Ένα Δυναμικό Μοντέλο Υδατογράφησης Λογισμικού βασισμένο σε Γραφήματα Κλήσεων Συναρτήσεων

Η ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Μηχανικών Η/Υ & Πληροφορικής

Εξεταστική Επιτροπή

από τον

Ιωάννη Χιόνη

ως μέρος των Υποχρεώσεων για τη λήψη του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ

ΣΤΗΝ ΘΕΩΡΙΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Φεβρουάριος 2014

Αφιερωση

Η εργασία αυτή

είναι αφιερωμένη σε όσους βίωσαν το μαγευτικό ταξίδι

που προσφέρει απλόχερα η έρευνα

ακόμη και αν δεν κατάφεραν να φτάσουν στην ανέκαθεν δυσπρόσιτη Ιθάκη τους.

Ευχαριστιες

Ευχαριστώ εκ βάθους καρδίας του γονείς μου Αναστάσιο και Ευαγγελία για την αμέριστη υποστήριξη που μου παρείχαν όλα τα χρόνια των σπουδών μου, χωρίς την οποία δεν θα μπορούσα να είχα επιτύχει και αυτό το στόχο. Ευχαριστώ θερμά τον επιβλέποντα και πάντα Εκεί καθηγητή μου Σταύρο Δ. Νικολόπουλο για την συμπαράσταση, την αδιάπαυστη καθοδήγηση και την άριστη επιστημονική συνεργασία που μου προσέφερε καθ όλη τη διάρκεια εκπόνησης αυτής της Μεταπτυχιακής Εργασίας όπως και για την αμέριστη εμπιστοσύνη που μου επέδειξε στα πρώτα βήματα της ερευνητικής μου πορείας. Παράλληλα, ευχαριστώ πολύ την υποψήφια διδάκτωρ Μαρία Γ. Χρόνη για την συνεργασία και πολύτιμη συνεισφορά της στην προσέγγιση ερευνητικών θεμάτων, καθώς και για την μεθοδική διατύπωση των ερευνητικών μας αποτελεσμάτων σε επιστημονικές εργασίες, ορισμένες από τις οποίες βραβεύτηκαν ως καλύτερες εργασίες Διεθνών Επιστημονικών Συνεδρίων. Τέλος, ευχαριστώ τα μέλη της τριμελούς εξεταστικής επιτροπής κκ. Χρήστο Νομικό και Γιώργο Μανή για την συμμετοχής τους στην επιτροπή καθώς και για τις εύστοχες και πολύτιμες παρατηρήσεις τους που συνέβαλαν ουσιαστικά στη βελτίωση της παρούσας εργασίας.

Περιεχομενα

1 Εισαγωγή **1**

1.1 Στόχος και Δομή Διατριβής 1

1.2 Πνευματικά Δικαιώματα και Προστασία 2

1.3 Τεχνικές Προστασίας Λογισμικού 10

1.3.1 Υδατογράφηση 11

1.3.2 Συσκότιση 13

1.3.3 Θωράκιση 16

1.4 Ιστορική Αναδρομή 18

1.4.1 Σκιαγράφηση Τεχνικών Υδατογράφησης Ψηφιακών Μέσων 19

1.4.2 Εργασίες σε Υδατογράφηση Λογιμσικού 20

1.4.3 Επισκόπιση Υδατογραφημάτων Βασισμένα σε Γράφους 34

2 Υδατογράφηση Λογισμικού **39**

2.1 Υδατογραφήματα 39

2.1.1 Κατηγοριοποίηση 40

2.1.2 Ιδιότητες Υδατογραφημάτων Λογισμικού 42

2.1.3 Χαρακτηριστικά Αξιολόγησης Υδατογραφημάτων Λογισμικού 47

2.2 Αλγοριθμικές Τεχνικές Υδατογράφησης Λογισμικού 49

2.3 Μεθοδολογία Κατασκευής και Ενσωμάτωσης Υδατογραφημάτων 51

2.4 Επιθέσεις σε Υδατογραφήματα Λογισμικού 54

2.5 Γραφήματα Παραγόμενα από Κώδικες Λογισμικών 56

2.6 Ανάλυση Κώδικα Λογισμικού 57

2.7 Ιδανική Κλάση Γραφοθεωρητικών Υδατογραφημάτων 60

2.8 Η Σημασία της Διαφάνειας 61

3 Το Μοντέλο Υδατογράφησης Λογισμικού WaterRPG **65**

3.1 Προγενέστερα Αποτελέσματα 65

3.1.1 Αλγόριθμος Chroni & Nikolopoulos 65

3.1.2 Ανάλυση του Αλγορίθμου Chroni & Nikolopoulos 77

3.1.3 Επιθέσεις στο Αναγώγιμο Μεταθετικό Γράφημα 79

3.1.4 Αρχική Προσέγγιση του Μοντέλου Υδατογράφησης 79

3.2 Βασική Ιδέα του WaterRPG Μοντέλου 84

3.3 Διαδικασίες Ενσωμάτωσης και Εξαγωγής του Υδατογραφήματος 87
3.4 Συνιστώσες του Μοντέλου WaterRPG 90
3.4.1 Δυναμικό Γράφημα Κλήσεων 90
3.4.2 Συνθήκες Ελέγχου 91
3.4.3 Πρότυπα Κλήσεων 92
3.4.4 Κανόνες Εκτέλεσης 94
3.5 Αλγόριθμος Υδατογράφησης Λογισμικού 95
3.5.1 Φάση Ενσωμάτωσης Υδατογραφήματος 96
3.5.2 Φάση Εξαγωγής Υδατογραφήματος 101
3.5.3 Απόδειξη Ορθότητας Αλγορίθμου 102

4 Εφαρμογή και Αξιολόγηση του WaterRPG Μοντέλου **107**
4.1 Πτυχές Εφαρμογής του Μοντέλου WaterRPG 107
4.1.1 Καθολική Εφαρμογή 108
4.1.2 Διαφανής Εφαρμογή 110
4.2 Ανάλυση του Μοντέλου Υδατογράφησης WaterRPG 111
4.2.1 Πραγματικό Κόστος Προτύπου Κλήσεων 113
4.2.2 Μεταβλητές Υδατογράφησης 116
4.3 Αξιολόγηση του Μοντέλου Υδατογράφησης WaterRPG 117
4.3.1 Αξιολόγηση με Βάση Ιδιότητες 117
4.3.2 Πειραματική Αξιολόγηση 119

5 Συγκρίσεις Μοντέλων Υδατογράφησης Λογισμικού **125**
5.1 Συγκρίσεις Ιδιοτήτων 125
5.1.1 Μοντέλο 1 125
5.1.2 Μοντέλο 2 128
5.1.3 Μοντέλο 3 130
5.2 Συγκρίσεις Ανθεκτικότητας 132
5.2.1 Ανθεκτικότητα σε Συσκότιση Κώδικα 133
5.2.2 Ανθεκτικότητα σε Βελτιστοποίηση Κώδικα 135

6 Συμπεράσματα **137**
6.1 Καταληκτικές Παρατηρήσεις 137
6.2 Μελλοντική Εργασία 138

Ευρετηριο Σχηματων

1.1 Κατηγορίες δικαιωμάτων πνευματικής ιδιοκτησίας. 3
1.2 Τομείς που απευθύνονται τα δικαιώματα της πνευματικής ιδιοκτησίας. 5
1.3 Τα τμήματα ενός κινητού τηλεφώνου τα οποία προστατεύονται με δικαιώματα. 9
1.4 Μοντέλο συστήματος υδατογράφησης κώδικα. 12
1.5 Μοντέλο συστήματος συσκότισης κώδικα. 15
1.6 Μοντέλο συστήματος θωράκισης κώδικα. 17
1.7 Γενικό μοντέλο ψηφιακού συστήματος υδατογράφησης. 20
1.8 Τα διαφορετικά στάδια της μεταγλώττισης και της αντίστροφης μηχανικής. . 23
1.9 Γενική περίπτωση ενσωμάτωσης γραφοθεωριτικών υδατογραφημάτων. 35

2.1 Ιδιότητες υδατογραφημάτων λογισμικού. 44
2.2 Υδατογράφημα το οποίο φέρει τη στατική ιδιότητα και συγκεκριμένα ανήκει στα υδατογραφήματα κώδικα. 46
2.3 Μεθοδολογία ανάπτυξης και εφαρμογής μοντέλου υδατογράφησης λογισμικού. 53
2.4 Στάδια από τα οποία μεταβαίνει κάποιος κακόβουλος χρήστης με σκοπό να εντοπίσει και να εξάγει υδατογραφήματα καθώς και να εξαλείψει συσκοτίσεις. 55

3.1 Παράδειγμα πρώτης φάσης κωδικοποίησης. 67
3.2 Παράδειγμα δεύτερης φάσης κωδικοποίησης. 69
3.3 Παράδειγμα τρίτης φάσης αποκωδικοποίησης. 74
3.4 Παράδειγμα τέταρτης φάσης αποκωδικοποίησης. 76
3.5 Φάσεις και εφαρμογές του αλγορίθμου Chroni & Nikolopoulos. 77
3.6 Αρχική προσέγγιση μοντέλου υδατογράφησης λογισμικού WaterRPG. . . . 81
3.7 Βασική ιδέα ενσωμάτωσης του υδατογραφήματος στον κώδικα σύμφωνα με το WaterRPG μοντέλο. 85
3.8 Αντικατάσταση κλήσης συνάρτησης με αλληλουχία κλήσεων συναρτήσεων. . 86
3.9 Διαδικασία υδατογράφησης λογισμικού με το WaterRPG μοντέλο. 88
3.10 Πρότυπα κλήσεων συναρτήσεων. 93
3.11 Συνδυασμός συνθήκης ελέγχου, πρότυπο κλήσης και κανόνων εκτέλεσης. . . 95
3.12 Ενσωμάτωση υδατογραφήματος. 96
3.13 Πίνακας ο οποίος εμπεριέχει όλες τις κλήσεις συναρτήσεων που εκτελούνται κατά τη διάρκεια εκτέλεσης. 99
3.14 Πιθανές ροές εκτέλεσης. 100
3.15 Εξαγωγή υδατογραφήματος. 101

3.16 Ελάχιστο μονοπάτι πάνω στο δυναμικό γράφημα κλήσεων. 105
3.17 Αλληλουχία κλήσεων συναρτήσεων στο P και στο P^*. 106

4.1 Πτυχές εφαρμογής του δυναμικού μοντέλου WaterRPG. 108
4.2 Η χειρότερη και η καλύτερη κατάσταση υδατογράφησης με βάση τα δυναμικά γραφήματα κλήσεων συναρτήσεων. 112
4.3 Τμήματα που κώδικα που δημιουργούνται στις υδατογραφημένες συναρτήσεις. 113
4.4 Ενέργειες που επιτελούνται στην στοίβα κλήσεων για μία κλήση συνάρτησης. 114
4.5 Αντικατάσταση κλήσης με κύριο σώμα κληθείσας συνάρτησης. 123

5.1 Γραφοθεωριτικό υδατογράφημα. 126
5.2 Υδατογράφημα δυναμικού μονοπατιού. 129

Ευρετηριο Πινακων

4.1 Πλήθος Κλήσεων σε P και σε P^* 120
4.2 Χρόνος Εκτέλεσης (msec) 120
4.3 Χώρος Μόνιμης Αποθήκευσης (Kb) 120
4.4 Χώρος Αποθήκευσης Σωρού (Mb) 120
4.5 Bytecode Σύνολα Εντολών 121
4.6 Ενδεικτικές Εντολές Bytecode 122

5.1 Ανθεκτικότητα σε τεχνικές συσκότισης των μοντέλων 1, 2 και 3 134
5.2 Ανθεκτικότητα σε τεχνικές βελτιστοποίησης των μοντέλων 1, 2 και 3 136

Ευρετηριο Αλγοριθμων

1 Πρώτη φάση κωδικοποίησης *Encode_W_to_SiP* 66
2 Δεύτερη φάση κωδικοποίησης *Encode_SiP_to_RPG* 68
3 Τρίτη φάση αποκωδικοποίησης *Decode_RPG_to_SiP* 70
4 Τέταρτη φάση αποκωδικοποίησης *Decode_SiP_to_W* 75
5 Πρώτη φάση ενσωμάτωσης *Encode_RPG_to_code* 97
6 Δεύτερη φάση εξαγωγής *Decode_code_to_RPG* 102

Περιληψη

Ιωάννης Χιόνης
Τμήμα Μηχανικών Η/Υ και Πληροφορικής,
Σχολή Θετικών Επιστημών, Πανεπιστήμιο Ιωαννίνων, Ελλάδα
Φεβρουάριος 2014
Ενα Δυναμικό Μοντέλο Υδατογράφησης Λογισμικού βασισμένο σε
Γραφήματα Κλήσεων Συναρτήσεων
Επιβλέπον: Καθηγητής Σταύρος Δ. Νικολόπουλος

Στην παρούσα μεταπτυχιακή εργασία παρουσιάζουμε αρχικά τις βασικές έννοιες σχετικά με την πνευματική ιδιοκτησία και τα πνευματικά δικαιώματα, καθώς επίσης αναλύουμε λεπτομερώς τα προαπαιτούμενα θέτοντας τους θεμελιώδεις κανόνες αναφορικά με την προστασία λογισμικού και συγκεκριμένα με την υδατογράφηση. Εν συνεχεία, προτείνουμε μία νέα ιδέα ενσωμάτωσης και εξαγωγής αναγώγιμων μεταθετικών γραφημάτων μέσα σε κώδικες λογισμικού η οποία χειρίζεται μόνο πραγματικό κώδικα τροποποιώντας τα διαγράμματα ελέγχου ροής αυτού. Την ιδέα αυτή, την κωδικοποιούμε αναλυτικά και πλήρως σε ένα μοντέλο υδατογράφησης λογισμικού, που ονομάζουμε WaterRPG, παραθέτοντας τόσο τους αλγορίθμους κωδικοποίησης της δομής του υδατογραφήματος, όσο και τους αλγορίθμους εφαρμογής του υδατογραφήματος στο λογισμικό. Στη συνέχεια, παρουσιάζουμε λεπτομέρειες, ως επί το πλείστον προγραμματιστικές, οι οποίες αφορούν την ενσωμάτωση πληροφορίας σε κώδικα, ενώ ταυτόχρονα δίνουμε αποτελέσματα της εφαρμογής του αλγορίθμου σε λογισμικό. Τέλος πραγματοποιούμε συγκρίσεις του μοντέλου WaterRPG με προηγούμενα ήδη δημοσιευμένα σε επιστημονικές εργασίες μοντέλα υδατογράφησης λογισμικού εξάγοντας σημαντικές πληροφορίες σχετικές με την ευρωστία της δομής και την αποτελεσματικότητα που αυτή φέρει με την ενσωμάτωσή της.

