα θεωρούμε ότι όλοι οι κόμβοι έχουν τη λίστα με όλες τις διευθύνσεις.
2. Θα θεωρούμε ότι όλοι οι κόμβοι του δικτύου χωρίς αυτούς του επιτιθέμενου κάνουν συναλλαγές τη στιγμή που μας ενδιαφέρει να μελετήσουμε.
3. Θα θεωρούμε πολλαπλές συναλλαγές από έναν κόμβο σε μια δεδομένη χρονική στιγμή ως μια συναλλαγή.
4. Θα θεωρούμε ότι ο επιτιθέμενος δε γνωρίζει την ακριβή ώρα που ένας server θα κάνει τη προώθηση μιας συναλλαγής στο δίκτυο.
5. Θα θεωρούμε ότι οι πολιτικές διάδοσης είναι συμμετρικές ως προς τους γείτονες του κόμβου, δηλαδή ο κόμβος που προωθεί την πληροφορία δε θα χρησιμοποιεί τις διευθύνσεις των γειτόνων του για να επηρεαστεί η απόφασή του για το που θα το προωθήσει. Αντιθέτως, η επιλογή του γείτονα θα είναι αμερόληπτη.

6. Θα θεωρούμε πως όλοι οι κόμβοι έχουν μια λίστα από τις ενεργές διευθύνσεις και οι κόμβοι του δικτύου δεν μπορούν να διακρίνουν ποιες από αυτές ανήκουν στον επιτιθέμενο.

Στην επόμενη υποενότητα θα δοθεί η αυστηρή έννοια της ανωνυμίας σε ένα δίκτυο κρυπτονομισμάτων καθώς και ιδιότητες αυτής.

3.1.2 Μετρική

Στην ενότητα αυτή θα αναφερθούμε στη μετρική της ανωνυμίας, δηλαδή το πως μετράμε αν η ανωνυμία στο δίκτυο είναι ισχυρή καθώς θα γίνει και η παρουσίαση του προβλήματος που καλούμαστε να λύσουμε.

Ορισμός 3.1.7 *Με τον όρο μετρική εννοούμε τη μέτρηση της ανωνυμίας στα πλαίσια της προώθησης μιας πληροφορίας στο δίκτυο και ορίζεται να είναι η πιθανότητα*

$$\mathbb{P}_{M,G}(detection) = \frac{\sum_{v \in V_H} \mathbb{P}(M(X_v) = v)}{\hat{n}}$$

Η πιθανότητα αυτή ονομάζεται πιθανότητα ανακάλυψης.

Με την παραπάνω πιθανότητα ελέγχουμε την εκτίμηση ότι έχει βρεθεί ο σωστός κόμβος από τον οποίον έχει ξεκινήσει μια συναλλαγή, υπολογίζοντας όλους τους πιθανούς κόμβους $u \in V_H$ από τους οποίους μπορεί να ξεκίνησε η συναλλαγή X και συνδέοντάς τους με αυτήν, αναγνωρίζοντας το μονοπάτι της διάδοσης της συναλλαγής και τον ίδιο τον γράφο G.

Όπως είπαμε παραπάνω, θα μας χρειαστούν ορισμένες ποσότητες. Αυτές είναι η ακρίβεια και η ανάκληση. Για να τις ορίσουμε όμως αυτές θα χρειαστεί να ορίσουμε πρώτα κάποιες άλλες ποσότητες.

Ορισμός 3.1.8 *Έστω το σύνολο του $\mathbb{Z}_2$, το οποίο αποτελείται από κλάσεις ισοδυναμίας του 0 και του 1. Τώρα, έστω ότι έχουμε n αντικείμενα τα οποία πρέπει να αντιστοιχηθούν με το 0 ή το 1. Ορίζουμε ένα αντικείμενο που απεικονίζεται στη κλάση ισοδυναμίας του 1 ως θετικό, ενώ αν απεικονίζεται στη κλάση ισοδυναμίας του 0 αρνητικό. Χωρίς να γνωρίζουμε την αντιστοιχία αν αντιστοιχίσουμε ένα αντικείμενο της κλάσης 1 στη κλάση 1, αυτό ονομάζεται πραγματικός θετικός. Αντίστοιχα, αν αντιστοιχίσουμε ένα αντικείμενο της κλάσης 0 στη κλάση 0, αυτό ονομάζεται πραγματικός αρνητικός. Στην αντίθετη περίπτωση, αν αντιστοιχίσουμε ένα αντικείμενο της κλάσης 0 στη κλάση 1, τότε αυτό θα ονομάζεται ψεύτικος θετικός. Τέλος, αν αντιστοιχίσουμε ένα αντικείμενο της κλάσης 1 στη κλάση 0, αυτό θα ονομάζεται ψεύτικος αρνητικός.*

Μετά και από τον παραπάνω ορισμό μπορούμε να ορίσουμε πλέον την ακρίβεια και την ανάκληση σε ένα σύστημα.

Ορισμός 3.1.9 *Έστω ένα σύνολο αντικειμένων που ανήκουν σε μια κλάση 0 ή 1. Με τον όρο ακρίβεια εννοούμε την πιθανότητα να διαλέξουμε ένα αντικείμενο για την κλάση 1 και αυτό να είναι σωστό, δηλαδή πραγματικός θετικός.*

$$Precision = \frac{|TruePositives|}{|TruePositives| + |FalsePositives|}$$

Με τον συμβολισμό $|\cdot|$ συμβολίζουμε το πλήθος των στοιχείων του συνόλου.

Ορισμός 3.1.10 *Ορίζουμε ως ανάκληση ή πληρότητα του εκτιμητή την πιθανότητα της ανακάλυψης, δηλαδή σε ένα σύνολο n αντικειμένων κλάσης 1, ανάκληση είναι η πιθανότητα να επιλεχθεί ένα αντικείμενο για την κλάση 1 και όντως να είναι όντως πραγματικός θετικός.*

$$Recall = \frac{|TruePositives|}{|TruePositives| + |FalseNegatives|}$$

Στην περίπτωση που μελετάμε κάθε server είναι μια κλάση και κάθε συναλλαγή είναι το αντικείμενο που πρέπει να αντιστοιχηθεί σε κάποια κλάση. Έτσι, για κάθε server v και απεικόνιση M η ακρίβεια $D_M(v)$ σύγκρισης της κλάσης v με όλες τις άλλες κλάσεις υπολογίζεται ως:

$$D_M(u) = \frac{\mathbb{1}\{M(X_v) = v\}}{\sum_{w \in V_H} \mathbb{1}\{M(X_w) = v\}}$$

ενώ η ανάκληση $R_M(v)$ υπόλογίζεται ως:

$$\mathbb{1}\{M(X_v) = v\}$$

Με $\mathbb{1}$ συμβολίζουμε τη δυαδική συνάρτηση όπου απεικονίζει στοιχεία ενός συνόλου X στο $\mathbb{Z}_2$, δηλαδή

$$\mathbb{1} : X \to \mathbb{Z}_2$$

Στη περίπτωση που έχουμε n σύνολα m αντικειμένων, θα έχουμε n ακρίβειες και ανακλήσεις, μία για κάθε σύνολο. Σε αυτή την περίπτωση, βρίσκουμε την ακρίβεια του συστήματος να είναι ο μέσος όρος από τις ακρίβειες κάθε συνόλου. Το ίδιο ισχύει και για την ανάκληση. Άρα, η ακρίβεια υπολογίζεται να είναι

$$\mathbb{E}(D_M) = \frac{1}{n} \sum_{v \in V_H} \mathbb{E}(D_M(v))$$

Αντίστοιχα, η ανάκληση υπολογίζεται να είναι

$$\mathbb{E}(R_M) = \frac{1}{n} \sum_{v \in V_H} \mathbb{E}(R_M(v))$$

Αξίζει να σημειωθεί πως ανάλογα με την τακτική που ακολουθείται μπορεί να υπάρξει ανάμεσα σε δύο τακτικές ίδια αναμενόμενη μέση ανάκληση (ή σκέτο ανάκληση), αλλά διαφορετική αναμενόμενη μέση ακρίβεια (ή σκέτο ακρίβεια) [3].

Η ανωνυμία που θα δοθεί στο δίκτυο, θα είναι ένας συνδυασμός της αναμενόμενης μέσης ανάκλησης και ακρίβειας. Όταν αυτές οι ποσότητες αυτές είναι μεγάλες, τότε ευνοείται ο επιτιθέμενος. Άρα, για να υπάρξει ένα ισχυρό δίκτυο ως προς την ανωνυμία, θα πρέπει οι ποσότητες αυτές να είναι όσο πιο μικρές γίνεται.

Έστω μια απεικόνιση της στρατηγικής της επίθεσης M και D_M και R_M η μέση ακρίβεια και ανάκληση αντίστοιχα. Η μετρική που μας ενδιαφέρει είναι η ολική αναμενόμενη μέση ακρίβεια $\mathbb{D}_M = \mathbb{E}(D_M)$ και ακρίβεα $\mathbb{R}_M = \mathbb{E}(R_M)$. Η παραπάνω παραδοχή προέρχεται από τέσσερις μεταβλητές:

1. Την εύρεση του γράφου G.
2. Την απεικόνιση μεταξύ servers και μηνυμάτων X.
3. Την παρατήρηση των Time Stamps και της τοπολογικής πληροφορίας O.
4. Τη στρατηγική της απεικόνησης της επίθεσης M.

Όμοια, έστω $D_M(v)$ και $\mathbb{D}_M(v)$ να είναι η ακρίβεια και η αναμενόμενη ακρίβεια ενός $v \in V_H$ και $R_M(v)$ και $\mathbb{R}_M(v)$ η ακρίβεια και η αναμενόμενη ανάκληση του αντίστοιχα. Έστω, τώρα, οι ποσότητες D_{OPT} και R_{OPT} να είναι η ακρίβεια και η ανάκληση αντίστοιχα, που μπορεί να επιτευχθεί ακολουθώντας τη καλύτερη δυνατή τακτική. Αυτές οι δύο μέγιστες τιμές δεν είναι απαραίτητο να συμβούν ταυτόχρονα σε κάποια τακτική.

Το πρόβλημα που καλούμαστε να λύσουμε είναι η δημιουργία και συντήρηση του γράφου - δικτύου και η δημιουργία του πρωτοκόλλου διάδοσης. Όπως είπαμε και παραπάνω, ο στόχος μας είναι να δημιουργήσουμε έναν γράφο και ένα πρωτόκολλο που εκτός από τη ″μικρή καθυστέρηση″, να δίνει και όσο μικρότερη ακρίβεια και ανάκληση γίνεται.

Θα κατασκευάσουμε τώρα έναν βεβαρημένο γράφο, όπου για κάποια σταθερή τοπολογία τ κάθε κόμβος έχει την ίδια πιθανότητα να ονοματιστεί με κάποια αρίθμηση της τοπολογίας τ.