Extended Abstract in English

Ioannis Chionis
Department of Computer Science and Engineering,
School of Science, University of Ioannina, Greece
February 2014
A dynamic Software Watermarking Model based on
Function Call Graphs
Supervisor: Professor Stavros D. Nikolopoulos

This master thesis initially presents the basic principles related to intellectual property and copyrights, as well as sets and analyzes in details the prerequisites about software protection and more precisely about the software watermarking. The purpose of this thesis is to propose an innovative idea for embedding and extracting reducible permutation graphs into software only by manipulating the host code modifying its control flow. This idea is encoded in a model called WaterRPG quoting both algorithms about the construction of the watermark and about the implementation process to the code, including the implementation correctness . Then, details mostly pure programming are presented, which are related with the addition of hidden information within the code, while giving results of the implementation of the WaterRPG model on real application programs. Finally, the model presented in this thesis is compared with previous and already published in scientific papers models for sofwtare watermarking exporting meaningful inferences about the robustness and the effectiveness of the proposed model.

Κεφαλαιο 1

Εισαγωγη

1.1 Στόχος και Δομή Διατριβής

1.2 Πνευματικά Δικαιώματα και Προστασία

1.3 Τεχνικές Προστασίας Λογισμικού

1.4 Ιστορική Αναδρομή

1.1 Στόχος και Δομή Διατριβής

Η παρούσα διατριβή επικεντρώνεται κυρίως στην κατασκευή, ενσωμάτωση σε κώδικα και αξιολόγηση υδατογραφημάτων λογισμικού. Συγκεκριμένα, η διατριβή αυτή αφού σκιαγραφήσει το γενικό κλίμα που επικρατεί στην προστασία των πνευματικών δικαιωμάτων και παρουσιάσει ως ιστορική αναδρομή ένα αντιπροσωπευτικό σύνολο τεχνικών υδατογράφησης λογισμικού, στηρίζεται σε αποτελέσματα εργασιών οι οποίες είχαν δημοσιευτεί σε διεθνή επιστημονικά συνέδρια έχοντας ως κύριο στόχο να συνεισφέρει στον τομέα υδατογράφησης λογισμικού παρουσιάζοντας λεπτομερώς και αναλύοντας εις βάθος ένα ρηξικέλευθο δυναμικό μοντέλο υδατογράφησης. Το μοντέλο αυτό, ονόματι WaterRpg, εκ κατασκευής μπορεί και αντιμετωπίζει σημαντικές δυσκολίες που υπήρχαν στην ενσωμάτωση και εξαγωγή γραφημάτων σε κώδικες λογισμικών, είτε από την πλευρά των προγραμματιστών οι οποίοι προσπαθούν να θέσουν την προστασία (ενσωμάτωση), είτε από την πλευρά των κακόβουλων χρηστών οι οποίοι προσπαθούν να αναιρέσουν την προστασία (εξαγωγή).

Η διατριβή περιέχει 6 κεφάλαια στο σύνολό της. Εν αρχή στο Κεφάλαιο 1 ο αναγνώστης εισάγεται στα πνευματικά δικαιώματα καθώς επίσης και στις τεχνικές που έχουν αναπτυχθεί για την προστασίας τους. Στο Κεφάλαιο 2 ο αναγνώστης επικεντρώνεται στην τεχνική της υδατογράφησης πάνω στην οποία μαθαίνει βασικές έννοιες, ορισμούς, τεχνικές και αλγορίθμους, ενώ παράλληλα έρχεται αντιμέτωπος με δύσκολα και άλυτα προβλήματα. Στο Κεφάλαιο 3 παρουσιάζεται στον αναγνώστη το δυναμικό μοντέλο υδατογράφησης

WaterRpg το οποίο είναι βασισμένο σε μία συγκεκριμένη κατηγορία γραφημάτων, ενώ στο Κεφάλαια 4 το μοντέλο αυτό αναλύεται διεξοδικά και αξιολογείται. Τέλος, στα Κεφάλαια 5 και 6 παρουσιάζονται στον αναγνώστη συγκρίσεις του προτασσόμενου μοντέλου της εν λόγω διατριβής με άλλα ιδίας αλλά και διαφορετικής κατηγορίας μοντέλα καθώς και εμπεριστατωμένα συμπεράσματα τα οποία προήλθαν ύστερα από ενασχόληση του συγγραφέα στον τομέα των πνευματικών δικαιωμάτων και της προστασίας λογισμικού.

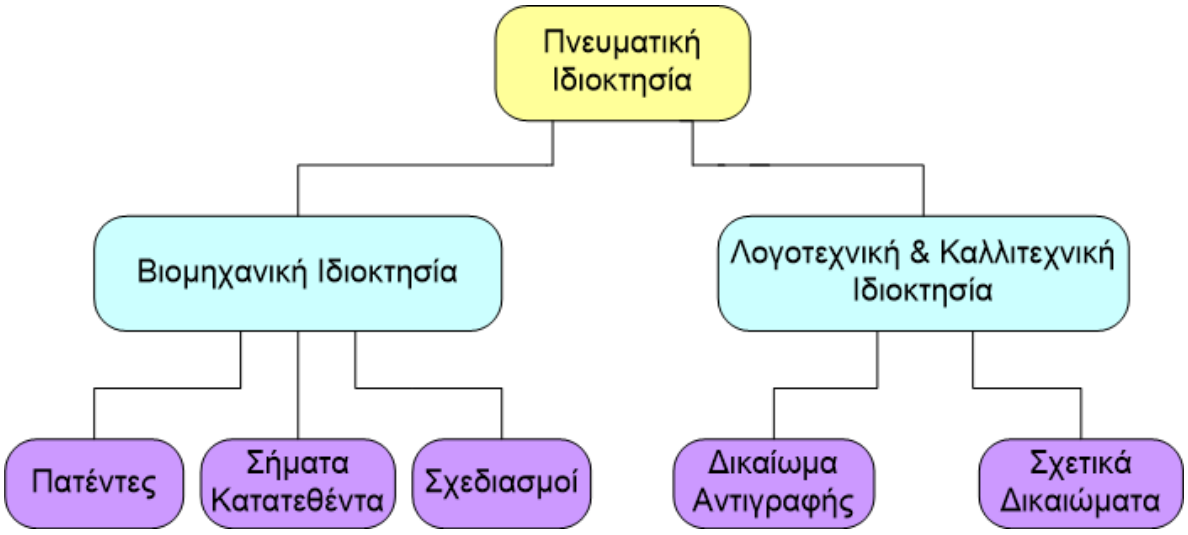
1.2 Πνευματικά Δικαιώματα και Προστασία

Πνευματικό δικαίωμα ή πνευματική ιδιοκτησία αποκτά ο δημιουργός πάνω στο πρωτότυπο έργο του, το οποίο ενδεχομένως περιέχει πνευματικό δημιούργημα λόγου, επιστήμης ή τέχνης που εκφράζεται με οποιαδήποτε μορφή. Ιδίως τα γραπτά, ή προφορικά κείμενα, τα λογισμικά ηλεκτρονικών υπολογιστών, οι μουσικές συνθέσεις, με κείμενο ή χωρίς, τα θεατρικά έργα, με μουσική ή χωρίς, οι χορογραφίες και οι παντομίμες, τα οπτικοακουστικά έργα, τα έργα των εικαστικών τεχνών, στα οποία περιλαμβάνονται τα σχέδια, τα έργα ζωγραφικής και γλυπτικής, τα χαρακτικά έργα και οι λιθογραφίες, τα αρχιτεκτονικά έργα, οι φωτογραφίες, τα έργα των εφαρμοσμένων τεχνών, οι εικονογραφήσεις, οι χάρτες, τα τρισδιάστατα έργα που αναφέρονται στη γεωγραφία, την τοπογραφία, την αρχιτεκτονική ή την επιστήμη αποτελούν κατηγορίες έργων οι οποίες μπορούν να προστατευτούν από τέτοιου είδους δικαιώματα [1].

Συγκεκριμένα η δημιουργία του μορφώματος της πνευματικής ιδιοκτησίας και η νομική της προστασία είναι σχετικά πρόσφατες και ανάγονται περί τον 18ο αιώνα. Πρόδρομος της προστασίας ήταν η απονομή προνομίων σε τυπογράφους, αρχικά στην πόλη της Ιταλίας τη Βενετία από το έτος του 1469 και αργότερα και σε άλλα ευρωπαϊκά κράτη. Τα προνόμια αυτά όμως, αφορούσαν την προστασία του τυπογράφου από ανατυπώσεις φιλολογικών και μουσικών έργων από τρίτους και όχι την προστασία του δημιουργού, ενώ σύντομα κατέληξαν να γίνουν όργανο της λογοκρισίας εκ μέρους των ηγεμόνων στα εκδιδόμενα έργα. Ο πρώτος νόμος που αναγνώρισε δικαίωμα στον δημιουργό ήταν ο αγγλικός "Act for the Encouragement of Learning, by vesting the Copies of Printed Books in the Authors or purchasers of such Copies, during the Times therein mentioned" του έτους 1709 επί βασιλείας της Αννας. Ο νόμος αυτός απένεμε αποκλειστικό δικαίωμα στον δημιουργό επί του έργου του διάρκειας 14 ετών από τη δημοσίευση του έργου με δυνατότητα παράτασης για άλλα τόσα, εφ όσον αυτός ήταν ακόμη εν ζωή.

Στην Ελλάδα η προστασία της πνευματικής ιδιοκτησίας άργησε να κάνει την εμφάνισή της καθώς τα χρόνια εκείνα αποτελούσαν μία μεταπολεμική εποχή για εκείνη. Συγκεκριμένα ο ποινικός νόμος του έτους 1835 ήταν ο πρώτος νόμος ο οποίος προέβλεπε στο άρθρο του υπ αριθμόν 432 ιδιαίτερο ποινικό αδίκημα την καταπάτηση των πνευματικών δικαιωμάτων σε πολλά για την τότε εποχή αντικείμενα. Παράλληλα ο ίδιος νόμος στο άρθρο του υπ αριθμόν 371, προέβλεπε και το αδίκημα της κλοπής. Όριζε δε ως κλοπή, την αφαίρεση ξένων κινητών πραγμάτων από το προστατευόμενο αντικείμενο. Παρ όλα αυτά οι διατάξεις αυτές απείχαν σε αρκετά μεγάλο βαθμό από το να παρέχουν μια τρόπον τινά ολοκληρωμένη

και αποτελεσματική προστασία στους δημιουργούς. Ο πρώτος ολοκληρωμένος νόμος περί πνευματικής ιδιοκτησίας ήταν ο 2387/1920. Ο νόμος αυτός προέβλεπε την προστασία των έργων όσο ζούσε ο δημιουργός και για 50 χρόνια ύστερα από τον θάνατό του. Προκειμένου όμως ο νόμος αυτός να ανταποκρίνεται σε νεώτερες ανάγκες, τροποποιήθηκε κατ επανάληψιν και αντικαταστάθηκε τελικά πλήρως από τον νόμο 2121/1993, ο οποίος έφερε αρκετές ανανεώσεις και καινοτομίες στο σύστημα προστασίας. Στον ανανεωμένο νόμο η προστασία παρέμεινε επιπλέον 50 χρόνια μετά τον θάνατο του δημιουργού, για να παραταθεί περαιτέρω 20 χρόνια το έτος 1997 και να φτάσει αισίως στα 70. Παράλληλα από τις αρχές της δεκαετίας του 90 έχει αρχίσει η εναρμόνιση στο πλαίσιο της Ευρωπαϊκής Κοινότητας του δικαίου της πνευματικής ιδιοκτησίας με την έκδοση αλλεπάλληλων οδηγιών, οι οποίες ρυθμίζουν κατά ενιαίο τρόπο για όλα τα κράτη - μέλη αρκετές πτυχές της πνευματικής ιδιοκτησίας.



Σχήμα 1.1: Κατηγορίες δικαιωμάτων πνευματικής ιδιοκτησίας.

Ακόμα και στην σημερινή εποχή όμως, την γνωστή ως εποχή της τεχνολογίας και του διαδικτύου όπου κατά κόρον γίνεται χρήση ηλεκτρονικών προγραμμάτων, στην Ελλάδα αλλά και γενικά ανά τον κόσμο δεν έχει θεσπιστεί όπως θα έπρεπε η προστασία των πνευματικών δικαιωμάτων σε λογισμικά ηλεκτρονικών υπολογιστών, ή τουλάχιστον αυτά που έχουν θεσπιστεί δεν εφαρμόζονται σε ικανοποιητικό επίπεδο, αφήνοντας σοβαρά κενά τα οποία και τα εκμεταλλεύονται καθημερινά χιλιάδες χρηστών. Το γεγονός αυτό μπορεί με ένα πρώτο άκουσμα να φαντάζει αμελητέο ή κάτι το οποίο είναι χαμηλότερης προτεραιότητας, αλλά όπως θα αποδειχθεί στη συνέχεια είναι ζωτικής σημασίας πρώτα για τους δημιουργούς, είτε αυτοί είναι μεμονωμένα άτομα, είτε αυτοί είναι ολόκληρες εταιρίες και ύστερα για τα κράτη τα οποία χάνουν τεράστια ποσά χρημάτων κάθε χρόνο από τους φόρους όπου υπό φυσιολογικές συνθήκες θα εισέπρατταν από τις πωλήσεις τους.

Τα λογισμικά, αυτή τη στιγμή προστατεύονται με βάση τις διατάξεις περί πνευματικής ιδιοκτησίας. Η Ευρωπαϊκή Οδηγία 91/250 “Για την νομική προστασία των προγραμμάτων

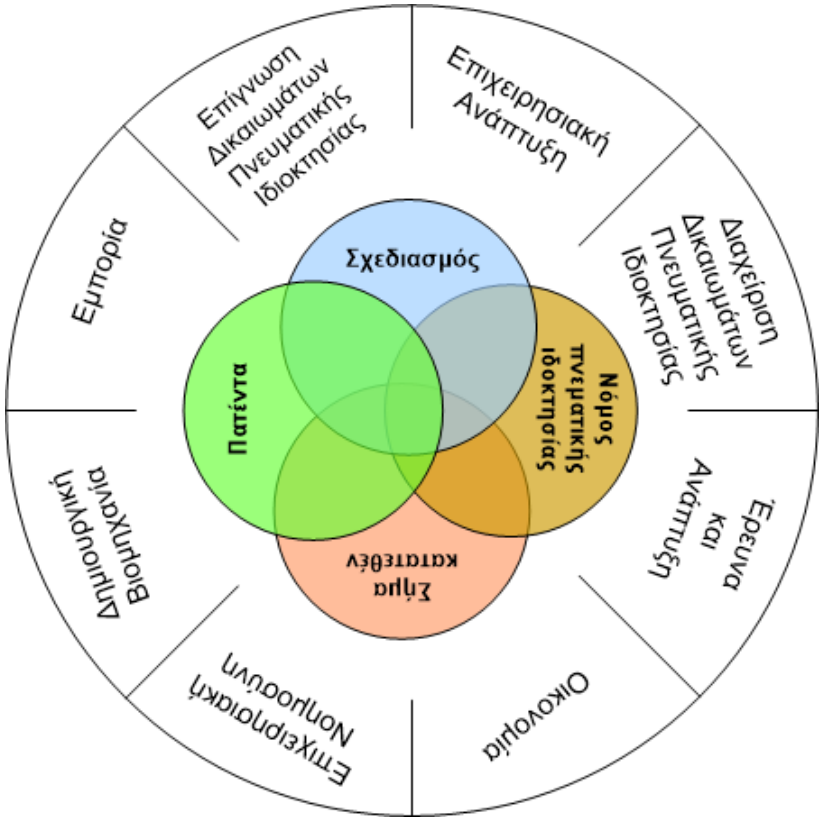
ηλεκτρονικών υπολογιστών" ενσωματώθηκε στην ελληνική έννομη τάξη με την έκδοση του νόμου 2121/1993 περί πνευματικής ιδιοκτησίας, με κάποιες παρεκκλίσεις. Ειδικότερα, αντικείμενο προστασίας είναι τα προγράμματα ηλεκτρονικών υπολογιστών και το προπαρασκευαστικό υλικό σχεδιασμού τους, εφόσον το υλικό είναι τέτοιο ώστε να μπορεί σε ένα μετέπειτα στάδιο να προκύψει από αυτό το πρόγραμμα του ηλεκτρονικού υπολογιστή. Συγκεκριμένα, προστατεύεται κάθε μορφή έκφρασης ενός προγράμματος, ενώ μένουν εκτός προστατευτικού κλοιού οι ιδέες και οι αρχές, στις οποίες βασίζεται οποιοδήποτε στοιχείο του προγράμματος, καθώς οι ιδέες και οι αρχές στις οποίες βασίζονται τα συστήματα διασύνδεσής του, δηλαδή τα μέρη του προγράμματος με τα οποία επικοινωνούν μεταξύ τους το πρόγραμμα, ο ηλεκτρονικός υπολογιστής και ο χρήστης.

Τα προγράμματα καταρχήν προστατεύονται μόνο ως έργα λόγου με το δίκαιο της πνευματικής ιδιοκτησίας και όχι ως εφευρέσεις με το δίκαιο της βιομηχανικής ιδιοκτησίας. Το δίκαιο της πνευματικής ιδιοκτησίας είναι άτυπο, δηλαδή για την προστασία του περιουσιακού και ηθικού δικαιώματος του δημιουργού του προγράμματος ηλεκτρονικού υπολογιστή δεν απαιτείται, ούτε προβλέπεται κάποια τυπική πράξη, ήτοι να ακολουθηθεί κάποια διαδικασία κατοχύρωσης του προγράμματος. Σε περίπτωση που τυχόν ανακύψει κάποιο πρόβλημα λόγω της προσβολής των δικαιωμάτων του δημιουργού πάνω στο πρόγραμμα από τρίτους, είναι θέμα απόδειξης ποιος προηγήθηκε στη δημιουργία του προγράμματος αυτού, η οποία απόδειξη επιτυγχάνεται με κάθε μέσο. Για την απόκτηση βέβαιης χρονολογίας, δηλαδή ενός χρονικού σημείου, κατά το οποίο αποδεικνύεται ότι υπήρχε το πρόγραμμα, μπορεί για παράδειγμα να γίνει πράξη κατάθεσής του (π.χ., εκτυπωμένο και σε CD) σε συμβολαιογράφο. Αυτή η διαδικασία δεν προβλέπεται από τις διατάξεις περί πνευματικής ιδιοκτησίας, αλλά συνηθίζεται στην πράξη προς διευκόλυνση της απόδειξης αυτής καθ αυτής.

Ο Παγκόσμιος Οργανισμός Διανοητικής Ιδιοκτησίας (World Intellectual Property Organization - WIPO), ο οποίος δημιουργήθηκε το έτος 1967 με απώτερο σκοπό να ενθαρρύνει και συνάμα να προσφέρει κίνητρο για δημιουργία, καθώς επίσης και να συμβάλει στην προώθηση της προστασίας των δικαιωμάτων πνευματικής ιδιοκτησίας σε όλο τον κόσμο, είναι υπεύθυνος για τη διοίκηση διεθνών θεμάτων πνευματικής ιδιοκτησίας. Σχετικά με τα λογισμικά τώρα, αυτά προστατεύονται στη Διεθνή Σύμβαση της Βέρνης που διευθύνεται από τον οργανισμό WIPO, ο οποίος έχει κυρωθεί από την Ελλάδα. Έτσι, όσον αφορά τη διεθνή προστασία των προγραμμάτων και μόνον όταν η προστασία ζητείται σε χώρα διαφορετική από τη χώρα προέλευσης του έργου, οι χώρες που έχουν προσχωρήσει στη Σύμβαση αυτή και οι οποίες απαρτίζουν Ένωση κρατών για την προστασία των δικαιωμάτων των δημιουργών, προστατεύουν τα προγράμματα.

Η προστασία των προγραμμάτων γίνεται εξομοιώνοντάς τα με λογοτεχνικά έργα και εφαρμόζοντας τις διατάξεις της αρχής της εθνικής μεταχείρισης των ξένων δημιουργών ή της αρχής της εξομοίωσής τους προς τους υπηκόους της χώρας, όπου ζητείται η έννομη προστασία, σε συνδυασμό πάντα με τις διατάξεις που προσδιορίζουν τη χώρα προέλευσης του προγράμματος. Όταν η νομοθεσία ενός κράτους δεν παρέχει στο δημιουργό μεγαλύτερη προστασία με βάση την αρχή της εθνικής προστασίας, εφαρμόζονται οι διατάξεις για τα ελάχιστα όρια προστασίας που προβλέπει η Σύμβαση. Επιπλέον, η Συνθήκη του WIPO για

την πνευματική ιδιοκτησία, η οποία κυρώθηκε από την Ευρωπαϊκή Ένωση το έτος 2000, συνδέεται με την ανωτέρω Σύμβαση της Βέρνης, δίχως να εισάγει παρεκκλίσεις από τις υποχρεώσεις που έχουν τα κράτη - μέλη σύμφωνα με αυτή [60].



Σχήμα 1.2: Τομείς που απευθύνονται τα δικαιώματα της πνευματικής ιδιοκτησίας.

Ο δημιουργός του εκάστοτε έργου είναι το υποκείμενο του δικαιώματος πνευματικής ιδιοκτησίας. Στο ηπειρωτικό ευρωπαϊκό σύστημα καθώς επίσης και στο ελληνικό δίκαιο, δημιουργός μπορεί να είναι μόνο κάποιο φυσικό πρόσωπο, αφού μόνον ο άνθρωπος έχει τη δυνατότητα να δημιουργεί πρωτότυπα πνευματικά έργα. Εν αντιθέσει στο αμερικανικό δίκαιο, δημιουργός κάλλιστα μπορεί να είναι και νομικό πρόσωπο, το οποίο έχει επενδύσει οικονομικά στη δημιουργία ενός έργου.

Η πνευματική ιδιοκτησία είναι αυτή που προστατεύει τα έργα από τρίτους. Το έργο δημιουργείται και αρχίζει να υπάρχει με την εξωτερίκευσή του από τον δημιουργό του. Ως ιδέα στο μυαλό του δημιουργού προφανώς δεν δύναται να προστατευτεί, καθώς οι τεχνικές οι οποίες αποκρυπτογραφούν εγκεφάλους βρίσκονται σε πολύ πρώιμο στάδιο. Η εξωτερίκευση σε ορισμένες περιπτώσεις μπορεί να είναι και εφήμερη (π.χ., η προφορική απαγγελία ενός ποιήματος). Αντίθετα στο αμερικανικό δίκαιο απαιτείται σταθερή αποτύπωση (π.χ., ένα ποίημα θα προστατεύεται μόνο αν καταγραφεί γραπτώς ή ηχογραφηθεί). Στο σημείο αυτό σημειώνεται πως το δίκαιο προστατεύει μονάχα το συγκεκριμένο εξωτερικευμένο έργο και

όχι τις ιδέες που πιθανόν να κρύβονται από πίσω διακρίνοντας μεταξύ μορφής, αποτύπωσης και ιδέας. Βέβαια η διάκριση αυτή είναι πολύ δύσκολη να πραγματοποιηθεί καθολικά και οριζοντίως καθώς είναι εντελώς υποκειμενική και πολλές φορές μη προσδιορίσιμη [63].

Ο δημιουργός αποκτά πάνω στο έργο του πνευματική ιδιοκτησία η οποία περιλαμβάνει δύο απόλυτα και αποκλειστικά δικαιώματα. Το πρώτο είναι το δικαίωμα της εκμετάλλευσης του έργου (περιουσιακό δικαίωμα), ενώ το δεύτερο είναι το δικαίωμα προστασίας του προσωπικού του δεσμού με το έργο (ηθικό δικαίωμα). Εστιάζοντας σε ακόμη μεγαλύτερο βαθμό στην πνευματική ιδιοκτησία σημειώνεται πως το περιουσιακό δικαίωμα του δημιουργού αναφέρεται στην οικονομική εκμετάλλευση του έργου του. Κάθε εξουσία απαγόρευσης μιας πράξης ή μιας μορφής εκμετάλλευσης σημαίνει αυτομάτως πως ο δημιουργός μπορεί να αξιώνει αμοιβή γι αυτήν.

Έργα συνεργασίας θεωρούνται όσα έχουν δημιουργηθεί με την άμεση σύμπραξη δύο ή περισσοτέρων δημιουργών. Οι δημιουργοί ενός έργου, το οποίο αποτελεί προϊόν συνεργασίας, είναι και οι αρχικοί συνδικαιούχοι του περιουσιακού και του ηθικού δικαιώματος επί του έργου αυτού. Αν δεν έχει πιστοποιηθεί ρητά με κάποιον συγκεκριμένο τρόπο, τα δικαιώματα του έργου ανήκουν κατά ίσα μέρη στους συνδημιουργούς.

Συλλογικά έργα θεωρούνται όσα έχουν δημιουργηθεί με τις αυτοτελείς συμβολές περισσοτέρων δημιουργών υπό την πνευματική διεύθυνση ή αλλιώς καθοδήγηση και το συντονισμό ενός φυσικού προσώπου. Το πρόσωπο αυτό, είναι ο αρχικός δικαιούχος του περιουσιακού και του ηθικού δικαιώματος επί του συλλογικού έργου. Οι δημιουργοί των επιμέρους συμβολών είναι οι αρχικοί δικαιούχοι του περιουσιακού και του ηθικού δικαιώματος επί των συμβολών τους και μόνο, εφόσον αυτές είναι δεκτικές χωριστής εκμετάλλευσης. Όταν ένα έργο θεωρείται σύνθετο, απαρτιζόμενο ουσιαστικά από τμήματα που έχουν δημιουργηθεί χωριστά, οι δημιουργοί των τμημάτων επί του συνθέτου έργου είναι και οι αποκλειστικοί αρχικοί δικαιούχοι των δικαιωμάτων του τμήματος που δημιούργησε ο καθένας τους, εφόσον βέβαια όπως και στα συλλογικά έργα αυτό είναι δεκτικό χωριστής εκμετάλλευσης.

Εστιάζοντας στο περιουσιακό δικαίωμα, αξίζει να σημειωθεί πως αυτό χωρίζεται σε επιμέρους εξουσίες. Η διάκριση αυτή είναι ιδιαίτερα σημαντική, επειδή ο νόμος επιτρέπει ρητά τη χωριστή εκμετάλλευση κάθε μιας από τις εξουσίες αυτές. Ο δημιουργός μπορεί συνεπώς να μεταβιβάσει ή να παραχωρήσει άδεια εκμετάλλευσης μόνο για μία ή για ένα υποσύνολο από αυτές τις εξουσίες, κρατώντας τις άλλες για τον εαυτό του ή παραχωρώντας τις εν συνεχεία σε τρίτους. Οι επιμέρους αυτές εξουσίες είναι οι ακόλουθες:

- **Η εγγραφή:** Η εξουσία εγγραφής είναι η εξουσία η οποία επιτρέπει να αποτυπωθεί μια εκτέλεση του έργου σε ένα υλικό υπόστρωμα (π.χ., η ηχογράφηση μιας μουσικής σύνθεσης). Η εξουσία αυτή αναφέρεται σε έργα που δεν προϋποθέτουν την εγγραφή για την ίδια τους τη δημιουργία (π.χ., οπτικοακουστικά έργα), αλλά σε έργα που προϋπάρχουν της εκτέλεσης και εγγραφής (π.χ., μουσική σύνθεση, θεατρικό έργο).