Έστω τώρα $\mathcal{T}$ το σύνολο όλων των τοπολογιών ενός γράφου $G(V, E)$ με $|V| = n$ και Σ το σύνολο όλων των τακτικών διάδοσης που είναι ανεξάρτητες του γράφου. Ο

επιτιθέμενος έχει μόνο τον αλγόριθμο εκτίμησης για την απεικόνιση των συναλλαγών σε κόμβους. Δοθείσας μιας τοπολογίας $\tau \in \mathcal{T}$ και μιας στρατηγικής $\sigma \in \Sigma$ και έστω $\mathcal{M}_{\tau,\sigma}$ το σύνολο των απεικονίσεων των στρατηγικών που απεικονίζουν $\hat{n}$ συναλλαγές σε $\hat{n}$ servers, με ότι γνώση υπάρχει από την τοπολογία και τη στρατηγική.

Ορισμός 3.1.11 *Ορίζουμε ως περιοχή ανίχνευσης για μία τοπολογία $\tau \in \mathcal{T}$ και μια στρατηγική $\sigma \in \Sigma$ να είναι το σύνολο της ακρίβειας και της ανάκλησης που μπορούν να επιτευχθούν*

$$\Omega(\tau,\sigma) = \{(D,R) \quad | \quad \exists M \in \mathcal{M}_{\tau,\sigma}, D = \mathbb{D}_M, R = \mathbb{R}_M\}$$

Ο στόχος μας είναι η περιοχή ανίχνευσης να γίνει όσο το δυνατόν μικρότερη. Συνοψίζοντας, το πρόβλημα είναι το εξής:

Να χαρακτηριστούν τα, ανεξάρτητα του πρωτοκόλλου, άνω φράγματα στην περιοχή ανίχνευσης. Επιπλέον, να βρεθεί το ζεύγος $(\tau^\star, \sigma^\star)$ του οποίου η περιοχή ανίχνευσης είναι υποσύνολο οποιασδήποτε άλλης περιοχής ανίχνευσης κάθε άλλης τοπολογίας γράφου και στρατηγικής, δηλαδή

$$\Omega(\tau^\star,\sigma^\star) = \bigcap_{\sigma\in\Sigma,\tau\in\mathcal{T}} \Omega(\tau,\sigma)$$

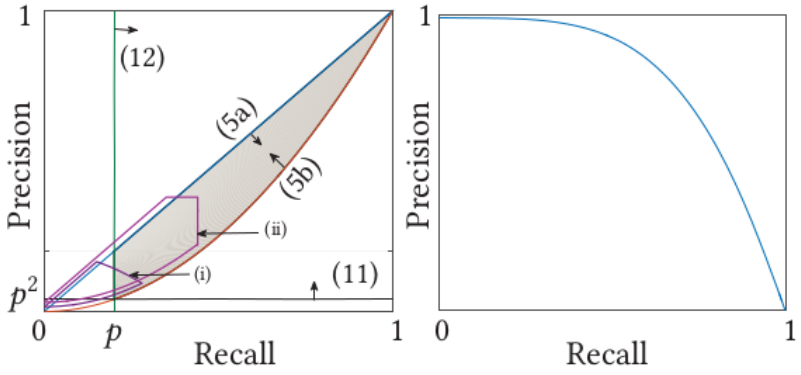
Για την επίλυση αυτού του προβλήματος, θα πρέπει πρώτα να δούμε τις ιδιότητες της μετρικής της ανωνυμίας που θα εκμεταλλευτούμε.

3.1.3 Ιδιότητες Μετρικής

Τα περισσότερα συστήματα προσφέρουν ανωνυμία ανά χρήστη [18–21] και δε χρησιμοποιούν την ακρίβεια και την ανάκληση όπως εδώ. Σε αυτήν την υποενότητα θα προσπαθήσουμε να βρούμε τα φράγματα αυτών των δύο ποσοτήτων. Θα δοθούν θεωρήματα και προτάσεις οι αποδείξεις των οποίων βρίσκονται στη βιβλιογραφία [3]. Θα εξηγηθεί πως χρησιμοποιούνται οι ποσότητες αυτές και θα βρεθούν τα φράγματά τους.

Αρχικά, μπορούμε να δούμε τις καμπύλες της ακρίβειας και της ανάκλησης όπως δίνονται συνήθως εμπειρικά (Σχήμα 3.2 δεξιά) για μια μόνο εκτίμηση.

Εμείς, αντιθέτως, θέλουμε να βρούμε τη περιοχή ανακάλυψης για όλες τις εκτιμήσεις που μπορεί να υπάρξουν καθώς χρησιμοποιούμε τη μέση ακρίβεια και ανάκληση. Συνήθως, με τη μέση ανάκληση, αν αυξηθεί η ανάκληση μιας κλάσης, θα μειωθεί μιας άλλης έτσι είναι δύσκολο να κάνουμε μια υπόθεση για το πως θα μοιάζει αυτή η καμπύλη, ακόμα και αν οι ακραίες τιμές (1,0) και (0,1) μπορούν να επιτευχθούν.



Σχήμα 3.2: Καμπύλες Ακρίβειας - Ανάκλησης

Θεώρημα 3.1.1 *Κάθε απεικόνιση στρατηγικής $M \in \mathcal{M}_{\tau,\sigma}$ σε ένα δίκτυο με τοπολογία $\tau \in \mathcal{T}$ και στρατηγική διάδοσης $\sigma \in \Sigma$ έχει ακρίβεια και ανάκληση των οποίων οι τιμές φράσσονται ως εξής:*

$$\mathbb{D}_M \leq \mathbb{R}_M \leq \sqrt{\mathbb{D}_M}$$

Το συγκεκριμένο θεώρημα μας δίνει μια ιδέα για το τι τιμές μπορούν να πάρουν και που φράσσονται η ακρίβεια και η ανάκληση για κάθε πιθανή εκτίμηση. Έτσι, βγαίνει το συμπέρασμα ότι οι ακραίες τιμές δεν μπορούν να επιτευχθούν και οι μόνες τιμές που μπορούν να επιτευχθούν είναι αυτές ανάμεσα στις κόκκινες και μπλε γραμμές όπως φαίνεται στο Σχήμα 3.2 στα αριστερά (το σχήμα είναι όπως δόθηκε στη βιβλιογραφία [3]).

Από το θεώρημα αυτό, φαίνονται τα άνω φράγματα των ποσοτήτων αυτών. Τώρα, θα προσπαθήσουμε να βρούμε τα κάτω τους φράγματα. Όπως είδαμε και στην Υποενότητα 2.4 η επίθεση στο δίκτυο εξαρτάται από τη γνώση του επιτιθέμενου στη τοπολογία του δικτύου (first-spy). Ο επιτιθέμενος, γνωρίζει πάντα τη τοπολογία της γειτονίας του. Μπορούμε να δείξουμε ότι μια τέτοια επίθεση έχει ακρίβεια τουλάχιστον p^2 και ανάκληση p, όπου p είναι όπως είδαμε το ποσοστό των κόμβων που ελέγχει ο επιτιθέμενος στο σύνολο των κόμβων του δικτύου.

Θεώρημα 3.1.2 *Η μέγιστη ακρίβεια και ανάκληση σε ένα δίκτυο με ποσοστό ελεγχόμενων κόμβων από τον επιτιθέμενο στο σύνολο των κόμβων του δικτύου να είναι p και οποιαδήποτε στρατηγική διάδοσης είναι κάτω φραγμένη ως εξής:*

$$\mathbb{D}_{OPT} \geq p^2$$

$$\mathbb{R}_{OPT} \geq p$$

Τώρα, θα δούμε πως μπορεί μια βελτιστοποιηθεί μια εκδοχή, δηλαδή γνωρίζοντας τα φράγματα μιας περιοχής θέλουμε να βρούμε τη στρατηγική και τη τοπολογία που

θα μας δώσει τη μέγιστη ακρίβεια και ανάκληση αντίστοιχα. Υπενθυμίζουμε ότι δεν είναι αναγκαίο μια εκδοχή να μεγιστοποιεί και τα δύο ταυτόχρονα. Σε αυτή την περίπτωση η καμπύλη θα είναι όπως φαίνεται στο Σχήμα 3.2 αριστερά στο (*i*). Στην περίπτωση που μια εκδοχή μεγιστοποιεί και τα δύο ταυτόχρονα, η καμπύλη θα είναι όπως φαίνεται στο (*ii*), δηλαδή θα καταλήγει σε ένα σημείο.

Θεώρημα 3.1.3 *Η εκδοχή για την οποία παίρνουμε τη μέγιστη ακρίβεια για παρατηρήσεις $O = (S, \Gamma)$, επιτυγχάνεται κάνοντας αντιστοίχηση σε έναν διμερή γράφο $(V_H, \mathcal{X})$. Επίσης, με μια τέτοια αντιστοίχιση προκύπτει ένας βεβαρημένος γράφος με βάρος ακμών $\mathbb{P}(X_v = x|O)$ σε κάθε ακμή $(v, x) \in V_H \times \mathcal{X}$ του γράφου και το βάρος αυτό είναι το μέγιστο δυνατό.*

Πόρισμα 3.1.3.1 *Η μέγιστη αναμενόμενη πληρωμή ενός server u υπό παρακολούθησης $O = (S, \Gamma)$ για τις επιθέσεις, είναι άνω φραγμένη από:*

$$\mathbb{E}(\mathbb{D}_{OPT}(v)|O) \leq \max_{x \in \mathcal{X}} \mathbb{P}(X_v = x|O)$$

Οι πιθανότητες του πορίσματος είναι πολύ δύσκολο να υπολογιστούν καθώς υπάρχουν πολλοί παράμετροι που πρέπει να λάβουμε υπόψιν. Στην περίπτωση όμως που ο επιτιθέμενος μπορέσει να να τις βρει κατά προσέγγιση, τότε μπορεί να βρει τα βάρη του γράφου σε υπολογιστικά πολυωνυμικό χρόνο.

Θεώρημα 3.1.4 *Η μέγιστη ανάκληση για μια εκτίμηση του επιτιθέμενου με παρατήρηση $O = (S, \Gamma)$, είναι μια απεικόνιση που αντιστοιχεί κάθε συναλλαγή $x \in \mathcal{X}$ σε οποιοδήποτε server $v \in arg\max_{v \in V_H} \mathbb{P}(X_v = x|O)$*

Η επίθεση first spy που είδαμε έχει τη μέγιστη δυνατή ανάκληση με την εκδοχή που χρησιμοποιεί. Μια περίπτωση που για μια εκδοχή θα έχουμε μέγιστη ακρίβεια και μέγιστη ανάκληση είναι αυτή για την οποία k servers είναι εξίσου πιθανοί κόμβοι από τους οποίους έχουν ξεκινήσει k συναλλαγές.

Αφού βρήκαμε φράγματα για την ακρίβεια και την ανάκληση καθώς και εκδοχές για τις οποίες έχουμε μέγιστη ακρίβεια και ανάθεση, ήρθε η στιγμή να δούμε τον τρόπο με τον οποίο θα κατασκευαστεί το νέο σύστημα.