- **Η αναπαραγωγή:** Η εξουσία αναπαραγωγής είναι η εξουσία να παραχθούν νέα σταθερά υλικά υποστρώματα, τα οποία επαναλαμβάνουν την υλική ενσωμάτωση του έργου (π.χ., φωτοτυπίες, cd, εκτύπωση βιβλίων, αντιγραφή πινάκων ζωγραφικής στη

μνήμη του υπολογιστή). Αναπαραγωγή είναι και η τοποθέτηση ενός έργου (uploading) σε έναν διακομιστή (server), καθώς και το κατέβασμα (downloading) και αποθήκευση του αντιτύπου μέσω διαδικτύου στον υπολογιστή του χρήστη.

- **Η δημιουργία παράγωγου έργου:** Η εξουσία δημιουργίας παράγωγου έργου καλύπτει τη μετάφραση, τη διασκευή, τη μετατροπή (π.χ., δημιουργία ταινίας από βιβλίο) του αρχικού έργου. Καταλαμβάνει όμως μόνο τη δημόσια χρήση του παράγωγου έργου. Ουσιαστικά ο δημιουργός δεν μπορεί να απαγορεύσει την ιδιωτική μετάφραση του έργου του, παρά μόνο τη δημόσια χρήση αυτής της μετάφρασης.

- **Η διανομή:** Η ενέργεια αυτή αφορά τη θέση σε κυκλοφορία ήδη (νόμιμα ή παράνομα) αναπαραχθέντων αντιτύπων. Ο δημιουργός έχει την εξουσία να αποφασίζει αν και με ποιον τρόπο θα διανεμηθούν τα αντίτυπα του έργου του. Η εξουσία του όμως περιορίζεται μονάχα στην πρώτη διάδοση του κάθε αντιτύπου. Από τη στιγμή που ένα αντίτυπο διατεθεί με τη συναίνεσή του στο κοινό (π.χ., πωληθεί ένα αντίτυπο του βιβλίου του), ο δημιουργός δεν μπορεί πλέον να καθορίσει την περαιτέρω κυκλοφορία του εντός των ορίων της Ευρωπαϊκής Ένωσης. Ο αγοραστής του βιβλίου καθίσταται δηλαδή ελεύθερος να διαθέσει (πωλήσει) το συγκεκριμένο αντίτυπο περαιτέρω χωρίς τη συναίνεση του δημιουργού.

- **Η εκμίσθωση και ο δημόσιος δανεισμός:** Η εξουσία αυτή αναφέρεται στο γεγονός πως η πώληση ενός ή περισσοτέρων αντιτύπων δε δίνει την εξουσία στον αγοραστή να τα εκμισθώνει δημόσια. Η εκμίσθωση και ο δημόσιος δανεισμός (π.χ., βιβλία από βιβλιοθήκες) είναι χωριστή εξουσία του δημιουργού. Χαρακτηριστικό παράδειγμα είναι τα μαγαζιά τα οποία ενοικιάζουν ταινίες στα οποία η αγορά αντιτύπων μιας βιντεοκασέτας ή ενός dvd δεν παρέχει στον αγοραστή το δικαίωμα να τα νοικιάζει σε άλλους μέσω τέτοιων μαγαζιών, αλλά πρέπει να πάρει ειδική άδεια από το δημιουργό προς τούτο.

- **Η δημόσια εκτέλεση και παρουσίαση:** Ο δημιουργός έχει τη χωριστή εξουσία να επιτρέπει ή να απαγορεύει το να καθίσταται το έργο του προσιτό στο κοινό. Η δημόσια εκτέλεση και η παρουσίαση σε αντίθεση με την αναπαραγωγή δεν αφορούν τη δημιουργία νέων υλικών φορέων του έργου (π.χ., φωτοτυπία, αντιγραφή cd). Η δημόσια εκτέλεση είναι η παρουσίαση στο κοινό με την παρεμβολή συνήθως κάποιου άλλου καλλιτέχνη και ενδεχομένως κάποιου μηχανήματος (π.χ., δημόσια εκτέλεση ενός θεατρικού από έναν ηθοποιό ή μιας σύνθεσης από ένα μουσικό ζωντανά ή με δημόσια εκτέλεση ηχογράφησης). Χαρακτηριστικό παράδειγμα δημόσιας εκτέλεσης είναι η μουσική σε μπαρ και γενικά σε δημόσιους χώρους. Η παρουσίαση είναι το να καταστεί το έργο αυτούσιο χωρίς την παρεμβολή άλλου καλλιτέχνη προσιτό στον καθένα διά ζώσης, ενσυρμάτως ή ασυρμάτως (π.χ., μέσω διαδικτύου). Χαρακτηριστικά τέτοια παραδείγματα παρουσίασης είναι η έκθεση έργων ζωγραφικής, γλυπτικής και φωτογραφίας.

- **Η ραδιοτηλεοπτική μετάδοση:** Είναι η χωριστή εξουσία του δημιουργού να μπορεί να απαγορεύει τη μετάδοση του έργου του μέσω τηλεόρασης (επίγειας ή δορυφορικής) και ραδιοφώνου.

Σύμφωνα με τον οργανισμό WIPO, τα δικαιώματα πνευματικής ιδιοκτησίας (copyrights) είναι η νόμιμη προστασία που εκτείνεται στον κάτοχο των δικαιωμάτων για την πρωτότυπη δουλειά που δημιούργησε. Η προστασία αυτή συμβιβάζει τις δύο κύριες ομάδες δικαιωμάτων, οι οποίες όπως έχουν αναφερθεί είναι τα οικονομικά και τα πνευματικά δικαιώματα [65].

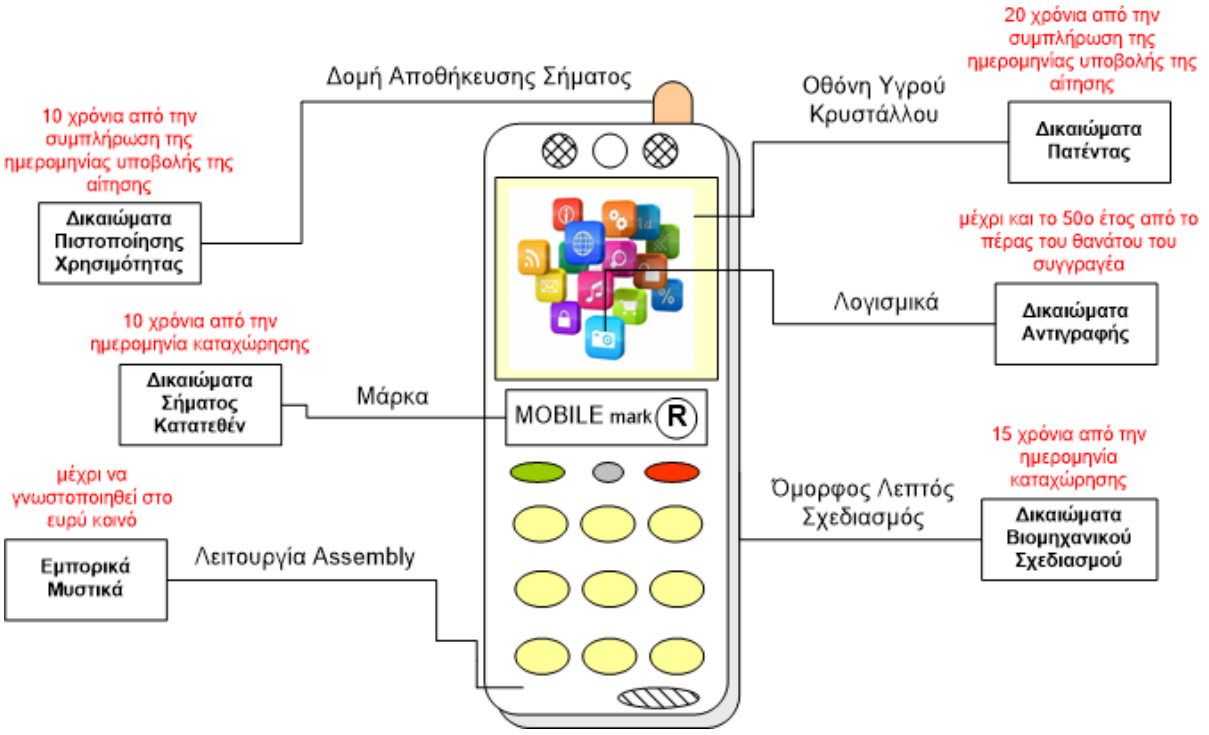
Η ραγδαία ανάπτυξη των χρηστών του διαδικτύου ανά την υφήλιο, η οποία έχει γνωρίσει ιδιαίτερη άνθηση την τελευταία δεκαετία λόγω της εύκολης και συνάμα ταχύτατης διάδοσης ψηφιακού υλικού, καθώς επίσης και η αδυναμία από τεχνικής άποψης, προστασίας των νόμιμων δικαιούχων, οδηγεί με την πάροδο του χρόνου σε μία ανεξέλεγκτη καταπάτηση πνευματικών δικαιωμάτων. Σύμφωνα με την τελευταία παγκόσμια μελέτη περί πειρατείας λογισμικού του οργανισμού Συμμαχία Επιχειρήσεων Λογισμικού (Business Software Alliance - BSA), ενός επιφανή μη κερδοσκοπικού οργανισμού του κλάδου της Πληροφορικής ο οποίος ιδρύθηκε το 1988 και στοχεύει στην προώθηση ενός ασφαλούς και παράλληλα νόμιμου ψηφιακού κόσμου, εκπροσωπώντας τη βιομηχανία του εμπορικού λογισμικού και των συνεργαζόμενων κατασκευαστών υλικού, το έτος 2011 πάνω από τους μισούς χρήστες (ποσοστό 57%) ηλεκτρονικών υπολογιστών παραδέχεται πως έχει χρησιμοποιήσει πειρατικό λογισμικό [5].

Επιπλέον για το έτος 2013, ο οργανισμός BSA σε συνεργασία με το INSEAD, ενός εκ των κορυφαίων πανεπιστημίων διοίκησης σε παγκόσμια κλίμακα, πραγματοποιούν μία νέα έρευνα με εξίσου αξιοσημείωτα αποτελέσματα. Η έρευνα αυτή αποκαλύπτει πως η αύξηση χρήσης γνήσιου λογισμικού θα είχε μεγαλύτερα οφέλη για την ελληνική οικονομία σε αντίθεση με αντίστοιχη αύξηση χρήσης πειρατικού λογισμικού. Το συμπέρασμα αυτό είναι καθ όλα προφανές, δεν είναι όμως διόλου προφανές πως σύμφωνα με την έρευνα αυτή μία αύξηση 1% στη χρήση γνήσιου λογισμικού θα επέφερε μία αύξηση της τάξεως των 388 εκατομμυρίων δολαρίων στην εθνική παραγωγή σε σύγκριση με τα 90 εκατομμύρια δολάρια από μια αντίστοιχη αύξηση πειρατικού λογισμικού. Αυτό, εν ολίγοις σημαίνει ότι η χρήση γνήσιου λογισμικού θα επέφερε 299 εκατομμύρια δολάρια επιπλέον οικονομικό όφελος στο ελληνικό δημόσιο. Επιπρόσθετα, η προκειμένη μελέτη επιβεβαιώνει ότι η αυξανόμενη χρήση γνήσιου λογισμικού συνεπάγεται σημαντική αύξηση στο Ακαθάριστο Εγχώριο Προϊόν - ΑΕΠ, ενώ τα οικονομικά οφέλη που προκύπτουν από τη χρήση του προφανώς και είναι πολύ υψηλότερα από εκείνα του πειρατικού.

Αμεσο συμπέρασμα της έρευνας αυτής είναι πως η χρήση γνήσιου λογισμικού μειώνει τον ελλοχεύωντα κίνδυνο και επιτρέπει υψηλότερες επιδόσεις που αφορούν την καρδία της λειτουργίας των υγειών επιχειρήσεων. Παράλληλα, επιβεβαιώνει ότι η χρήση γνήσιου λογισμικού δεν είναι μόνο ωφέλιμη για τις επιχειρήσεις αυτές καθ αυτές, αλλά αποτελεί και αξιοσέβαστη κινητήρια δύναμη για την εθνική οικονομική ανάπτυξη, είτε σε περιόδους ευημερίας είτε σε περιόδους οικονομικής κρίσης [6].

Κοιτάζοντας όμως, πίσω από αυτά τα διόλου ασήμαντα αποτελέσματα των ερευνών αυτών με μία μικρή ανάλυση γίνεται αμέσως φανερό πως οι εταιρίες οι οποίες κατασκευάζουν

λογισμικά, προκειμένου να επιβιώσουν και να αναπτυχθούν, πρέπει να έρθουν πιο κοντά στους χρήστες ηλεκτρονικών υπολογιστών, καθώς και οι δύο οι πλευρές θα έχουν να αποκομίσουν κέρδη με την κίνηση αυτή. Αυτό είναι αληθές, διότι από την μία πλευρά οι χρήστες θα μπορούν να χρησιμοποιούν τα αυθεντικά προγράμματα των εκάστοτε εταιριών, με την αντίστοιχη υποστήριξη (support) και όχι τα πειρατικά προγράμματα τα οποία διανέμονται σχεδόν ελεύθερα στο διαδίκτυο, και παράλληλα από την άλλη πλευρά οι εταιρίες ρίχνοντας τις τιμές που προσέφεραν τα προϊόντα τους ως τώρα, θα καταφέρουν να αποκτήσουν πολύ περισσότερους πελάτες. Με απλά λόγια, κατεβάζοντας τις τιμές τους, το όφελος θα είναι διπλό για τις εταιρίες, καθώς και θα αποκτήσουν περισσότερα κέρδη, αφού θα πωλούν περισσότερα προϊόντα σε αριθμό με αποτέλεσμα μεγαλύτερο μερίδιο αγοράς για αυτές και θα έχουν ευχαριστημένους πελάτες με αποτέλεσμα καλύτερη διαφήμιση.



Σχήμα 1.3: Τα τμήματα ενός κινητού τηλεφώνου τα οποία προστατεύονται με δικαιώματα.

Βασικό ερέθισμα σε αυτήν την ασύστολη χρήση πειρατικών λογισμικών είναι το γεγονός ότι ενώ το υλικό που χρησιμοποιούν οι ηλεκτρονικοί υπολογιστές γίνεται με την πάροδο των χρόνων φθηνότερο, τα λογισμικά που χρησιμοποιούν οι ηλεκτρονικοί υπολογιστές γίνονται κατά πολύ ακριβότερα. Επί της ουσίας, όπως έχει δείξει η ως τώρα ιστορία των ηλεκτρονικών υπολογιστών, καθώς τα χρόνια περνούν οι άνθρωποι θα έχουν την δυνατότητα να αγοράζουν με λιγότερα χρήματα ηλεκτρονικούς υπολογιστές καλύτερων δυνατοτήτων, οι οποίοι θα χρησιμοποιούν όμως κατά πολύ πιο ακριβά λογισμικά. Αμεσο επακόλουθο αυτής της αντιστρόφως ανάλογης μεταβολής, είναι οι χρήστες ηλεκτρονικών

υπολογιστών με σκοπό να κάνουν την δουλειά τους να καταφεύγουν σε λύσεις όχι και τόσο νόμιμες, αλλά σε μερικές χώρες ιδιαίτερα ακίνδυνες, αποκτώντας πειρατικά λογισμικά με τις εταιρίες παραγωγής λογισμικών να βάλλονται στο επίκεντρο παραμένοντας αμέτοχες στα πεπραγμένα δίχως να είναι σε θέση να πράξουν τα πρέποντα.

Ο στόχος των εκάστοτε υποκλοπών λογισμικού, είτε μπορεί να είναι η κατανόηση των αλγορίθμων που κωδικοποιούνται μέσα σε αυτά και εν συνεχεία η παραγωγή νέου λογισμικού βάσει των εν λόγω αλγορίθμων, είτε μπορεί να είναι η απευθείας χρήση και προώθηση του λογισμικού χωρίς τη νόμιμη άδεια. Προκειμένου όμως να γίνει απευθείας χρήση και προώθηση του λογισμικού, πρέπει ο εκάστοτε κακόβουλος χρήστης στηριζόμενος στην αγχίνοιά του να βεβαιωθεί σε όσο το δυνατόν μεγαλύτερο βαθμό είναι εφικτό, πως μέσα στο λογισμικό δεν εμπεριέχονται στοιχεία τα οποία μπορούν σε μελλοντικό χρόνο να τον ενοχοποιήσουν. Ακόμα και στην περίπτωση όπου αυτός επιθυμεί να κατανοήσει τους κωδικοποιημένους αλγορίθμους και να δημιουργήσει δικό του λογισμικό βάσει αυτούς, πρέπει να είναι όσο το δυνατόν πιο σίγουρος πως έχει απαλλαγεί από οτιδήποτε στοιχείο είναι άσχετο με τους αλγορίθμους αυτούς καθ αυτούς. Τα στοιχεία αυτά συνήθως είναι κάποιες δομές δεδομένων (π.χ., γραφήματα, πίνακες, μοτίβα εντολών) και κύριο μέλημα του κακόβουλου χρήστη είναι αρχικά να τις αναγνωρίσει στο εσωτερικό του κώδικα και εν συνεχεία να τις απενεργοποιήσει ή να τις αλλοιώσει ή ακόμα και να τις εξάγει εντελώς από αυτόν προτού προβεί σε οποιαδήποτε είδους ενέργεια, δίχως να αλλοιώσει την όποια λειτουργικότητα.

Στην περίπτωση κατά την οποία ο εισβολέας έχει καταφέρει να απαλλαγεί από τυχόν ενοχοποιητικά στοιχεία τα οποία βρισκόντουσαν στο εσωτερικό του κώδικα, τότε ο πραγματικός ιδιοκτήτης καθίσταται απευθείας ανήμπορος να διεκδικήσει τα αυτονόητα (π.χ., μέσω μίας δικαστικής διαμάχης στα δικαστήρια). Επομένως όσο η αντίληψη και οι γνώσεις των κακόβουλων χρηστών αυξάνεται στον τομέα της Πληροφορικής τόσο πιο δύσκολη γίνεται η δουλειά των εταιριών στην καταπολέμηση της πειρατείας, καθώς είναι αναγκασμένες προκειμένου να παραμείνουν αλώβητες να εφευρίσκουν τεχνικές προστασίας ολοένα και πιο έξυπνες και συνάμα αποτελεσματικές. Ουσιαστικά δηλαδή να βρίσκονται πάντα τουλάχιστον ένα βήμα μπροστά, από οποιονδήποτε επιθυμεί και προσπαθεί να καρπωθεί δικά τους περιουσιακά στοιχεία.

1.3 Τεχνικές Προστασίας Λογισμικού

Τα λογισμικά τίθονται σε κυκλοφορία ανά τον κόσμο είτε ως λογισμικά ανοικτού κώδικα (open source) είτε ως λογισμικά κλειστού κώδικα (closed source) τα λεγόμενα και αλλιώς ως ιδιοταγές λογισμικά (proprietary software). Το γεγονός αυτό είναι καθ όλα ανεξάρτητο από τους σκοπούς τους οποίους ενδεχομένως να υπηρετούν τα εκάστοτε λογισμικά. Η βασική τους διαφορά είναι πως τα πρώτα διανέμονται εντελώς δωρεάν και ο καθένας μπορεί να έχει πρόσβαση στον κώδικά τους και να τα τροποποιεί, ενώ τα δεύτερα προκειμένου κάποιος να τα αποκτήσει πρέπει να καταβάλει κάποιο χρηματικό ποσό δίχως όμως να του παρέχεται η δυνατότητα τροποποίησής τους ή πρόσβασης στον πηγαίο κώδικα.

Συγκεκριμένα ένα λογισμικό ανοικτού τύπου επιτρέπει στον χρήστη την αντιγραφή, την τροποποίηση και την διανομή από οποιονδήποτε και για οποιοδήποτε σκοπό, ακόμη δε και εμπορικό. Πάρα ταύτα, πολλές φορές χρησιμοποιείται και ο όρος ανοικτό λογισμικό που αφορά λογισμικό για το οποίο διατίθεται ο πηγαίος κώδικας, αλλά υπάρχουν περιορισμοί στην ελεύθερη χρήση του (π.χ., στην εμπορική του χρήση). Ουσιαστικά, αυτό που γίνεται είναι να περιέχει περιοριστικούς όρους όπως η υποχρεωτική αναφορά στο όνομα του δημιουργού ή κατόχου των πνευματικών δικαιωμάτων, δίχως αυτοί οι όροι να περιορίζουν στο ελάχιστο τις προηγούμενες ελευθερίες τροποποίησης και διακίνησης.

Από την άλλη πλευρά όμως ένα λογισμικού κλειστού τύπου παρέχει την δυνατότητα στον χρήστη να το χρησιμοποιεί μόνον εντός περιορισμένου περιβάλλοντος και μάλιστα ορισμένες φορές και μονάχα σε έναν μικρό αριθμό ηλεκτρονικών υπολογιστών (π.χ., το λογισμικό των Windows μπαίνει σε 3 HDDs ή SSDs το πολύ), απαγορεύοντας την αποσυμπίληση, την ανάλυση, την τροποποίηση, καθώς και την αναδημιουργία του. Με τον όρο αποσυμπίλησή δύναται να εννοηθεί η προσπάθεια ανάκτησης του πηγαίου κώδικα. Συνήθως απαγορεύονται επίσης η αντιγραφή και διανομή του, είτε αυτή πραγματοποιείται αφιλοκερδώς, είτε αυτή γίνεται επί πληρωμής από τρίτα πρόσωπα, καθώς και η χρήση του από ένα μεγάλο αριθμό ανθρώπων.

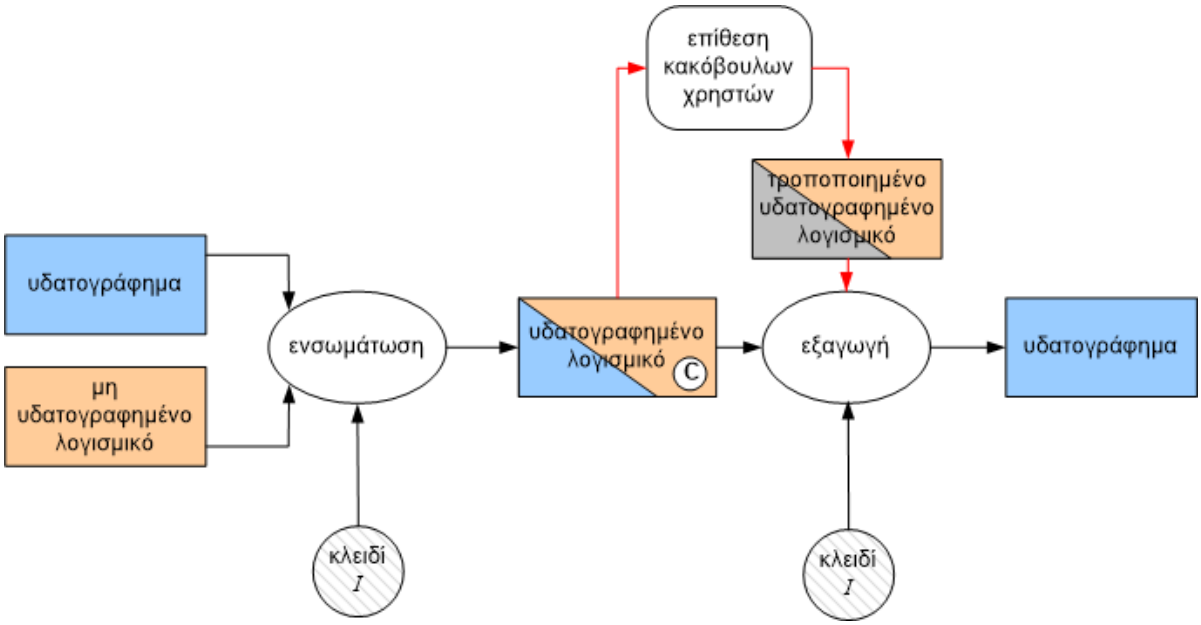
Προκειμένου οι πραγματικοί ιδιοκτήτες να είναι σε θέση να αποδεικνύουν ανά πάσα ώρα και στιγμή την γνησιότητα των πνευματικών τους δημιουργημάτων τους, όπου στη συγκεκριμένη περίπτωση είναι τα λογισμικά, καθώς επίσης και να κάνουν όσο γίνεται πιο δύσκολο, απαιτητικό και χρονοβόρο το έργο των κακόβουλων χρηστών έχουν επινοηθεί ορισμένες τεχνικές προστασίας. Αυτές είναι η υδατογράφηση, η συσκότιση και η θωράκιση λογισμικού. Η κάθε μία από τις τεχνικές αυτές έχει επινοηθεί για συγκεκριμένο σκοπό και μπορούν να εφαρμοστούν με οποιονδήποτε συνδυασμό κάθε φορά. Πάρα ταύτα, καμία από αυτές δεν θέτει ένα αδιαπέραστο τοίχος μεταξύ των λογισμικών και των κακόβουλων χρηστών.

1.3.1 Υδατογράφηση

Όταν ένα λογισμικό αρχίσει να διανέμεται, είναι πολύ πιθανόν να καταλήξει με τον έναν ή τον άλλον τρόπο σε χέρια κακόβουλων χρηστών. Αν τα πνευματικά του δικαιώματα δεν είναι κατοχυρωμένα, τότε η κατάσταση αυτή αποτελεί κατάσταση κινδύνου καθώς το λογισμικό επί της ουσίας δεν έχει ιδιοκτήτη. Πράγμα τα οποίο σημαίνει πως ο οποιοσδήποτε μπορεί εύκολα να το υιοθετήσει και να επωφεληθεί από αυτό είτε οικονομικά, είτε ηθικά δίχως να χρειάζεται να δώσει κάποια αναφορά. Μία τεχνική βάσει της οποίας κάποιος μπορεί να κατοχυρώσει τα πνευματικά του δικαιώματα έναντι ενός λογισμικού το οποίο και του ανήκει είναι η υδατογράφηση λογισμικού (software watermarking).

Υδατογράφημα λογισμικού ονομάζεται μία δομή ή αλλιώς ένα μοναδικό αναγνωριστικό (unique identifier) το οποίο ενσωματώνεται είτε στον πηγαίο, είτε στον εκτελέσιμο κώδικα ενός λογισμικού. Το αναγνωριστικό αυτό επί της ουσίας είναι επιπρόσθετος κώδικας ο οποίος προστίθεται στον αρχικό και η μορφή αναπαράστασής του μπορεί να είναι οτιδήποτε (π.χ., μετάθεση, γράφημα, πίνακας, συμβολοσειρά). Στην γενική της μορφή η υδατογράφηση

λογισμικού αποτελείται από δύο κύριες φάσεις, την φάση της σχεδίασης και δημιουργίας του υδατογραφήματος και την φάση της εφαρμογής υδατογράφησης στον κώδικα όπου εκεί λαμβάνουν χώρα οι λειτουργίες ή αλλιώς διαδικασίες της ενσωμάτωσης και της εξαγωγής της δημιουργηθείσας δομής.



Σχήμα 1.4: Μοντέλο συστήματος υδατογράφησης κώδικα.

Στην γενική του μορφή το πρόβλημα της υδατογράφησης λογισμικού μπορεί να περιγραφεί ως το πρόβλημα της ενσωμάτωσης μίας δομής, έστω w, στον κώδικα ενός προγράμματος, έστω P, με επιδιωκόμενο αποτέλεσμα την κατασκευή ενός νέου υδατογραφημένου προγράμματος, έστω P_w, τέτοιο ώστε η πλέον ενσωματωμένη δομή w να μπορεί να σε μελλοντικό χρόνο να εντοπιστεί, να αναγνωριστεί και να εξαχθεί από το P_w ακόμη και αν το P_w έχει υποστεί τροποποιήσεις (επιθέσεις), χωρίς ταυτόχρονα να αλλοιωθεί στο ελάχιστο η λειτουργικότητά του [48].