3.2 Ισχυροποίηση της Ανωνυμίας

Στην τελευταία αυτή ενότητα, θα παρουσιαστεί και το τελικό αποτέλεσμα της βελτίωσης του δικτύου. Όπως είδαμε και παραπάνω, το δίκτυο του bitcoin είναι ένας στατικός γράφος, δηλαδή δεν αλλάζει μορφή. Η μέθοδος διάδοσης ενός μηνύματος

είναι αυτή της διάχυσης (diffusion), δηλαδή κάθε φορά κάποιος κόμβος καθιστά το πολύ δύο άλλους κόμβους υπεύθυνους για την προώθηση του μηνύματος.

Ένας τέτοιος γράφος, με το συγκεκριμένο πρωτόκολλο, τη συγκεκριμένη τοπολογία και δυναμικότητα είναι εύκολο να αναλυθεί όπως αποδείχθηκε παραπάνω. Έτσι, προκύπτει η ανάγκη της βελτιστοποίησης του παραπάνω δικτύου.

Χρησιμοποιώντας, λοιπόν, τις ιδιότητες που είδαμε στην Υποενότητα 3.1.3 θα κατασκευάσουμε το νέο δίκτυο αλλάζοντας τις εξής τρεις ιδιότητες:

1. Το πρωτόκολλο της διάδοσης μιας πληροφορίας μέσα στο δίκτυο
2. Την τοπολογία του γράφου που θα περιγράφει το δίκτυο.
3. Τη δυναμικότητα του δικτύου.

Με την αλλαγή αυτών των ιδιοτήτων, θα πραγματοποιηθεί το ζητούμενο, δηλαδή η διάδοση ενός μηνύματος μέσα σε ένα μικρό χρονικό διάστημα καθώς και ο περιορισμός της περιοχής ανίχνευσης στο μικρότερο δυνατό επίπεδο με αποτέλεσμα την καλύτερη, στατιστικά, ανωνυμία ως προς τον χρήστη του δικτύου.

Στην υποενότητα που ακολουθεί, θα παρουσιαστούν διάφορα πρωτόκολλα διάδοσης και θα δοθεί το νέο πρωτόκολλο που θα χρησιμοποιηθεί.

3.2.1 Πρωτόκολλο Διάδοσης Μηνυμάτων στο Δίκτυο

Το πρωτόκολλο που που προτείνεται, ονομάζεται **dandelion**. Το συγκεκριμένο πρωτόκολλο, εκμεταλλεύεται ιδιότητες άλλων πρωτοκόλλων για τη διάδοση μιας πληροφορίας. Αρχικά, θα παρουσιάσουμε τα πρωτόκολλα που χρησιμοποιούνται και έπειτα θα παρουσιαστεί το τελικό αποτέλεσμα.

Το Πρωτόκολλο της Πλημμύρας (Flooding)

Το flooding είναι ένα πρωτόκολλο βάση του οποίου ένα μήνυμα διαδίδεται από έναν κόμβο σε όλους τους γειτονικούς του κόμβους με σταθερή καθυστέρηση. Όπως είναι εύκολο να αντιληφθεί κανείς, λόγο του σταθερού ρυθμού και του τρόπου διάδοσης του μηνύματος, είναι εύκολο κάποιος να εντοπίσει τον κόμβο που ξεκίνησε τη διάδοση σε έναν μη κατευθυνόμενο γράφο. Στην περίπτωση όμως του κατευθυνόμενου δεν είναι τόσο προφανές. Θα δείξουμε ότι πράγματι, το πρωτόκολλο αυτό, έχει την ίδια αδυναμία και σε έναν κατευθυνόμενο, στατικό d- κανονικό γράφο.

Θυμίζουμε πως d-κανονικός γράφος ονομάζεται ο γράφους του οποίου όλες οι κορυφές έχουν βαθμό d, δηλαδή σε κάθε κορυφή προσπίπτουν d ακμές.

Πρόταση 3.2.1 *Η αναμενόμενη ακρίβεια του πρωτόκολλου της πλημμύρας σε έναν στατικό, d-κανονικό γράφο είναι τουλάχιστον*

$$D_{OPT} \geq (1 - (1 - p)^d) \geq p$$

Με το παραπάνω βλέπουμε ότι αν ο επιτιθέμενος σε έναν στατικό γράφο γνωρίζοντας τη τοπολογία του και εκμεταλλευόμενος την ιδιότητα του πρωτοκόλλου, ότι δηλαδή ακολουθεί μια διαδικασία κατά την οποία η διάδοση γίνεται προς τους γείτονες του κάθε κόμβου με σταθερό ρυθμό, είναι εύκολο να βρει τον κόμβο που στέλνει το μήνυμα.

Παράδειγμα 3.2.1 *Έστω το δίκτυο του bitcoin το οποίο είναι ένας στατικός γράφος, σχεδόν 16-κανονικός. Αν υποθέσουμε ότι ο επιτιθέμενος διαθέτει 9 από τους συνολικά 90 servers που υπάρχουν εκείνη τη στιγμή στο δίκτυο τότε το* $p = \frac{9}{90} = 0.1$. *Άρα η αναμενόμενη ακρίβεια θα είναι κάτω φραγμένη από το* $1 - (1 - 0.1)^{16} = 1 - 0.9^{16} = 0.814697981$. *Με λίγα λόγια, ο επιτιθέμενος μπορεί να βρει τον κόμβο με ακρίβεια μεγαλύτερη του* 81%.

Η παραπάνω πρόταση, όμως αναφέρεται σε έναν στατικό γράφο. Θα δείξουμε ότι και σε έναν δυναμικό γράφο (ένας γράφος που αλλάζει μορφή ανά κάποια χρονικά διαστήματα), ο επιτιθέμενος μπορεί να έχει μεγάλη ακρίβεια.

Πρόταση 3.2.2 *Η αναμενόμενη ακρίβεια του πρωτόκολλου της πλημμύρας σε έναν δυναμικό, d-κανονικό γράφο είναι κάτω φραγμένη από*

$$D_{OPT} \geq cp$$

για κάποιο σταθερό $c > 0$ *που εξαρτάται από το* p.

Το αποτέλεσμα, λοιπόν, μας δείχνει πως μια τέτοια προσέγγιση δεν είναι ικανή να μας δώσει την ανωνυμία που επιθυμούμε.

Διάχυση Μέσω Άλλου (Diffusion-by-Proxy)

Στην Ενότητα 2.3 αναλύσαμε πλήρως την διαδικασία της διάχυσης που χρησιμοποιεί το bitcoin καθώς και τις αδυναμίες αυτού του πρωτοκόλλου. Στην πραγματικότητα, είναι ένα πρωτόκολλο σαν αυτό της πλημμύρας με τη διαφορά ότι οι χρόνοι διάδοσης από έναν κόμβο σε έναν άλλον είναι τυχαίοι.

Στην περίπτωση της διάχυσης μέσω κάποιου άλλου, γίνεται προσπάθεια για την εξάλειψη της συμμετρικότητας στη μετάδοση. Κάθε κόμβος που θέλει να διαδώσει ένα μήνυμα, βλέπει όλους τους διαθέσιμους κόμβους του δικτύου και επιλέγει

κάποιον τυχαία για τη διάδοση. Με αυτή την προσέγγιση, φαίνεται ότι ο γράφος του δικτύου είναι δυναμικός καθώς αλλάζουν διαρκώς και τυχαία οι βαθμοί των κορυφών.

Μια υπόθεση που θα μπορούσε να κάνει κάποιος είναι ότι η ακρίβεια της επίθεσης θα πρέπει να είναι μικρή από τη στιγμή που ο επιτιθέμενος δε μπορεί να γνωρίζει τη τοπολογία του γράφου. Στη πραγματικότητα κάτι τέτοιο δεν ισχύει, κάτι που φαίνεται στην παρακάτω πρόταση.

Πρόταση 3.2.3 *Η αναμενόμενη ακρίβεια της επίθεσης first-spy στο πρωτόκολλο της διάχυσης μέσω άλλου είναι φραγμένη από*

$$D_{FS} \geq \frac{p}{1-p}(1 - e^{p-1})$$

Στην περίπτωση αυτή, έχουμε ότι καθώς οι κόμβοι θα ενημερώσουν τον επιτιθέμενο για την συναλλαγή που έκανε ο κόμβος που τον ενδιαφέρει με πιθανότητα p, αλλά καθώς και άλλοι κόμβοι θα του μεταφέρουν το ίδιο μήνυμα μέσω της ίδιας ακμής θα μπορεί να βρει το μονοπάτι που θα τον οδηγήσει στον κόμβο που τον ενδιαφέρει.

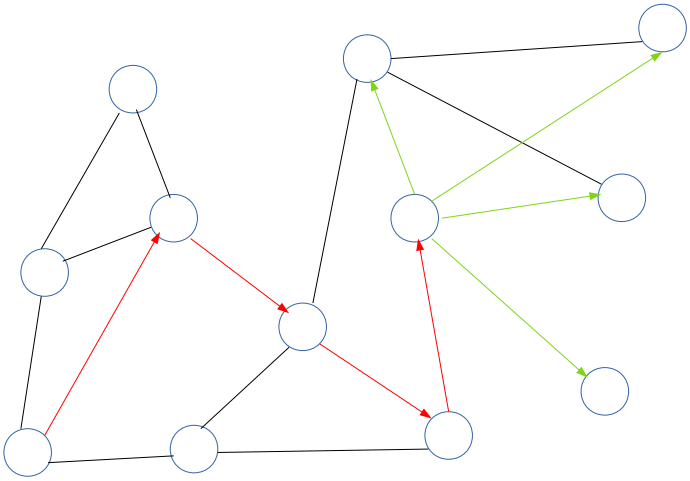
DANDELION

Από τα παραπάνω μπορούμε να βγάλουμε το κύριο συμπέρασμα για το πρωτόκολλο που χρειάζεται να ακολουθεί ένα δίκτυο. Ένα τέτοιο πρωτόκολλο είναι το dandelion που ακολουθεί μια ασύμμετρη μετάδοση σε έναν αραιό γράφο.

Το πρωτόκολλο dandelion αποτελείται από δύο φάσης. Τη φάση της ανωνυμίας (*anonPhase*) και τη φάση της διάδοσης (Σχήμα με κόκκινα βέλη είναι η φάση της ανωνυμίας και με πράσινα η φάση της διάδοσης).

- **Η Φάση της Ανωνυμίας:** Το πρωτόκολλο μεταδίδει το μήνυμα σε μια τυχαία σειρά από ένα τυχαίο αριθμό κόμβων.
- **Η Φάση της Διάδοσης:** Γίνεται η προώθηση του μηνύματος προς όλο το δίκτυο βάση του πρωτοκόλλου της διάχυσης.