Με τον όρο λειτουργικότητα (functionality), δύναται να εννοηθεί πως το υδατογραφημένο πρόγραμμα P_w για κάθε ακολουθία εισόδου I παράγει πανομοιότυπο αποτέλεσμα με το μη υδατογραφημένο πρόγραμμα P στα ίδια χρονικά επίπεδα. Πιο τυπικά, δοθέντος ενός προγράμματος P, ενός υδατογραφήματος w και ενός κλειδιού key το πρόβλημα της υδατογράφησης λογισμικού μπορεί να περιγραφεί με τις ακόλουθες δύο συναρτήσεις οι οποίες ορίζουν επακριβώς το ζητούμενο:

- $embed(P, w, key) \longrightarrow P_w$
- $extract(P_w, key) \longrightarrow w$

Για λόγους καλύτερης κατανόησης σημειώνεται πως το κλειδί είναι εκείνη η ακολουθία εισόδου (π.χ., μία σειρά από πλήκτρα τα οποία πρέπει να πληκτρολογηθούν και ύστερα το πρόγραμμα να τερματιστεί ή ένα αρχείο το οποίο πρέπει να δοθεί ως είσοδος στο πρόγραμμα) η οποία δίνεται στο υδατογραφημένο λογισμικό προκειμένου να πραγματοποιηθεί η εξαγωγή του υδατογραφήματος. Το γεγονός αυτό όμως δεν σημαίνει πως για να εφαρμοστεί μία τεχνική υδατογράφησης λογισμικού θα πρέπει υποχρεωτικά να υπάρχει κάποιο κλειδί. Αυτό όπως θα γίνει γνωστό στο Κεφάλαιο 2 έχει να κάνει ανάλογα με το αν το υδατογράφημα έχει την ιδιότητα του στατικού ή δυναμικού υδατογραφήματος.

Η τεχνική της υδατογράφησης και συγκεκριμένα της υδατογράφησης λογισμικού, η οποία συχνά χαρακτηρίζεται και ως η τελευταία γραμμή άμυνας ενάντια στην πειρατεία, επικεντρώνεται στην ενσωμάτωση και στην μετέπειτα εξαγωγή πληροφορίας μέσα από κώδικα του λογισμικού. Σκοπός της είναι να πιστοποιηθεί η αυθεντικότητά του εκάστοτε λογισμικού, η αναγνώριση του δημιουργού καθώς και να τεθεί ένα εμπόδιο σε όσους επιθυμούν και επιδιώκουν να ιδιοποιήσουν αλλότρια πνευματική ιδιοκτησία. Στη βέλτιστη περίπτωση η υδατογράφηση πρέπει να ενσωματώνει πληροφορία στο λογισμικό τέτοια ώστε να καθίσταται αχώριστη από αυτό και να αντιστέκεται σε κάθε είδους επίθεση έτσι ώστε να μπορεί να εξαχθεί από αυτό αβίαστα. Προσοχή όμως, όπως προαναφέρθηκε και προηγουμένως, αυτού του είδους η τεχνική δεν απαγορεύει σε κανέναν να υποκλέψει, παρά μόνο θέτει ένα τρόπον τινά ηθικό εμπόδιο.

1.3.2 Συσκότιση

Υπάρχουν περιπτώσεις στις οποίες αυτό που είναι επιθυμητό δεν είναι η ενσωμάτωση κάποιας δομής μέσα στον κώδικα του λογισμικού, αλλά η θόλωσή του μέσω μετασχηματισμών. Η επιθυμία αυτή επιτυγχάνεται με την τεχνική της συσκότισης (obfuscation). Στόχοι της τεχνικής αυτής είναι, πρώτον να καταστήσει των κώδικα δυσνόητο στα μάτια κακόβουλων χρηστών και δεύτερον να ακυρώσει τις τεχνικές της αντιστροφής μηχανικής (reverse engineering) και της αποσυσκότισης (deobfuscation) στο εκτελέσιμο αρχείο. Στην πραγματικότητα οι εν λόγω τεχνικές προκειμένου να εφαρμοστούν, σε ορισμένες των περιπτώσεων ο κώδικας χρήζει χειροκίνητης αποσφαλμάτωσης (manually debugging), η οποία σε περιπτώσεις όπου τα λάθη (bugs) βρίσκονται σε πολλά και συνάμα τυχαία σημεία στον κώδικα, ενώ παράλληλα η έκταση του λογισμικού είναι μεγάλη, καθίσταται αρκετά χρονοβόρα λειτουργία.

Ο λόγος για τον οποίον είναι επιθυμητή η θόλωση ενός κώδικα, είναι διττός. Κάποιος ο οποίος θέλει να υποκλέψει ένα λογισμικό, είτε μπορεί να θέλει να πάρει στα χέρια του αυτούσιο τον πηγαίο κώδικα, γεγονός όπου αν η γλώσσα δεν είναι πρόσφορη (π.χ., C, Haskell) όπως είναι η γλώσσα προγραμματισμού Java, είναι κάπως δύσκολο να συμβεί, είτε μπορεί να θέλει απλά να κατανοήσει τον κωδικοποιημένο αλγόριθμο προκειμένου να κατασκευάσει αυτός την δική του έκδοση μελλοντικά, γεγονός το οποίο είναι ευκολότερο από την αποκόμιση καθαρού πηγαίου κώδικα. Και στις δύο περιπτώσεις, αυτό μπορεί να αποφευχθεί ή τουλάχιστον να αποτελέσει επίπονη διαδικασία εφαρμόζοντας αποτελεσματικά την τεχνική της συσκότισης.

Ένας γρήγορος τρόπος διάκρισης των τεχνικών συσκότισης είναι να χωριστούν σε μονόδρομες και σε αμφίδρομες. Οι μονόδρομες τεχνικές είναι αυτές οι οποίες όταν πραγματοποιηθούν στον κώδικα του λογισμικού αφαιρούν την δυνατότητα να έρθει ο κώδικας στην πρότερή του μορφή (π.χ., αλλαγή ονομάτων ή διαγραφή πληροφοριών για την αποσφαλμάτωση). Από την άλλη πλευρά οι αμφίδρομες τεχνικές όταν εφαρμοστούν στον κώδικα δεν αφαιρούν την δυνατότητα να έρθει ο κώδικας στην πρότερή του μορφή, απλά αυτού του είδους οι τεχνικές ενδέχεται όταν εφαρμοστούν να απαιτούν για παράδειγμα γραμμικό χρόνο και σε περίπτωση όπου χρειαστεί ο κώδικας να έρθει στην αρχική του μορφή (αντίστροφη διαδικασία) να χρειάζεται εκθετικός χρόνος. Το γεγονός αυτό στην τεχνική της συσκότισης είναι ιδιαίτερα επιθυμητό καθώς τίθεται ένα σημαντικό χρονικό εμπόδιο στους κακόβουλους χρήστες [26] [8] [52] [57] [3].

Όπως στην τεχνική της υδατογράφησης, έτσι και στην τεχνική της συσκότισης υπάρχει ο ορισμός του βασικού προβλήματος. Δοθέντος ενός συνόλου μετασχηματισμών συσκότισης $T = T_1, T_2, \ldots, T_n$ και ενός προγράμματος P το οποίο εμπεριέχει αντικείμενα πηγαίου κώδικα (π.χ., κλάσεις, συναρτήσεις, συνθήκες ελέγχου, δομές επανάληψης) το ζητούμενο είναι να κατασκευαστεί ένα καινούριο πρόγραμμα $P' = \ldots, S_{j'} = T_i(S_{j'}), \ldots$ έτσι ώστε να ισχύουν τα ακόλουθα:

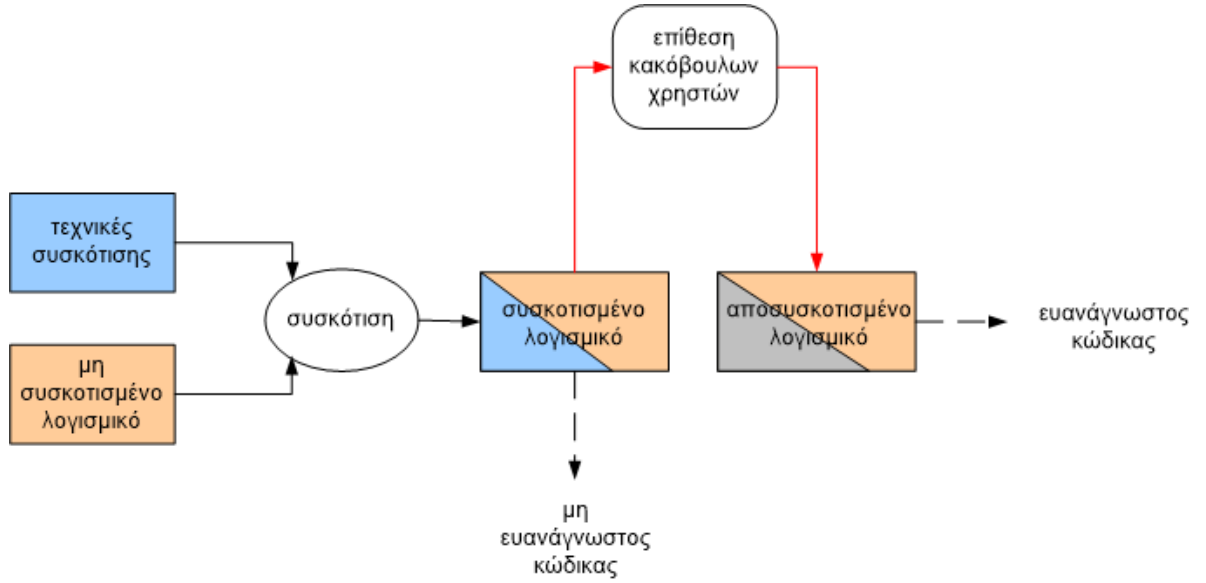
- Το P' να έχει την ίδια ακριβώς λειτουργικότητα με το P.
- Η συσκότιση στο P' να καθιστά τις τεχνικές της αποσυσκότισης (deobfuscation) και της αντίστροφης μηχανικής (reverse engineering) πολύ περισσότερο χρονοβόρες διαδικασίες από ότι θα ήταν αν λάμβαναν χώρα στο P.
- Η ανθεκτικότητα του μετασχηματισμού $T_i(S_j)$ να είναι η μέγιστη δυνατή, έτσι ώστε είτε να καθίσταται πολύ απαιτητικό να φτιαχτεί ένα αυτόματο εργαλείο προκειμένου να αναιρέσει τους μετασχηματισμούς, είτε αν αυτό κατασκευαστεί να είναι ιδιαίτερα χρονοβόρο στην εφαρμογή του.
- Η μη ορατότητα κάθ ενός από τους μετασχηματισμούς $T_i(S_j)$ να είναι η μέγιστη δυνατή, έτσι ώστε οι στατιστικές ιδιότητες του $S_{j'}$ να είναι παραπλήσιες με του S_j.
- Το κόστος του P' σε χρόνο εκτέλεσης και σε χώρο αποθήκευσης (μόνιμης ή προσωρινής κατά την διάρκεια εκτέλεσης) να είναι το ελάχιστο δυνατό.

Η συσκότιση κώδικα θα έλεγε κανείς πως μοιάζει με την βελτιστοποίηση κώδικα (optimization). Αυτές οι δύο οι έννοιες επί της ουσίας είναι πολύ κοντά, γι αυτό και κάποιες από τις τεχνικές που χρησιμοποιούνται από τις δύο αυτές τεχνικές είναι κοινές (π.χ., συγχώνευση μεθόδων, διαγραφή πληροφοριών για την αποσφαλμάτωση, ξεδίπλωμα επαναλήψεων). Η βασική τους διαφορά, η οποία και τις χαρακτηρίζει, είναι πως από την μία πλευρά η συσκότιση κώδικα στοχεύει στην κατασκευή ενός δυσνόητου αντικειμένου προς τρίτους θέλοντας παράλληλα να ελαχιστοποιήσει τον χρόνο εκτέλεσής του, ενώ από την άλλη πλευρά η βελτιστοποίηση κώδικα στοχεύει μονάχα στην ελαχιστοποίηση του χρόνου εκτέλεσης αδιαφορώντας εντελώς για πράξεις εναντίων τρίτων [29].

Οι μετασχηματισμοί συσκότισης κατατάσσονται σε κατηγορίες και αξιολογούνται βάσει ορισμένων κριτηρίων αξιολόγησης. Τα κριτήρια αυτά είναι τα εξής:

- **Βαθμός δυσκολία κατανόησης** (potency): Σε ποιο βαθμό δυσκολίας κατατάσσεται η αποκωδικοποίηση του κώδικα από αυτόν που τον διαβάζει προκειμένου να κατανοήσει την λειτουργία του.
- **Ανθεκτικότητα** (resilience): Πόσο αποτελεσματικά αντιστέκονται οι μετασχηματισμοί του κώδικα είτε σε εργαλεία αποσυσκότισης, είτε σε χειροκίνητες τροποποιήσεις.
- **Κόστος** (cost): Πόσο πιο αργή καθίσταται η ροή εκτέλεσης του λογισμικού και πόσο περισσότερο αποθηκευτικό χώρο χρειάζεται.
- **Ποιότητα** (quality): Πόσο αποτελεσματικά συνδυάζονται τα προηγούμενα κριτήρια.

Πάρα ταύτα το κύριο μειονέκτημα των συσκοτίσεων, το οποίο πολύ πιθανόν να είναι και αναπόφευκτο είναι, πως όσο πιο καλά συσκοτίζουν έναν κώδικα λογισμικού, τόσο περισσότερο αργό τον καθιστούν κατά την διάρκεια εκτέλεσής του. Με αποτέλεσμα να δημιουργείται ένα σοβαρό θέμα για το επίπεδο συσκότισης που είναι επιθυμητό να προσφερθεί κάθε φορά (trade off).



Σχήμα 1.5: Μοντέλο συστήματος συσκότισης κώδικα.

Πάντα όμως, κάποιος από τους κακόβουλους χρήστες θα είναι σε θέση να σπάσει τις τεχνικές υδατογράφησης και συσκότισης, ξεδιαλύνοντας έτσι την μορφή του κώδικα και αναγνωρίζοντας με επιτυχία το υδατογράφημα μέσα σε αυτόν, καθιστώντας έτσι τον πραγματικό ιδιοκτήτη εν δυνάμει ψεύτικο, μη ικανό να αποδείξει τα αυτονόητα. Αν κάποιος

εισβολέας εντοπίσει το υδατογράφημα οι πιθανότητες να μη καταφέρει να το αλλοιώσει είναι μηδενικές. Αν κάποιος φτιάξει μία καλή ποντικοπαγίδα, τότε κάποιος άλλος θα φτιάξει καλύτερο ποντίκι [24].

1.3.3 Θωράκιση

Υπάρχουν πολλές καταστάσεις κατά τις οποίες είναι επιθυμητό ένα λογισμικό να σταματήσει να λειτουργεί ορθά αν κάποιος το τροποποιήσει με οποιονδήποτε τρόπο. Το γεγονός αυτό ξεφεύγει από τα πλαίσια των τεχνικών της υδατογράφησης και της συσκότισης, δεν παύει όμως να είναι μία ανάγκη και συνάμα επιθυμία η οποία είναι απαραίτητη. Επομένως προκειμένου να γίνει πράξη, δημιουργήθηκε μία τεχνική η οποία ονομάζεται θωράκιση (tamperproofing) κατά την οποία αν ένα πρόγραμμα είναι υδατογραφημένο και ο κώδικας που χτίζει το υδατογράφημα αλλάξει, ή αν κάποιος ιός ηλεκτρονικών υπολογιστών προσκολλήσει σε ένα πρόγραμμα, ή ακόμα αν το τμήμα κώδικα το οποίο σχετίζεται με την προστασία ενός προγράμματος το οποίο παρέχεται διαδικτυακά σε χρήστες τροποποιηθεί το πρόγραμμα να παύσει να είναι λειτουργικό.

Στη γενική περίπτωση η θωράκιση χωρίζεται σε 2 διακριτές φάσεις, οι οποίες μπορούν και λαμβάνουν χώρα, μέσα στον κώδικα του θωρακισμένου λογισμικού, ή σε διαφορετικό λογισμικό αλλά στο ίδιο υπολογιστικό σύστημα ή ακόμα και σε διαφορετικό λογισμικό σε διαφορετικό υπολογιστικό σύστημα. Αξίζει να σημειωθεί επίσης πως οι φάσεις αυτές μπορούν να λάβουν χώρα πολύ εύκολα και σε οποιονδήποτε συνδυασμό των προηγουμένων (π.χ., η πρώτη φάση να πραγματοποιείται μέσα στον κώδικα του θωρακισμένου λογισμικού, ενώ η δεύτερη φάση να πραγματοποιείται μέσα σε διαφορετικό λογισμικό αλλά στο ίδιο υπολογιστικό σύστημα).

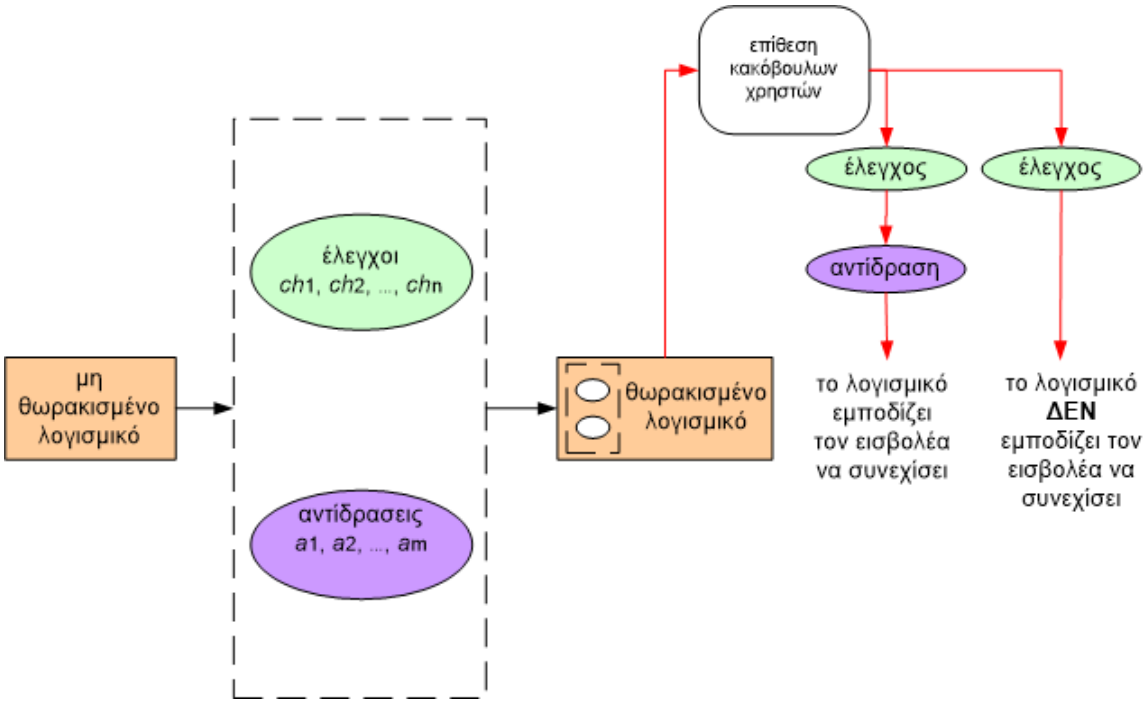
Η πρώτη εκ των 2 φάσεων αφορά στον έλεγχο πιθανής επέμβασης από κακόβουλους χρήστες, ενώ η δεύτερη αφορά στην αντίδραση που θα πρέπει να υιοθετήσει το λογισμικό αν ο έλεγχος της φάσης πρώτης αποτιμηθεί ως θετικός. Αμφότερα στις φάσεις αυτές προστίθεται επιπλέον κώδικας ο οποίος επιτελεί τις εκάστοτε λειτουργίες.

Στην πρώτη φάση, η οποία είναι η φάση του ελέγχου, ο έλεγχος μπορεί να λάβει χώρα με τους ακόλουθους τρόπους:

- Ελέγχοντας τον κώδικα του λογισμικού αυτόν καθ αυτόν (π.χ., μέσω μίας συνάρτησης κατακερματισμού).
- Ελέγχοντας τα παραγόμενα αποτελέσματα που η εκτέλεση του κώδικα δημιουργεί αν είναι ορθά (π.χ., σε ένα πρόγραμμα το οποίο εκτελεί ταξινομήσεις αν μετά από μία ακολουθία εισόδου οι πίνακες είναι ταξινομημένοι).
- Ελέγχοντας σε ποιο περιβάλλον πραγματοποιείται η εκτέλεση (π.χ., αν κατά την διάρκεια εκτέλεσής του έχει συνδεθεί στο πρόγραμμα κάποιος αποσφαλματωτής).

Τονίζεται πως ο τρίτος κατά σειρά έλεγχος είναι ιδιαίτερα επίπονος στην εφαρμογή του καθώς πρέπει να λάβει υπόψιν του πολλές και δύσκολα προσδιορίσιμες παραμέτρους. Επι-

πλέον ο κώδικας ο οποίος πρέπει να προστεθεί για να κάνει αυτόν τον έλεγχο δίνει εύκολα στόχο καθώς δεν έχει συνάφεια με τον υπόλοιπο κώδικα ό,τι και αν αυτός υλοποιεί.



Σχήμα 1.6: Μοντέλο συστήματος θωράκισης κώδικα.

Από την άλλη πλευρά, στη δεύτερη φάση της θωράκισης, την φάση της αντίδρασης οι ενέργειες οι οποίες μπορούν να πραγματοποιηθούν είναι οι ακόλουθοι:

- Τερματισμός της εκτέλεσης του λογισμικού σε εκείνο το σημείο του ελέγχου ή σε κάποιο άλλο σημείο στην μετέπειτα εκτέλεσή του προκειμένου να μην δοθεί κάποιο ερέθισμα / πληροφορία σε κακόβουλους χρήστες.
- Ανακατασκευή του κώδικα λογισμικού σε προηγούμενο ορθό στάδιο (π.χ., να εκτελείται κώδικας ο οποίος υπάρχει ως αντίγραφο ασφαλείας σε περιπτώσεις βλάβης).
- Χειραγώγηση της ροής εκτέλεσης του κώδικα προκειμένου να παράγει λανθασμένα αποτελέσματα (π.χ., να εκτελείται κώδικας ο οποίος εκ κατασκευής του είναι λανθασμένος).
- Μείωση της ταχύτητας εκτέλεσης του κώδικα (π.χ., να εισέρχεται σε κατάσταση αδράνειας για ορισμένο χρονικό διάστημα).
- Αναφορά της επίθεσης που υπέστη ο κώδικας (π.χ., μέσω του διαδικτύου).
- Καταστροφή αρχείων ζωτικής σημασίας του λογισμικού.

Στο σημείο αυτό αξίζει να σημειωθεί πως σε ένα λογισμικό θωρακίζοντάς το μπορούν να ενσωματωθούν πολλοί και διαφορετικοί έλεγχοι, όπως πολλές και διαφορετικές αντιδράσεις [24]. Αυτό εξαρτάται πρώτον, από τον παράγοντα της προστασίας που είναι επιθυμητή να δοθεί και δεύτερον, από τον παράγοντα διαφάνειας (stealthy) προς τους κακόβουλους χρήστες.

Όπως είναι αναμενόμενο, και στην τεχνική της θωράκισης υπάρχει ο ορισμός του βασικού προβλήματος. Έστω $I_d(P, E)$ και $I_a(P, E)$ να είναι κατηγορήματα εντός ενός προγράμματος P και ενός περιβάλλοντος E στο οποίο αυτά εκτελούνται. Τότε το P έχει θωρακιστεί επιτυχώς αν καθ όλη τη διάρκεια εκτέλεσής του το $I_d(P, E)$ κρατάει και το P έχει δεχθεί επίθεση επιτυχώς αν σε κάποιο σημείο κατά την διάρκεια εκτέλεσης το $I_a(P, E) \cup \neg I_d(P, E)$ κρατάει και δεν είναι ανιχνεύσιμο από το P. Σημειώνεται πως το I_a θα μπορούσε να σημαίνει πως το πρόγραμμα P μπορεί να τυπώσει ελεύθερα το ιδιωτικό (μυστικό) κλειδί και το I_d θα μπορούσε να σημαίνει πως το πρόγραμμα P δεν μπορεί να εκτελεστεί ούτε και να τυπώσει ελεύθερα το ιδιωτικό (μυστικό) κλειδί [24] [29].

Συνήθως η θωράκιση κώδικα λογισμικού δεν εφαρμόζεται μόνη της για προφανείς λόγους, γι αυτό συνδυάζεται ή με υδατογράφηση ή με συσκότιση. Στόχος της είναι να φυλά το των κώδικα του λογισμικού από τυχόν κακόβουλες ενέργειες, και όπως κάθε φύλακας δύσκολα μπορεί να κρυφτεί πάντα λειτουργεί σε συνεργασία με κάτι άλλο.

1.4 Ιστορική Αναδρομή

Τα πρώτα υδατογραφήματα ιστορικά ήταν σε χαρτί και έκαναν την εμφάνιση τους μαζί με την τέχνη της κατασκευής χειροποίητου χαρτιού περίπου 7 αιώνες πριν. Το παλαιότερο υδατογραφημένο κομμάτι χαρτιού που έχει βρεθεί, χρονολογείται περί το 1292 στην πόλη Fabriano της Ιταλίας, μια πόλη με μεγάλη επιρροή στην ανάπτυξη βιομηχανιών χαρτιού. Αναφορικά σημειώνεται πως στα τέλη του 13ου αιώνα, λειτουργούσαν περίπου 40 μύλοι χαρτιού όπου ο καθένας από αυτούς παρήγαγε χαρτί με διαφορετική μορφή, ποιότητα και τιμή. Το μειονέκτημα ήταν όμως πως το χαρτί που κατασκευάζονταν είχε τραχεία επιφάνεια και δεν ενδεικνύονταν για γραφή.

Η πρώτη ύλη χαρτιού επεξεργαζόταν από τεχνίτες της εποχής προκειμένου η επιφάνεια να αποκτούσε λεία υφή. Το επεξεργασμένο αυτό χαρτί γινόταν κατάλληλο για γράψιμο με αποτέλεσμα να μπορούσε να βγει στην αγορά πιο εύκολα. Ο ανταγωνισμός όμως τόσο μεταξύ των παραγωγών όσο και μεταξύ των εμπόρων ήταν πολύ υψηλός και δύσκολα μπορούσαν να κρατηθούν αρχεία πιστοποίησης για την προέλευση, την ποιότητα και τον τύπο του. Για τον λόγο αυτόν η υδατογράφηση επινοείται και εισάγεται με σκοπό να εξαλειφθεί οποιαδήποτε πιθανότητα σύγχυσης. Μετά την εφεύρεσή της, η υδατογράφηση διαδόθηκε αστραπιαία σε ολόκληρη την Ιταλία και την Ευρώπη. Αν και στην αρχή χρησιμοποιήθηκε μονάχα για την πιστοποίηση της φίρμας του εκάστοτε χαρτιού, με την πάροδο των χρόνων χρησιμοποιήθηκε και ως ένδειξη της ποιότητας, του τύπου, της χρονολόγησης και της αυθεντικότητας του.

1.4.1 Σκιαγράφηση Τεχνικών Υδατογράφησης Ψηφιακών Μέσων

Η υδατογράφηση ψηφιακών μέσων (digital watermarking) συμπεριλαμβάνει τις υδατογραφήσεις ψηφιακών εικόνων, βίντεο, σημάτων ήχου, λογισμικών και ψηφιακών κειμένων. Ό,τι επί της ουσίας μπορεί να ψηφιοποιηθεί, μπορεί να αποτελέσει και μέρος στο οποίο είναι δυνατόν να ενσωματωθεί πληροφορία και να εξαχθεί σε μελλοντικό χρόνο. Παρ όλα αυτά, οι τρόποι προσέγγισης κατά την εφαρμογή της υδατογράφησης είναι πολλοί και διάφοροι, εξαρτώμενοι πάντα από την σκοπιά με την οποία χειρίζεται η εκάστοτε τεχνική το υποψήφιο αντικείμενο (π.χ., μία τεχνική υδατογράφησης λογιστικού μπορεί να αντιλαμβάνεται το λογισμικό ως ένα σύνολο το οποίο περιέχει ως υποσύνολα ομάδες εντολών, ενώ μία άλλη ως ένα σύνολο το οποίο περιέχει ως υποσύνολα ροές εκτέλεσης).