Μπορούμε να κατασκευάσουμε δύο διαφορετικούς γράφους για κάθε φάση. Για τη φάση της ανωνυμίας ο γράφος G και για τη φάση της διάδοσης ο γράφος H, ο οποίος είναι και ο γράφος του δικτύου που χρησιμοποιεί το δίκτυο του bitcoin. Καθώς ο γράφος H έχει μικρή καθυστέρηση ως προς τη μετάδωση και όχι τόσο καλή ανωνυμία, προσθέσαμε και ένα ακόμα κομμάτι, αυτό του γράφου G ώστε να



Σχήμα 3.3: Οι Δύο Φάσεις Διάδοσης Μηνύματος

ενδυναμωθεί και αυτή η ιδιότητα. Το τελικό αποτέλεσμα έχει αυξημένο χρόνο διάδοσης αλλά φραγμένο από ένα μικρό άνω φράγμα, πράγμα που θεωρείται αποδεκτό αν υπολογίσει την ανωνυμία που προσφέρεται.

Ο αλγόριθμος (Πίνακας 3.1) αναλύει αυτή τη διαδικασία, δηλαδή τη διαδικασία μετάδοσης ενός μηνύματος X_v από τον κόμβο v χρησιμοποιώντας τον γράφο G για την φάση της ανωνυμίας, με το $\mathcal{N}_{out}$ να συμβολίζει το σύνολο των γειτονικών κόμβων του v όπου ο v μπορεί να στείλει το μήνυμα. Τέλος, χρησιμοποιεί τον γράφο H για τη φάση της διάδοσης του μηνύματος σε ολόκληρο το δίκτυο.

Έχοντας πλέον αυτούς τους δύο γράφους, μπορούμε να μελετήσουμε την ακρίβεια και τη ανάκληση σε αυτούς. Καθώς όμως γνωρίζουμε ότι ο γράφος H είναι ο ίδιος με αυτόν του δικτύου του bitcoin, γνωρίζουμε την ακρίβεια και την ανάκληση για αυτόν. Οπότε, αρκεί να μελετήσουμε αυτές τις ποσότητες για τον γράφο μονοπάτι, της ανωνυμίας.

Μπορεί να παρατηρηθεί ότι στον γράφο G δεν χρειάζονται οι ακριβείς ώρες που προωθείται το μήνυμα (Time Stamps). Έστω τώρα, για κάποιον server v που δεν διαχειρίζεται ο επιτιθέμενος, το σύνολο $S'_u \subseteq S_u$ το οποίο είναι το S_u αφαιρώντας όλες τις συναλλαγές εκτός από αυτές που προέρχονται από μια τριάδα $(x, u, T_u(x))$ η οποία αντιστοιχεί στη πρώτη διάδοση του μηνύματος x σε κάποιον επιτιθέμενο. Με λίγα λόγια, θέλουμε την τριάδα αυτή αν ο u είναι ο πρώτος κόμβος του επιτιθέμενου που του προωθήθηκε το μήνυμα x από έναν κόμβο του δικτύου που δεν διαχειρίζεται ο ίδιος. Επίσης, συμβολίζουμε με $S' = \{S'_v | v \in V_H\}$.

Με τα παραπάνω έχουμε ότι με τι διάδοση αυτή έχουμε κατασκευάσει μια αλυσίδα $X - (S', \Gamma) - S$ συναλλαγών, η οποία είναι Μαρκοβιανή, οπότε σε αυτή την περίπτωση

ΕΙΣΟΔΟΣ: X_v, v, G, H, $q \in (0,1)$
anonPhase $\leftarrow$ *True*
head $\leftarrow v$
resipients $\leftarrow \{v\}$
ΟΣΟ *anonPhase* **ΚΑΝΕ**
    /* Πρόώθηση μηνύματος σε τυχαίο κόμβο */
    targer $\sim Unif(\mathcal{N}_{out}(G, head))$
    recipients $\leftarrow$ *recipients* $\bigcup \{X_u\}$ από το *head* στο *target*
    head $\leftarrow$ *target*
    $u \sim Unif([0,1])$
    ΑΝ $u \leq q$ **ΤΟΤΕ**
        anonPhase $\leftarrow$ *False*
    ΤΕΛΟΣ
ΤΕΛΟΣ
/* Ξεκινάει τη διάχυση στο H από τον κόμβο *head* */
DIFFUSION$(X_v, head, H)$

Πίνακας 3.1: Αλγόριθμος Διάδοσης DANDELION

μας αρκεί να χρησιμοποιήσουμε το S' αντί του S για τον υπολογισμό της πιθανότητας συναλλαγής και καθώς είπαμε ότι δεν χρειάζονται οι χρόνοι, από τις παραπάνω τριάδες μπορούμε να παραλείψουμε τη παράμετρο $T_u(x)$, μια παράμετρος που είναι απαραίτητη στα προηγούμενα πρωτόκολλα που μελετήθηκαν. Άρα καταλήγουμε σε ένα σύνολο που θα συμβολίσουμε και πάλι με S_u για κάποιον server v το οποίο περιέχει τις δυάδες (x, u).

Θεώρημα 3.2.1 *Η αναμενόμενη μέγιστη ανάκληση για το πρωτόκολλο διάδοσης dandelion σε οποιοδήποτε συνεκτικό γράφο n κορυφών με ένα ποσοστό κόμβων p που διαχειρίζεται ο επιτιθέμενος στο συνολικό πλήθος των κόμβων είναι ίσος με* $\mathcal{R}_{OPT} = p + O\left(\frac{1}{n}\right)$.

Το αποτέλεσμα του θεωρήματος αυτού είναι απόρροια του γεγονότος ότι ο γράφος G αναπτύσσεται σε μονοπάτι προς τυχαία κατεύθυνση. Αυτού του είδους η ασυμμετρία είναι που δίνει στην μεταβλητή της ανάκλησης την τιμή p στην καλύτερη περίπτωση.

Τώρα, αυτό που έχει μείνει είναι να μελετήσουμε την ακρίβεια που προσφέρει το πρωτόκολλο αυτό. Για να μελετηθεί η ποσότητα αυτή θα χρειαστεί να μελετηθεί η τοπολογία του δικτύου.

3.2.2 Τοπολογία Δικτύου

Σε αυτή την υποενότητα, θα μελετήσουμε τις τοπολογίες που θα μπορεί να έχει το δίκτυο και ποια από αυτές μας δίνει την καλύτερη αναλογία μεταξύ ανάκλησης και ακρίβειας. Οι τοπολογίες που θα μελετηθούν είναι αυτές των στατικών δένδρων, δυναμικών δένδρων και των δυναμικών μονοπατιών τα οποία και χρησιμοποιούμε καθώς όπως θα δειχτεί, αυτά είναι που μας δίνουν την καλύτερη αναλογία που ψάχνουμε.

Στατικά Δένδρα

Καθώς, όπως αναφέραμε και παραπάνω, αυτό που θέλουμε να καταφέρουμε είναι να μπερδέψουμε το μήνυμα με πολλούς χρήστες. Τα δένδρα είναι ένας πολύ καλός γράφος για την εκθετική προώθηση ενός μηνύματος. Έστω, λοιπόν, ένα κατευθυνόμενο, d-κανονικό δένδρο με ρίζα με την κατεύθυνση της κάθε ακμής να είναι προς τον κόμβο γονέα του.

Με την παραπάνω κατασκευή, στην πρώτη φάση κάθε κόμβος θα προωθεί το μήνυμα προς τον γονέα του, δηλαδή προς τη ρίζα. Ανάλογα με το πόσου κόμβοι θα επιλεγούν για την πρώτη φάση, όσοι κόμβοι βρίσκονται κοντά στη ρίζα, είναι πιο εύκολο να αναμιχθούν τα μηνύματά τους με αυτά των άλλον χρηστών, με αποτέλεσμα ο επιτιθέμενος να μη γνωρίζει ποιο από τα μηνύματα που έλαβε είναι του κόμβου που ενδιαφέρεται. Με την ίδια λογική, για τους κόμβους που βρίσκονται κοντά στα φύλλα, είναι πιο δύσκολο να αναμίξουν τα μηνύματά τους με μηνύματα άλλων χρηστών. Αυτού του είδους η συμμετρία δίνει μια μεγαλύτερη μέση ακρίβεια.

Πρόταση 3.2.4 *Η αναμενόμενη ακρίβεια υπό την μέθοδο της εκτίμησης μέσω σύγκρισης MAT σε οποιοδήποτε δένδρο είναι*

$$D_{MAT} \geq p$$

Η πρόταση μας λέει πως όταν η τοπολογία του γράφου είναι γνωστή (καθώς το δένδρο είναι στατικό) ο επιτιθέμενος μπορεί να χρησιμοποιήσει την τεχνική first spy και να για κάθε κόμβο που χρησιμοποιεί να αθροίζονται οι ακρίβειες. Έτσι για πλήθος $\hat{n}p$ κόμβων που διαχειρίζεται η ακρίβεια θα είναι p.

Δυναμικά Δένδρα

Είδαμε ότι με τα στατικά δένδρα η ακρίβεια γίνεται p, μικρότερη από αυτή που είχε αρχικά το δίκτυο. Μπορούμε όμως να κατεβάσουμε αυτή την ποσότητα και να τη φτάσουμε κοντά στο κάτω φράγμα το οποίο είναι όπως είδαμε p^2.

Το πρόβλημα που είχαμε πριν είναι ότι ο επιτιθέμενος γνώριζε τη τοπολογία του δικτύου και χρησιμοποιούσε κόμβους για την παρατήρηση της κίνησης στο δίκτυο. Αυτό που προσπαθούμε να κάνουμε είναι να μειώσουμε αυτούς τους κόμβους. Μια τέτοια ενέργεια μπορεί να γίνει εφικτή, αν ο επιτιθέμενος δεν γνωρίζει την τοπολογία του δικτύου. Χρησιμοποιώντας, έτσι, δυναμικά δένδρα, θα καταφέρουμε να φτιάξουμε μια τοπολογία με την οποία ο επιτιθέμενος θα έχει γνώση μόνο της τοπολογίας της γειτονιάς του και θα παρατηρήσουμε να έχουμε καλύτερα αποτελέσματα με αυτόν τον τρόπο.

Ορισμός 3.2.1 *Ένα d-αδικό δένδρο ορίζεται να είναι ένα δένδρο με ρίζα που κάθε κορυφή έχει d-παιδιά ή καθόλου παιδιά και κάθε φύλλο βρίσκεται στο ίδιο επίπεδο*

Θεωρούμε ένα d-αδικό κατευθυνόμενο (προς τη ρίζα) δένδρο.

Πρόταση 3.2.5 *Η αναμενόμενη ακρίβεια της first-spy εκτίμησης σε ένα d-αδικό δένδρο, με $d \geq 2$ είναι φραγμένη στο*

$$\mathbb{D}_{FS} \geq \frac{p}{2}$$

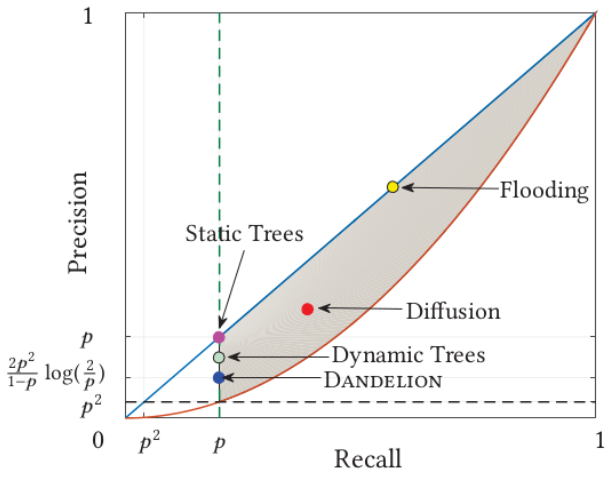
Όπως φαίνεται από τη πρόταση, η ακρίβεια μειώθηκε καθώς ο επιτιθέμενος δεν γνωρίζει τη τοπολογία, αλλά δε μειώθηκε τόσο πολύ γιατί το πρόβλημα με τους κόμβους που βρίσκονται κοντά στα φύλλα παραμένει. Αυτό μας δείχνει ότι τα δένδρα είναι καλύτερη τοπολογία από αυτή που ήδη χρησιμοποιείται, αλλά έχει και αυτή προβλήματα.

Δυναμικοί Γράφοι Μονοπάτια

Τέλος, θα μελετήσουμε του γράφους μονοπάτια ή γραμμές, οι οποίοι είναι στην πραγματικότητα 2-κανονικά δένδρα. Στην περίπτωση αυτή, δεν έχουμε το πρόβλημα με τα φύλλα που είχε δημιουργηθεί στα προηγούμενα δένδρα. Το πρόβλημα εδώ βρίσκεται στη δυσκολία που υπάρχει στο να μπερδευτούν τα μηνύματα από πολλούς χρήστες μεταξύ τους. Αυτό το πρόβλημα όμως ξεπερνιέται χρησιμοποιώντας λίγους κόμβους για τον γράφο αυτό, με αποτέλεσμα ο επιτιθέμενος (από τη στιγμή που ο γράφος είναι δυναμικός) να μη μπορεί να υπολογίσει ποιοι κόμβοι είναι αυτοί που χρησιμοποιούνται και έτσι η ακρίβεια θα πέσει κοντά στο ελάχιστο.

Θεώρημα 3.2.2 *Η αναμενόμενη ακρίβεια του DANDELION σε έναν δυναμικό γράφο γραμμή με n κόμβους από τους οποίους ένα ποσοστό $p < 1/3$ ελέγχεται από τον επιτιθέμενο, το άνω φράγμα της είναι:*

$$D_{OPT} \leq \frac{2p^2}{1-p} \log\left(\frac{2}{p}\right) + O\left(\frac{1}{n}\right)$$



Σχήμα 3.4: Τελικά Αποτελέσματα Πρωτοκόλλων - Διάγραμμα Ακρίβειας Ανάκλησης

Άρα από το θεώρημα καταλαβαίνουμε πως όσο πιο μικρό το p τόσο πιο κοντά στην ελάχιστη τιμή της θα βρίσκεται η ακρίβεια. Επίσης, το θεώρημα όπως είπαμε βασίζεται στο ότι ο επιτιθέμενος δε μπορεί να προβλέψει ποιο κόμβοι επιλέχθηκαν για προώθηση του μηνύματος και θα πρέπει να χρησιμοποιήσει τεχνικές όπως αυτή του first spy.

Οπότε το πρωτόκολλο διάδοσης DANDELION που εφαρμόζεται σε έναν δυναμικό γράφο γραμμή έχει την καλύτερη δυνατή αναλογία ακρίβειας και ανάκλησης (Σχήμα 3.4 [3]).

3.2.3 Κατασκευή του Δικτύου

Είδαμε πως θεωρητικά, ένα τέτοιο δίκτυο έχει μια ισχυρή ανωνυμία. Το ερώτημα όμως είναι μπορεί να κατασκευαστεί και αν ναι, πως μπορεί να υλοποιηθεί και να διατηρηθεί η δυναμικότητα του δικτύου;

Αρχικά, θα προσπαθήσουμε να κατασκευάσουμε τον γράφο γραμμή. Σε έναν τέτοιο γράφο, όλα τα μηνύματα ρέουν από την ίδια ″γραμμή″. Για να κάνουμε κάτι τέτοιο θα χρησιμοποιήσουμε μέρος της στρατηγικής που ακολουθείται στο υπάρχον δίκτυο. Γνωρίζουμε ότι κάθε κόμβος έχει 8 συνδέσεις που στέλνει. Μπορούμε να ζητήσουμε από τους servers να διαθέσουν μια από αυτές σε κάποιον τυχαίο άλλο server. Το πρωτόκολλο που θα ακολουθεί, δηλαδή θα είναι όπως φαίνεται στον αλγόριθμο στον Πίνακα 3.2. Ελέγχει k κόμβους και συνδέεται με αυτόν με τον ελάχιστο βαθμό.

Πρόταση 3.2.6 *Έστω ότι ο αλγόριθμος του Πίνακα 3.2 κατασκευάζει μια γραμμή από k κόμβους από συνολικά n κόμβους. Έστω, τώρα ότι οι κόμβοι που είναι γειτονικοί*

ΕΙΣΟΔΟΣ : Το σύνολο των κόμβων $V = \{v_1, ..., v_n\}$, k
ΕΞΟΔΟΣ : Ένας συνεκτικός γράφος $G(V, E)$ μέσου βαθμού 2
ΓΙΑ $u \leftarrow V$ **ΚΑΝΕ**
    /* Διάλεγει k τυχαίους κόμβους */
    $u_i \sim Unif(V \setminus \{v\})$ για $i \in \{1, ...k\}$
    $u \leftarrow arg_{u_i} deg_{in}(u_i)$
    /* Κάνει τη σύνδεση */
    $E = E \cup (v \rightarrow u)$
ΤΕΛΟΣ
ΕΠΕΣΤΡΕΨΕ $G(V, E)$

Πίνακας 3.2: Αλγόριθμος Επιλογής Μιας Ακμής από k Ακμές

με τα φύλλα του γράφου που κατασκευάζεται από τον αλγόριθμο έχουν βαθμούς $(d_1, ..., d_m)$ *με* $d_1 < ... < d_m$. *Τότε με πιθανότητα* $1 - o(1)$, *ο μέγιστος βαθμός* d_m *ικανοποιεί την συνθήκη:*

$$d_m = \begin{cases} \frac{\log n}{\log \log n}(1 + o(1)) + \Theta(1) & \alpha\nu \quad k = 1 \\ \frac{\log \log n}{\log k}(1 + o(1)) + \Theta(1) & \alpha\nu \quad k > 1 \end{cases}$$

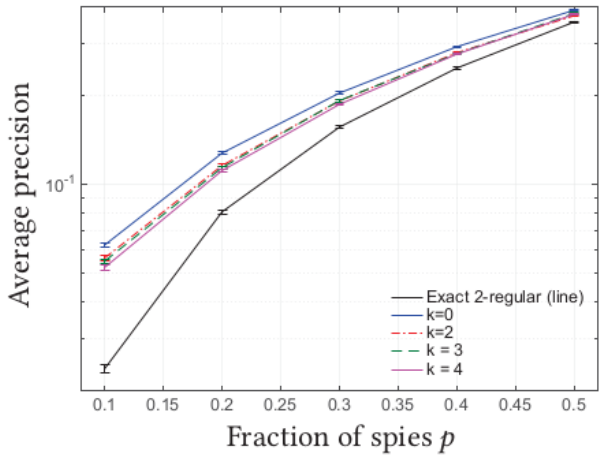
Η πρόταση αυτή μας λέει πως από τη στιγμή που οι βαθμοί των κορυφών προέρχονται από τη κατασκευή, αν μπορέσουμε να κατεβάσουμε το μέγιστο βαθμό στο 2, τότε και ο ελάχιστος θα είναι 2. Με αυτόν τον τρόπο μπορούμε να ελαχιστοποιήσουμε την ακρίβεια συνδέοντας τις κορυφές ενός κανονικού, γράφου με $k = 2$, με κόμβους με τον ελάχιστο βαθμό συνδέσεων εισόδου. Για μεγαλύτερο k, αυξάνεται η κανονικότητα του γράφου με λογαριθμικό ρυθμό.

Όπως είδαμε, τα φύλλα αυξάνουν την ακρίβεια της επίθεσης. Αυτό που έχουμε να κάνουμε λοιπόν είναι να αυξήσουμε το k για να μειωθεί ο αριθμός των φύλλων. Μπορούμε να κάνουμε μια σύγκριση μεταξύ τεχνικών όπως αυτή του first spy και των k-επιλογών ακμών για διάφορες τιμές του k. Αν παρατηρήσουμε στο Σχήμα **;;** [3], όταν πάμε από $k = 1$ σε $k = 2$ γίνεται η πιο ραγδαία μείωση της ακρίβεια. Βέβαια, όσο μεγαλύτερες τιμές παίρνει το k τόσο καλύτερα αποτελέσματα θα παίρνουμε.

Τώρα υπάρχει το εξής πρόβλημα. Ένας επιτιθέμενος μπορεί να μάθει εύκολα τη τοπολογία ενός δύο κανονικού γράφου ή γράφου γραμμή. Αν ο επιτιθέμενος έχει δύο κόμβους μέσα σε έναν τέτοιο γράφο και κάποιος κόμβος ανάμεσα από αυτούς τους δύο επιχειρήσει να κάνει μια συναλλαγή, ο επιτιθέμενος θα καταλάβει πως αυτός ο κόμβος βρίσκεται ανάμεσα στους δικούς του.

Με την λογική αυτή η λύση στο πρόβλημα είναι η εξής. Πρέπει ο γράφος γραμμή να αλλάζει γρήγορα τη μορφή του. Το πόσο γρήγορα θα πρέπει να αλλάζει εξαρτάται από το p, δηλαδή από το ποσοστό των κόμβων του επιτιθέμενου στο δίκτυο.

Παράδειγμα 3.2.2 *Αν υποθέσουμε ότι το ποσοστό των κόμβων που ανήκουν στον*



Σχήμα 3.5: Σύγκριση First-Spy και Προσέγκισης k Ακμών για Διάφορες Τιμές του k

επιτιθέμενο είναι $p = 0.15$. *Πρέπει να σιγουρευτούμε ότι ο γράφος θα αλλάξει μορφή πριν ο επιτιθέμενος μάθει το* 40% *των κόμβων που αποτελούν του γράφο. Θυμίζουμε ότι για κάθε συναλλαγή που γίνεται, η διάδοση γίνεται από διαφορετικό κόμβο. Αν υποθέσουμε ότι όλο το δίκτυο βλέπει 3 συναλλαγές ανά δευτερόλεπτο και έχει περίπου 5.500 servers και καθώς κάθε παρατηρητή που θα έχει ο επιτιθέμενος θα αποτελείται από 7 κόμβους από τους οποίους οι 5 θα του είναι άγνωστοι, τότε θα πρέπει η αλλαγή στη τοπολογία του γράφου να γίνεται κάθε* $5500 \cdot \frac{5}{7} \cdot 0.4 \cdot \frac{1}{3}$ *το οποίο είναι περίπου* 9 *λεπτά.*

Παράρτημα Α΄

Πορτοφόλι

Στο κεφάλαιο αυτό παρουσιάζεται ο κώδικας σε γλώσσα C που γράφτηκε για τη δημιουργία ενός ηλεκτρονικού πορτοφολιού. Το πορτοφόλι αυτό ανήκει στην κατηγορία των cold πορτοφολιών καθώς δεν έχει πρόσβαση στο δίκτυο. Επίσης, είναι γραμμένο για διαφορετικά συστήματα. Για την δημιουργία διευθύνσεως για το δίκτυο του bitcoin, ο χρήστης θα πρέπει να επιλέξει την πρώτη επιλογή που του δίνεται.

Α΄.1 Αρχεία Κεφαλίδες - Header Files

Στο παράρτημα αυτό, δίνονται τα αρχεία κεφαλίδες (header files). Ο κώδικας έχει γραφτεί με τρόπο τέτοιο ώστε κάθε module να είναι ανεξάρτητο. Η κεφαλίδα με το όνομα **myWallet.h** είναι αυτή που χρησιμοποιείται στη δημιουργία του πορτοφολιού.

myWallet.h

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <gmp.h>
#include <string.h>
#include "mathFunc.h"
#include "padding.h"
#include "sha256.h"
#include "ripemd160.h"
#include "walletFunc.h"
```

walletFunc.h

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <gmp.h>
#include <string.h>

#define ui32 unsigned int
#define ui64 unsigned long
#define uc8 unsigned char

unsigned char* base58(mpz_t,ui32);
void F4B(mpz_t, mpz_t);
void con(mpz_t , mpz_t ,mpz_t, ui32);
```

mathFunc.h

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

#define ui32 unsigned int
#define ui64 unsigned long
#define uc8 unsigned char

#define ROTL(x,pos) (((x)<<(pos)) | ((x)>>(32-(pos))))
#define ROTR(x,pos) (((x)>>(pos)) | ((x)<<(32-(pos))))

unsigned int CH(ui32, ui32, ui32);
unsigned int MAJ(ui32, ui32, ui32);
unsigned int BSIG0(ui32);
unsigned int BSIG1(ui32);
unsigned int SSIG0(ui32);
unsigned int SSIG1(ui32);
unsigned int f(ui32, ui32, ui32, ui32);
```

padding.h

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <gmp.h>
#include <string.h>

#define ui32 unsigned int
#define ui64 unsigned long
#define uc8 unsigned char

void charTompz(mpz_t,uc8*);
int msgPad(mpz_t,mpz_t,ui64,ui32);
unsigned int word(mpz_t, ui32, ui32);
unsigned int LEencode(ui32);
```

sha256.h

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <gmp.h>
#include <string.h>
#include "padding.h"
#include "mathFunc.h"

#define ui32 unsigned int
#define ui64 unsigned long
#define uc8 unsigned char

void setw(ui32*,mpz_t,int);
void sha256Dig(mpz_t, uc8*,mpz_t,ui64);
```

ripemd160.h

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <gmp.h>
#include <string.h>
#include "mathFunc.h"
#include "padding.h"

#define ui32 unsigned int
#define ui64 unsigned long
#define uc8 unsigned char

void ripemdDig(mpz_t, uc8*,mpz_t,ui64);
```

Να σημειωθεί πως η επανάληψη στις βιβλιοθήκες γίνεται επειδή κάθε πηγαίος κώδικας που χρησιμοποιεί τις κεφαλίδες μπορεί να λειτουργήσει ξεχωριστά. Τέλος, χρησιμοποιείται το gmp.h για τον χειρισμό μεγάλων αριθμών (μεγαλύτερων των 64-*bit*) που χρησιμοποιούνται είτε στα μηνύματα, είτε στα αποτελέσματα των συναρτήσεων κατακερματισμού.

Α΄.2 Πηγαίοι Κώδικες - Source Files

Σε αυτό το κομμάτι, θα δούμε τους πηγαίους κώδικες (sourse files) του προγράμματος ξεκινώντας από τα αρχεία που περιέχουν εργαλεία τα οποία χρησιμοποιούνται, θα ακολουθήσουν οι συναρτήσεις κατακερματισμού και τέλος το κύριο πρόγραμμα του πορτοφολιού.

Α΄.2.1 Συναρτήσεις Πορτοφολιού - walletFunc.c

Ο πηγαίος κώδικας walletFunc.c περιέχει συναρτήσεις που τις χρειαζόμαστε μόνο για τις λειτουργίες του πορτοφολιού. Τέτοιες συναρτήσεις είναι η σύνθεση δύο αριθμών σε έναν, η επιλογή των τεσσάρων πιο σημαντικών *Byte* ενός αριθμού και η μετατροπή ενός αριθμού στο 58-αδικό σύστημα. Το συγκεκριμένο αριθμητικό σύστημα χρησιμοποιείται για την αποφυγή παρανοήσεων καθώς το δίκτυο δεν επιστρέφει τα χρήματα της συναλλαγής πίσω στον ίδιο τον χρήστη σε περίπτωση λάθους. Έτσι, χρησιμοποιούνται όλοι οι αριθμοί και όλα τα κεφαλαία και πεζά γράμματα του αγγλικού αλφάβητου, ενώ αποφεύγονται χαρακτήρες που μοιάζουν μεταξύ τους όπως το 0 (μηδέν), το *I* (κεφαλαίο *i*), Ο (κεφαλαίο ο) και *l* (κεφαλαίο *L*).

walletFunc.c

```
#include "walletFunc.h"

/*
 * The con function: Concatenates 2 mpz_t into 1
 */

void con(mpz_t hash, mpz_t input1,mpz_t input2, ui32 input2_size){

        mpz_ior(hash,hash,input1);
        mpz_mul_2exp(hash,hash,input2_size);
        mpz_ior(hash,hash,input2);
}

/*
 * The F4B function: Gets the first 4 bytes
 */

void F4B(mpz_t checkSum, mpz_t dataHash){

        mpz_tdiv_q_2exp(checkSum,dataHash,224);
}

/*
 * The base58 function: Converces an mpz_t to a string based on a
 * 58-character base
 */

uc8* base58(mpz_t input,ui32 format){

        int i,j;
        ui64 rem;
        uc8 *address;
```

```
        uc8 *b="123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz";
        uc8 temp;
        mpz_t r,q;

        mpz_init(r);
        mpz_init(q);
        address=calloc(37,sizeof(uc8));
        i=0;
        while(mpz_cmp_ui(input,0)>0){
                rem=mpz_tdiv_qr_ui(input,r,input,58);
                *(address+i)=b[rem];
                i++;
        }

        switch(format){
                case (0): *(address+i)=b[0]; break;
                case (1): *(address+i)=b[2]; break;
                case (2): {     *(address+i)=b[0];
                                        i++;
                                        *(address+i)=b[35];
                                        i++;
                                        *(address+i)=b[34];
                }
        }
        for(j=0;j<i/2;j++){
                temp=*(address+j);
                *(address+j)=*(address+(i-j));
                *(address+(i-j))=temp;
        }

        mpz_clear(r);
        mpz_clear(q);

        return address;

}
```

Α΄.2.2 Μαθηματικές Συναρτήσεις - mathFunc.c

Ο πηγαίος κώδικας mathFunc.c περιέχει συναρτήσεις ολίσθησης των *bit*.

mathFunc.c

```
#include "mathFunc.h"

/*
 * All the mathematic functions the sha-256 and ripemd-160 use
 */

ui32 CH(ui32 word1, ui32 word2, ui32 word3){

        ui32 ch_word;

        ch_word=(((word1)&(word2))^((~(word1))&(word3)));

        return ch_word;
}
```

```
ui32 MAJ(ui32 word1, ui32 word2, ui32 word3){

        ui32 maj_word;

        maj_word=(((word1)&(word2))^((word1)&(word3))^((word2)&(word3)));

        return maj_word;
}

ui32 BSIG0(ui32 word){

        ui32 bsig0;

        bsig0=((ROTR(word,2))^(ROTR(word,13))^(ROTR(word,22)));

        return bsig0;
}

ui32 BSIG1(ui32 word){

        ui32 bsig1;

        bsig1=((ROTR(word,6))^(ROTR(word,11))^(ROTR(word,25)));

        return bsig1;
}

ui32 SSIG0(ui32 word){

        ui32 ssig0;

        ssig0=((ROTR(word,7))^(ROTR(word,18))^((word)>>(3)));

        return ssig0;
}

ui32 SSIG1(ui32 word){

        ui32 ssig1;

        ssig1=((ROTR(word,17))^(ROTR(word,19))^((word)>>(10)));

        return ssig1;
}

ui32 f(ui32 r, ui32 x, ui32 y, ui32 z){

        ui32 temp;

        switch(r){
                case 0: temp=x^y^z;break;
                case 1: temp=(x&y)|((~x)&z); break;
                case 2: temp=(x|(~y))^z; break;
                case 3: temp=(x&z)|(y&(~z)); break;
                case 4: temp=x^(y|(~z));
        }
        return temp;
}
```

Α΄.2.3 Προετοιμασία Λέξης - Padding.c

Ο πηγαίος κώδικας padding.c περιέχει συναρτήσεις όπως αυτή του *padding* για την προετοιμασία λέξεων. Διακρίνονται δύο περιπτώσεις, όπου επιλέγονται από μια μεταβλητή *flag* για τον τρόπο εισαγωγής των τελευταίων 64-*bit* σε Big Endian ή Little Endian μορφή. Έπειτα, ακολουθούν συναρτήσεις που χωρίζουν έναν αριθμό

σε λέξεις μήκους 32-*bit*, καθώς και συνάρτηση που κάνει κωδικοποίηση των λέξεων αυτών σε Little Endian μορφή (ο τρόπος αποθήκευσης των σύγχρονων μηχανημάτων είναι σε Big Endian μορφή και έτσι η μετατροπή είναι απαραίτητη, ωστόσο, αν η αρχιτεκτονική ενός συστήματος είναι Little Endian θα χρειαστεί αλλαγή του κώδικα για να υπάρχουν σωστά αποτελέσματα). Τέλος, υπάρχει και η συνάρτηση που μετατρέπει χαρακτήρες από μορφή *ASCII* σε μεγάλους ακεραίους.

padding.c

```
#include "padding.h"

extern int numBlocks;

/*
 * The charTompz function: ASCII to mpz
 */

void charTompz(mpz_t inmsg,uc8* cmsg)
{
        int block,i;
        ui64 length;

        block=0;
        length=strlen(cmsg);

        while(length > 0)
        {
                length--;
                for (i = 7; i >= 0; --i)
                if(cmsg[block] & (1 << i)) mpz_setbit(inmsg,i+length*8);
                block++;
        }
}

/*
 * This fanction does the padding of the message according to the flag
 * If the flag is 0 the padding of the last 64-bits (the length of the
 * message) is according to the Little Endian architecture
 * If the flag is 1 the padding of the last 64-bits is according to
 * the Big Endian architecture
 * The rest of the padding is according to the Big Endian architecture.
 * To convert the rest of the padding to the Little Endian architecture,
 * you must apply the "LEencode" function on the "word" function.
 */

int msgPad(mpz_t extmsg,mpz_t msg,ui64 msgLength,ui32 flag)
{
        int x,k,blocks,j;
        ui32 msgLength32;
        mpz_t L;

        x=(msgLength+1)%512;
        k=(448+(512-x))%512;

        mpz_ior(extmsg,extmsg,msg);

        mpz_mul_2exp(extmsg,extmsg,1);
        mpz_setbit(extmsg,0);
        mpz_mul_2exp(extmsg,extmsg,(64+k));
        mpz_init2(L, 64);

        if((flag!=0)&&(flag!=1)){
                printf("\n\nIncorrect flag!\n\n");
```

```
                mpz_clear(L);
                return (-1);
        }

/*
 * This part is the difference between LE and BE padding at our case.
 * We only encode the msgLength. The rest of the words will encode
 * by reference separately.
 */
        if(flag==0){
                msgLength32=(msgLength << 32) >> 32;
                msgLength32=LEencode(msgLength32);
                mpz_set_ui(L,msgLength32);
                mpz_mul_2exp(L,L,32);
                mpz_ior(extmsg,extmsg,L);
                msgLength32=(msgLength >> 32);
                msgLength32=LEencode(msgLength32);
                mpz_set_ui(L,msgLength32);
        }
        if(flag==1) mpz_set_ui(L,msgLength);

        mpz_ior(extmsg,extmsg,L);

        blocks=(msgLength+65+k)/512;

        mpz_clear(L);

        return blocks;
}

/*
 * The LEencode function: Big Endian to Little Endian
 * convertion by 32-bit word.
 */

ui32 LEencode(ui32 msg){

        ui32 x,y,le_msg;

        x=((msg & 0x000000FF)<<24)|((msg & 0xFF000000)>>24);
        y=((msg & 0x0000FF00)<<8)|((msg & 0x00FF0000)>>8);
        le_msg=x | y;

        return le_msg;
}

/*
 * The word fanction: Gives the jth 32-word on the ith 512 block
 * on the extend message. (Initial values for i=0 and j=0)
 */

ui32 word(mpz_t msg, ui32 i, ui32 j){



        ui32 w,x,y;
        ui32 bitIndex,loc;
        mpz_t temp;

        mpz_init2(temp,32);
        loc=(numBlocks-i-1)*512+(15-j)*32;
        bitIndex=loc;
        bitIndex=mpz_scan1(msg,bitIndex);
        while((bitIndex<loc+32)&&(bitIndex>=loc)){
                mpz_setbit(temp,bitIndex-loc);
                bitIndex=mpz_scan1(msg,bitIndex+1);
        }
        w=mpz_get_ui(temp);

        mpz_clear(temp);

        return w;
}
```

Α΄.2.4 Συνάρτηση Κατακερματισμού SHA-256 - sha256.c

Ο πηγαίος κώδικας sha256.c χρησιμοποιεί τις μαθηματικές συναρτήσεις (mathFunc.c) καθώς και τις συναρτήσεις για την προετοιμασία των λέξεων (padding.c). Με λίγα λόγια, αν κάποιος θέλει να χρησιμοποιήσει μόνο την $SHA-256$ μπορεί να πάρει αυτή τη τριάδα για να κάνει τη δουλεία του. Χρησιμοποιεί συγκεκριμένες μεταβλητές και σταθερές, καθώς και έναν ιδικό τρόπο επιλογής λέξεων εκτός αυτού στην padding.c. Τέλος, χρησιμοποιεί μια συνάρτηση για να βγάλει το τελικό αποτέλεσμα συνδέοντας όλα τα αποτελέσματα που προκύπτουν από τις παραπάνω συναρτήσεις.

sha256.c

```
#include "sha256.h"

/*
 * Initialization of the values.
 * H[i]: The initial hash values
 * K[i]: The constants
 */

extern int numBlocks;

ui32 K[64]={0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b,
        0x59f111f1, 0x923f82a4, 0xab1c5ed5, 0xd807aa98, 0x12835b01,
        0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7,
        0xc19bf174, 0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc,
        0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da, 0x983e5152,
        0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147,
        0x06ca6351, 0x14292967, 0x27b70a85, 0x2e1b2138, 0x4d2c6dfc,
        0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
        0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819,
        0xd6990624, 0xf40e3585, 0x106aa070, 0x19a4c116, 0x1e376c08,
        0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f,
        0x682e6ff3, 0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
        0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2};
ui32 IV[8]={0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a, 0x510e527f,
        0x9b05688c,0x1f83d9ab,0x5be0cd19};

/*
 * The setw function: Saves the values of the 32-bit words in an array
 */

void setw(ui32 *w,mpz_t msg,int i)
{
        ui32 t;
```

```
        for(t=0;t<16;t++) w[t]=word(msg,i-1,t);
        for(t=16;t<64;t++) w[t]=SSIG1(w[t-2])+w[t-7]+SSIG0(w[t-15])+w[t-16];
}

/*
 * The sha256Dig: Gives the sha-256 digest
 */

void sha256Dig(mpz_t dig,uc8* cmsg, mpz_t msg, ui64 msgLength)
{
        int i,j;
        ui32 *a,*H,*w;
        ui32 T1,T2,t;
        mpz_t extmsg;

        a=calloc(8,sizeof(ui32));
        H=calloc(8,sizeof(ui32));
        w=calloc(64,sizeof(ui32));

        if(msgLength==-1){
                msgLength=8*strlen(cmsg);
                mpz_init2(msg,msgLength);
                charTompz(msg,cmsg);
        }
        mpz_init2(extmsg,msgLength);

        /*
         *  The 1 input in the msgPad is for the Big Endian padding
         */

        numBlocks=msgPad(extmsg,msg,msgLength,1);

        for(j=0;j<8;j++) H[j]=IV[j];
        for(i=1;i<=numBlocks;i++){
                setw(w,extmsg,i);
                for(j=0;j<8;j++) a[j]=H[j];
                for(t=0;t<64;t++){
                        T1=a[7]+BSIG1(a[4])+CH(a[4],a[5],a[6])+K[t]+w[t];
                        T2=BSIG0(a[0])+MAJ(a[0],a[1],a[2]);
                        a[7]=a[6];
                        a[6]=a[5];
                        a[5]=a[4];
                        a[4]=a[3]+T1;
                        a[3]=a[2];
                        a[2]=a[1];
                        a[1]=a[0];
                        a[0]=T1+T2;
                }

                for(j=0;j<8;j++) H[j]+=a[j];
        }
        mpz_set_ui(dig,H[0]);



        for(j=1;j<8;j++){
                mpz_mul_2exp(dig,dig,32);
                mpz_add_ui(dig,dig,H[j]);
        }
        free(a);
        free(w);
        free(H);
        mpz_clear(extmsg);
}
```

Α΄.2.5 Συνάρτηση Κατακερματισμού RIPEMD-160 - ripemd160.c

Ο πηγαίος κώδικας ripemd160.c χρησιμοποιεί τις μαθηματικές συναρτήσεις (mathFunc.c) καθώς και τις συναρτήσεις για την προετοιμασία των λέξεων (padding.c), με διαφορετικό *flag* για να αλλάξει η αρχιτεκτονική σε Little Endian. Με λίγα λόγια, αν κάποιος θέλει να χρησιμοποιήσει μόνο την $RIPEMD-160$ μπορεί να πάρει αυτή τη τριάδα για να κάνει τη δουλεία του. Χρησιμοποιεί συγκεκριμένες μεταβλητές και σταθερές και κάνει δύο παράλληλους υπολογισμούς. Τέλος, χρησιμοποιεί μια συνάρτηση για να βγάλει το τελικό αποτέλεσμα συνδέοντας όλα τα αποτελέσματα που προκύπτουν από τις παραπάνω συναρτήσεις κάνοντας αρχικά μετατροπή του αποτελέσματος από Little Endian σε Big Endian.

ripemd160.c

```
#include "ripemd160.h"

/*
 * Initialization of the values.
 * h[i]: The initial hash values
 * KL[i], KR[i]: The left and right constants respectively
 * wordL[i], wordR[i]: The place on the extend message the
 * left and right 32-bit word is
 * posL[i], posR[i]: The position of the rotation of a variable
 * for its left and right value respectively
 */

extern int numBlocks;

ui32 iv[5]={0x67452301,0xEFCDAB89,0x98BADCFE,0x10325476,0xC3D2E1F0};
ui32 KL[5]={0x00000000,0x5A827999,0x6ED9EBA1,0x8F1BBCDC,0xA953FD4E};
ui32 KR[5]={0x50A28BE6,0x5C4DD124,0x6D703EF3,0x7A6D76E9,0x00000000};

ui32 wordL[80]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
        7, 4, 13, 1, 10, 6, 15, 3, 12, 0, 9, 5, 2, 14, 11, 8,
        3, 10, 14, 4, 9, 15, 8, 1, 2, 7, 0, 6, 13, 11, 5, 12,
        1, 9, 11, 10, 0, 8, 12, 4, 13, 3, 7, 15, 14, 5, 6, 2,
        4, 0, 5, 9, 7, 12, 2, 10, 14, 1, 3, 8, 11, 6, 15, 13};
ui32 wordR[80]={5, 14, 7, 0, 9, 2, 11, 4, 13, 6, 15, 8, 1, 10, 3, 12,
        6, 11, 3, 7, 0, 13, 5, 10, 14, 15, 8, 12, 4, 9, 1, 2,
        15, 5, 1, 3, 7, 14, 6, 9, 11, 8, 12, 2, 10, 0, 4, 13,
        8, 6, 4, 1, 3, 11, 15, 0, 5, 12, 2, 13, 9, 7, 10, 14,
        12, 15, 10, 4, 1, 5, 8, 7, 6, 2, 13, 14, 0, 3, 9, 11};

ui32 posL[80]={11, 14, 15, 12, 5, 8, 7, 9, 11, 13, 14, 15, 6, 7, 9, 8,
        7, 6, 8, 13, 11, 9, 7, 15, 7, 12, 15, 9, 11, 7, 13, 12,
        11, 13, 6, 7, 14, 9, 13, 15, 14, 8, 13, 6, 5, 12, 7, 5,
        11, 12, 14, 15, 14, 15, 9, 8, 9, 14, 5, 6, 8, 6, 5, 12,
        9, 15, 5, 11, 6, 8, 13, 12, 5, 12, 13, 14, 11, 8, 5, 6};
ui32 posR[80]={8, 9, 9, 11, 13, 15, 15, 5, 7, 7, 8, 11, 14, 14, 12, 6,
        9, 13, 15, 7, 12, 8, 9, 11, 7, 7, 12, 7, 6, 15, 13, 11,
        9, 7, 15, 11, 8, 6, 6, 14, 12, 13, 5, 14, 13, 13, 7, 5,
        15, 5, 8, 11, 14, 14, 6, 14, 6, 9, 12, 9, 12, 5, 15, 8,
        8, 5, 12, 9, 12, 5, 14, 6, 8, 13, 6, 5, 15, 13, 11, 11};
```

```
/*
 * The ripemdDig: Gives the ripemd-160 digest
 */

void ripemdDig(mpz_t dig,uc8* c,mpz_t msg,ui64 msgLength){

        ui32 temp, initTemp,T1,T2,i,j,r,rr;
        ui32 *LV,*RV,*w,*h;
        mpz_t extmsg;

        LV=calloc(5,sizeof(ui32));
        RV=calloc(5,sizeof(ui32));
        h=calloc(5,sizeof(ui32));
        w=calloc(16,sizeof(ui32));

        if(msgLength==0){
                msgLength=8*strlen(c);
                mpz_init2(msg,msgLength);
                charTompz(msg,c);
        }
        mpz_init2(extmsg,msgLength);

        /*
         * The 0 input in the msgPad is for the Little Endian padding
         */

        numBlocks=msgPad(extmsg,msg,msgLength,0);

        for(j=0;j<5;j++) h[j]=iv[j];

        for(i=0;i<numBlocks;i++){
                for(j=0;j<16;j++) {
                        w[j]=word(extmsg,i,j);

                        /*
                         * Here is the encoding of the word to LE
                         */

                        w[j]=LEencode(w[j]);
                }
                for(j=0;j<5;j++){
                        LV[j]=h[j];
                        RV[j]=h[j];
                }
                for(j=0;j<80;j++){
                        r=j/16;
                        rr=4-r;
                        initTemp=LV[0]+f(r,LV[1],LV[2],LV[3]);
                        initTemp+=(w[wordL[j]]+KL[r]);
                        temp=ROTL(initTemp,posL[j]);
                        temp+=LV[4];
                        LV[0]=LV[4];
                        LV[4]=LV[3];
                        LV[3]=ROTL(LV[2],10);
                        LV[2]=LV[1];
                        LV[1]=temp;
```

```
                        initTemp=RV[0]+f(rr,RV[1],RV[2],RV[3]);
                        initTemp+=w[wordR[j]]+KR[r];
                        temp=ROTL(initTemp,posR[j]);
                        temp+=RV[4];
                        RV[0]=RV[4];
                        RV[4]=RV[3];
                        RV[3]=ROTL(RV[2],10);
                        RV[2]=RV[1];
                        RV[1]=temp;
                }
                temp=h[1]+LV[2]+RV[3];
                h[1]=h[2]+LV[3]+RV[4];
                h[2]=h[3]+LV[4]+RV[0];
                h[3]=h[4]+LV[0]+RV[1];
                h[4]=h[0]+LV[1]+RV[2];
                h[0]=temp;
        }
                for(j=0;j<5;j++){
                        h[j]=LEencode(h[j]);
                }
        mpz_set_ui(dig,h[0]);
        for(j=1;j<5;j++){
                mpz_mul_2exp(dig,dig,32);
                mpz_add_ui(dig,dig,h[j]);
        }
        free(w);
        free(h);
        free(LV);
        free(RV);
        mpz_clear(extmsg);
}
```

Α΄.2.6 Κύριο Αρχείο - myWallet.c

Ο πηγαίος κώδικας που περιέχει τη βασική συνάρτηση δημιουργίας του πορτοφολιού. Δίνεται ένα τυχαίο δημόσιο κλειδί για τη δημιουργία της διεύθυνσης. Χρησιμοποιεί όλες τις παραπάνω συναρτήσεις και δίνει την επιλογή για τρεις διαφορετικές διευθύνσεις.

myWallet.c

```
#include "myWallet.h"

int numBlocks=0;

int main(){

        mpz_t t,temp,indx,pk,s1,keyHash,indx_keyHash;
        mpz_t s2,dataHash,checkSum,hex_address;
        ui32 format;
        uc8 *base_address;

        printf("\n\nPleas give the wallet format:\n");
        printf("\nFor P2PKH press 0.\n");
        printf("For P2SH press 1.\n");
        printf("For Bech32 press 2.\n\nFormat: ");
        scanf("%u",&format);
        if((format<0) || (format>2)){
                printf("\n\n Incorrect format!\n");
                return 1;
        }
```

```
/*
 * Initialization
 */

mpz_init2(t,32);
mpz_init2(temp,32);
mpz_init2(indx,8);
mpz_init2(pk,264);
mpz_init2(s1,256);
mpz_init2(keyHash,256);
mpz_init2(indx_keyHash,168);
mpz_init2(s2,256);
mpz_init2(dataHash,256);
mpz_init2(checkSum,32);
mpz_init2(hex_address,200);
base_address=calloc(37,sizeof(uc8));

/*
 * Set the value of public key.
 */

mpz_set_ui(pk,0x0250863a);
mpz_mul_2exp(pk,pk,32);
mpz_set_ui(temp,0xd64a87ae);
mpz_ior(pk,pk,temp);
mpz_mul_2exp(pk,pk,32);
mpz_set_ui(temp,0x8a2fe83c);
mpz_ior(pk,pk,temp);
mpz_mul_2exp(pk,pk,32);
mpz_set_ui(temp,0x1af1a840);
mpz_ior(pk,pk,temp);
mpz_mul_2exp(pk,pk,32);
mpz_set_ui(temp,0x3cb53f53);
mpz_ior(pk,pk,temp);
mpz_mul_2exp(pk,pk,32);
mpz_set_ui(temp,0xe486d851);
mpz_ior(pk,pk,temp);
mpz_mul_2exp(pk,pk,32);
mpz_set_ui(temp,0x1dad8a04);
mpz_ior(pk,pk,temp);
mpz_mul_2exp(pk,pk,32);
mpz_set_ui(temp,0x887e5b23);
mpz_ior(pk,pk,temp);
mpz_mul_2exp(pk,pk,8);
mpz_set_ui(temp,0x52);
mpz_ior(pk,pk,temp);

/*
 * Procedure
 */

mpz_set_ui(indx,0x00);
sha256Dig(s1,"",pk,264);
ripemdDig(keyHash,"",s1,256);
```

```
        con(indx_keyHash,indx,keyHash,160);
        sha256Dig(s2,"",indx_keyHash,168);
        sha256Dig(dataHash,"",s2,256);
        F4B(checkSum,dataHash);
        con(hex_address,indx_keyHash,checkSum,32);
        base_address=base58(hex_address,format);

        printf("\n\nYour address is: %s\n\n",base_address);

        /*
         * Free memory
         */

        free(base_address);
        mpz_clear(temp);
        mpz_clear(indx);
        mpz_clear(pk);
        mpz_clear(s1);
        mpz_clear(keyHash);
        mpz_clear(indx_keyHash);
        mpz_clear(s2);
        mpz_clear(dataHash);
        mpz_clear(checkSum);
        mpz_clear(hex_address);

        return 0;
}
```

Βιβλιογραφία

[1] Fergal Reid, Martin Harrigan. An Analysis of Anonymity in the Bitcoin System. In *Security and Privacy in Social Networks*, pages 197–223, 2013.

[2] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. In *bitcoin.org/bitcoin.pdf*, 2008.

[3] Shaileshh Bojja Venkatakrishnan, Giulia Fanti, Pramod Viswanath. Dandelion: Redesigning the Bitcoin Network for Anonymity. In *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, Article No.:22, 2017.

[4] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *IMC '13: Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140, 2013.

[5] Jules DuPont, Anna Cinzia Squicciarini. Toward De-Anonymizing Bitcoin by Mapping Users Location. In *CODASPY '15: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, 2015.

[6] Alex Biryukov, Dmitry Khovratovich, Ivan Pustogarov. Deanonymisation of Clients in Bitcoin P2P Network. In *CCS '14: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29, 2014.

[7] Miraje Gentilal, Paulo Martins, Leonel Sousa. TrustZone-backed bitcoin wallet. In *CS2 '17: Proceedings of the Fourth Workshop on Cryptography and Security in Computing Systems*, pages 25–28, 2017.

[8] Steven Goldfeder, Harry Kalodner, Dillon Reisman, Arvind Narayanan. When the cookie meets the blockchain: Privacy risks of Web payments via cryptocurrencies. In *CoRR, vol. abs/1708.04748*, 2017.

[9] Chris Clark. Bitcoin Internals, *A Technical Guide to Bitcoin* 2013.

[10] Michael Caughey. Bitcoin Step by Step, *The Best Way to Get Started*. 2013.

[11] Sam Patterson. Bitcoin Begginner, *A Step by Step Guide to Buying, Selling and Investing in Bitcoins*. 2013.

[12] Pedro Franco. Understanding Bitcoin, *Cryptography, Engineering and Economics*. *The Wiley Finance Series*, 2014.

[13] Ittay Eyal, Emin Gün Sirer. Majority Is Not Enough: Bitcoin Minning Is Vurnerable. In *COMMUNICATIONS OF THE ACM*, VOL. 61, NO. 07, pages 95–102, 2018.

[14] Hemang Subramanian. Decentralized Blockchain-Based Electronic Marketplaces. In *COMMUNICATIONS OF THE ACM*, VOL. 61, NO. 01, pages 78–84 2004.

[15] Nikolaos Chatzatoglou, Panagiotis Nastou. Digital Signature Construction Methods and Message Authentisity Codes. University of the Aegean, Dep. Mathematics, 2018.

[16] Hans Dobbertin, Antoon Bosselaers, Bart Preneel. RIPEMD-160: A strengthened version of RIPEMD. In *Fast Software Encryption*, pages 71–82, 1996.

[17] National Institute of Standards and Technology. Secure Hash Standard (SHS). In *FIPS 180-4*, 2015.

[18] R.Dingledin, N.Mathewson, P.Syverson. Tor: The second-generation onion router. In *Tecnical Report*. DTIC Document, 2004.

[19] G.Fanti, P.Kairouz, S.Oh, P.Viswanath. Spy vs Spy: Rumor Source Obfuscaton. In *SIGMETRICS Perform. Eval. Rev.*, Vol 43, page 271–284, Issue 1, 2015.

[20] M.J.Freedman, R.Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. CCs. ACM.*, 2002.

[21] Michael K Reiter, Aviel D Rudin. Crowds: Anonymity for web transactions. In *ACM Transactions on Information and System Security (TISSEC)1,1*, pages 66–92, 1998.