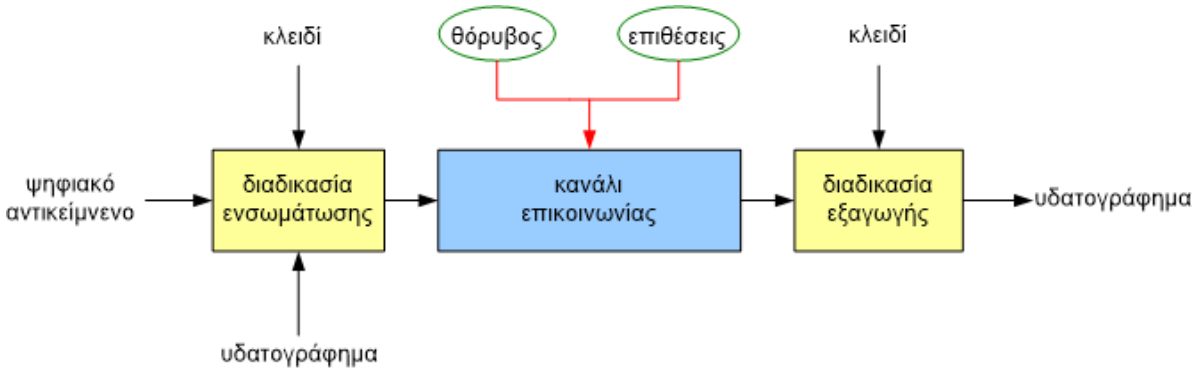
Όσο αναφορά τις εικόνες, τα βίντεο και τα σήματα ήχου οι τεχνικές οι οποίες έχουν αναπτυχθεί χωρίζονται σε 3 κύριες κατηγορίες. Η πρώτη περιλαμβάνει τις τεχνικές οι οποίες στηρίζονται στο χρόνο και στο χώρο πραγματοποιώντας τις αλλαγές απευθείας πάνω στο αντικείμενο προς υδατογράφηση (π.χ., τροποποιώντας τις φωτεινότητες των εικονοστοιχείων μίας εικόνας). Η δεύτερη περιλαμβάνει τις τεχνικές οι οποίες προκειμένου να εφαρμοστούν, πρέπει προηγουμένως να έχει επέλθει κάποιος μετασχηματισμός και ύστερα να εφαρμοστούν επάνω στον μετασχηματισμό αυτόν (π.χ., εφαρμόζοντας στην εικόνα τον μετασχηματισμό Fourier, ύστερα από την εφαρμογή αυτήν να λάβουν χώρα οι τροποποιήσεις οι οποίες είναι υπεύθυνες για την ενσωμάτωση της πληροφορίας επάνω στον Fourier και εν τέλει να γίνει ο μετασχηματισμός του αντιστρόφου Fourier προκειμένου η εικόνα να έρθει στην αρχική της μορφή αλλά προστατευμένη πλέον). Ενώ η τρίτη κατά σειρά περιλαμβάνει τις τεχνικές οι οποίες εφαρμόζονται επάνω στην συμπιεσμένη μορφή των αρχικών δεδομένων χρησιμοποιώντας ορισμένους από τους αλγορίθμους συμπίεσης (π.χ., JPEG2000 για τις εικόνες και MPEG - 1 layer - 3 το γνωστό MP3 για τα ηχητικά σήματα).

Στο σημείο αυτό αξίζει να σημειωθεί πως η τρίτη κατηγορία είναι ιδιαίτερα δημοφιλής στα βίντεο καθώς αυτά στις περισσότερες των περιπτώσεων αυτά είναι διαθέσιμα μονάχα σε συμπιεσμένη μορφή, για λόγους χωρητικότητας. Επομένως, από τη στιγμή όπου το κόστος για να γίνει η αποσυμπίεση με σκοπό να ενσωματωθεί το υδατογράφημα και μετά να γίνει και πάλι συμπίεση, για να μπορεί να διανεμηθεί να είναι φοβερά υψηλό [40] με αποτέλεσμα να αποφεύγεται.

Συνεχίζοντας με τα λογισμικά, οι τεχνικές υδατογράφησης οι οποίες έχουν προταθεί είναι πάρα πολλές και ορισμένες εξ αυτών εντελώς διαφορετικές μεταξύ τους. Αυτό είναι απολύτως δικαιολογημένο καθώς το λογισμικό είναι ένα αντικείμενο ιδιαίτερα ρευστό και σύνθετο, αποτελούμενο από διαφορετικές οντότητες, το οποίο μπορεί να κωδικοποιηθεί σε ποικίλες γλώσσες προγραμματισμού όπου η κάθε μία φέρει τις δικές της ιδιομορφίες. Θέλοντας όμως να δοθεί μία πρώτη αίσθηση των κατηγοριών αυτών αναφέρονται 3 κατηγορίες οι οποίες έχουν δώσει πολλά και πρόσφορα κίνητρα για περαιτέρω έρευνα στον τομέα προστασίας λογισμικού και γενικά των πνευματικών δικαιωμάτων.

Η πρώτη κατηγορία περιλαμβάνει τις τεχνικές οι οποίες αντιμετωπίζουν το λογισμικό ως ένα σύνολο ροών εκτέλεσης και κωδικοποιούν μέσα σε αυτό διάφορα είδη γραφημάτων. Η δεύτερη κατηγορία περιλαμβάνει τις τεχνικές οι οποίες εστιάζουν στις συνθήκες ελέγχου και

στις συνθήκες επανάληψης που εμπεριέχονται σε ένα λογισμικό κωδικοποιώντας δυαδικές ακολουθίες σε αυτές, ενώ η τρίτη κατηγορία περιλαμβάνει τις τεχνικές οι οποίες κωδικοποιούν το υδατογράφημα στη συμπεριφορά και στον τρόπο λειτουργίας των νημάτων τα οποία δημιουργούνται κατά την διάρκεια εκτέλεσης του λογισμικού.



Σχήμα 1.7: Γενικό μοντέλο ψηφιακού συστήματος υδατογράφησης.

Κλείνοντας με τα ψηφιακά κείμενα, σημειώνεται πως και εδώ έχουν προταθεί διάφορες τεχνικές υδατογράφησης, όχι σε τόσο ευρύ φάσμα όπως των προηγουμένων αντικειμένων, όλες όμως είναι ιδιαίτερα ευάλωτες σε επιθέσεις κακόβουλων χρηστών καθώς το κείμενο αποτελείται μονάχα από χαρακτήρες με αποτέλεσμα πολύ δύσκολα να ενσωματώνεται κάτι το οποίο δεν είναι ορατό, δίνοντας έτσι στόχο σε κακόβουλους χρήστες, με αποτέλεσμα η προστασία τους να αλλοιώνεται εύκολα και αβίαστα. Εστιάζοντας στις τεχνικές που έχουν αναπτυχθεί για το ψηφιακό κείμενο γίνεται φανερό πως αυτές χωρίζονται σε 2 κατηγορίες. Είτε πράττουν αλλαγές απευθείας πάνω σε αυτό (π.χ., αλλάζοντας τις αποστάσεις μεταξύ γραμμάτων, λέξεων και γραμμών τροποποιώντας το αρχείο κεφαλίδων του κειμένου αν το κείμενο είναι σε μορφή κατάλληλη όπως PDF ή αν δεν υπάρχει τέτοιο αρχείο οι τροποποιήσεις λαμβάνουν χώρα απευθείας πάνω στο κείμενο) είτε πράττουν αλλαγές σε μετασχηματισμένες μορφές του (π.χ., το τροποποιούν το κείμενο και το αντιμετοπίζουν ως εικόνα ή ως ψηφιακό σήμα εξελισσόμενο στο χρόνο) [58].

1.4.2 Εργασίες σε Υδατογράφηση Λογιμσικού

Οι υδατογραφήσεις λογισμικού έκαναν αισθητή την εμφάνισή τους από τα τέλη της δεκαετίας του 90. Από τότε, όπου η προσφορά και η ζήτηση στην αγορά λογισμικών άρχισαν να αυξάνουν με υψηλούς ρυθμούς δίνοντας κίνητρο στις βιομηχανίες να δημιουργούν όλο και περισσότερα λογισμικά προκειμένου να καλύπτουν είτε βασικές, είτε πλασματικές ανάγκες. Με την πάροδο του χρόνου λοιπόν προτείνονται και αξιολογούνται σε ερευνητικές εργασίες, τεχνικές υδατογράφησης με απώτερο σκοπό να προσφέρουν ποικίλα επίπεδα προστασίας στο λογισμικό.

Στο σημείο αυτό, έχοντας ως βάση τις εργασίες που έχουν δημοσιευτεί από τα τέλη της δεκαετίας του 90 έως και 2 δεκαετίες μετέπειτα στοχεύοντας να δοθεί μία σχετική εικόνα του κλίματος που έχει επικρατήσει στο τομέα υδατογράφησης λογισμικού, παρακάτω σκιαγραφείται η ροή εξέλιξης ενός αντιπροσωπευτικού συνόλου αυτών.

Στα τέλη της δεκαετίας του 90 και συγκεκριμένα το έτος 1998, οι C. Collberg και C. Thomborson δημοσιεύουν μία εργασία τους με τίτλο: On the limits of software watermarking. Η εργασία αυτή επί της ουσίας ήταν χωρισμένη σε δύο κύρια μέρη. Στο πρώτο οι συγγραφείς της, στηριζόμενοι σε μία από τις ιδιότητες των υδατογραφημάτων που σχετίζεται με τον τρόπο με τον οποίο αυτά εξάγονται από τον κώδικα του λογισμικού, κατηγοριοποιούσαν τα υδατογραφήματα σε στατικά και σε δυναμικά. Στο δεύτερο μέρος οι συγγραφείς είχαν στηριχτεί στο γεγονός πως η κατηγορία των δυναμικών υδατογραφημάτων αντιστέκεται αποτελεσματικότερα σε επιθέσεις κακόβουλων χρηστών από ότι η κατηγορία των στατικών εκ κατασκευής τους, καθώς δημιουργούνται κατά την διάρκεια εκτέλεσης του κώδικα και είχαν προτείνει δυναμικά υδατογραφήματα βασισμένα σε γραφοθεωριτικές προσεγγίσεις (graph theoretical approaches). Συγκεκριμένα η πρότασή τους εστίαζε σε δύο τύπους γραφημάτων που κωδικοποιούσαν έναν αριθμό n οι οποίοι αφού ενσωματωνόντουσαν στον κώδικα, εξάγονταν από αυτόν επεξεργάζοντας καταλλήλως τις δομές δεδομένων που είχαν δημιουργηθεί κατά την διάρκεια εκτέλεσής του.

Η υδατογράφηση πραγματοποιούνταν σε δύο κύριες φάσεις (την φάση κατά την οποία κατασκευάζονταν το υδατογράφημα και την φάση κατά την οποία αυτό ενσωματώνονταν στον κώδικα). Αρχικά μετατρέπονταν ένας ακέραιος αριθμός σε μία δομή η οποία δημιουργούταν είτε με κωδικοποίηση $radixk$ (κυκλική λίστα που κωδικοποιούσε έναν ακέραιο αριθμό - πρώτη δομή) είτε, με κωδικοποίηση $enumeration$ (δομή δέντρου που κωδικοποιούσε έναν ακέραιο αριθμό - δεύτερη δομή). Μετέπειτα κάθε ένα συστατικό (στοιχείο λίστας ή στοιχείο δέντρου) της δομής υδατογράφησης ενσωματώνονταν στο λογισμικό, σε συγκεκριμένα σημεία. Τα σημεία ενσωμάτωσης ήταν άρρηκτα συνδεδεμένα με μία συγκεκριμένη ακολουθία εισόδου, η οποία αποτελούσε και το κλειδί υδατογράφησης.

Από την άλλη πλευρά, την πλευρά της εξαγωγής του υδατογραφήματος από το λογισμικό, προκειμένου να εξαχθεί το ενσωματωμένο υδατογράφημα από τον κώδικα, δίνονταν η συγκεκριμένη ακολουθία κλειδί ως είσοδος στο λογισμικό με σκοπό κατά της διάρκεια εκτέλεσης να ενεργοποιηθούν και να δημιουργηθούν με την σωστή σειρά τα όλα τα συστατικά του υδατογραφήματος. Σημειώνεται πως ο πληθάριθμος των στοιχείων $(I_1, I_2, \ldots, I_n)$ της ακολουθίας κλειδί ήταν ίσος με τον πληθάριθμο των συστατικών του υδατογραφήματος $(w_1, w_2, \ldots, w_n)$.

Το υδατογράφημα κατασκευάζονταν τμηματικά στο ίχνος (trace) που δημιουργούσε το λογισμικό κατά την διάρκεια εκτέλεσής του, είτε με την βοήθεια επιπρόσθετων εντολών είτε με την βοήθεια χειραγώγησης των διευθύνσεων μνήμης. Ενώ εξάγονταν από αυτό, δίνοντας στο λογισμικό το κλειδί και παρατηρώντας εν συνεχεία ή ορισμένες ιδιότητες του ίχνους που δημιουργούσε η εκτέλεση ή την ακολουθία των τελεστών που εκτελούνταν. Εν κατακλείδι, προκειμένου να ολοκληρώσουν την εργασία τους οι C. Collberg και C. Thomborson είχαν παρουσιάσει ποικίλες επιθέσεις που θα μπορούσε να δεχθεί το υδατογράφημα που είχαν

προτείνει, εξάγοντας από αυτές ενδιαφέροντα συμπεράσματα για την αποδοτικότητα της τεχνική τους [27].

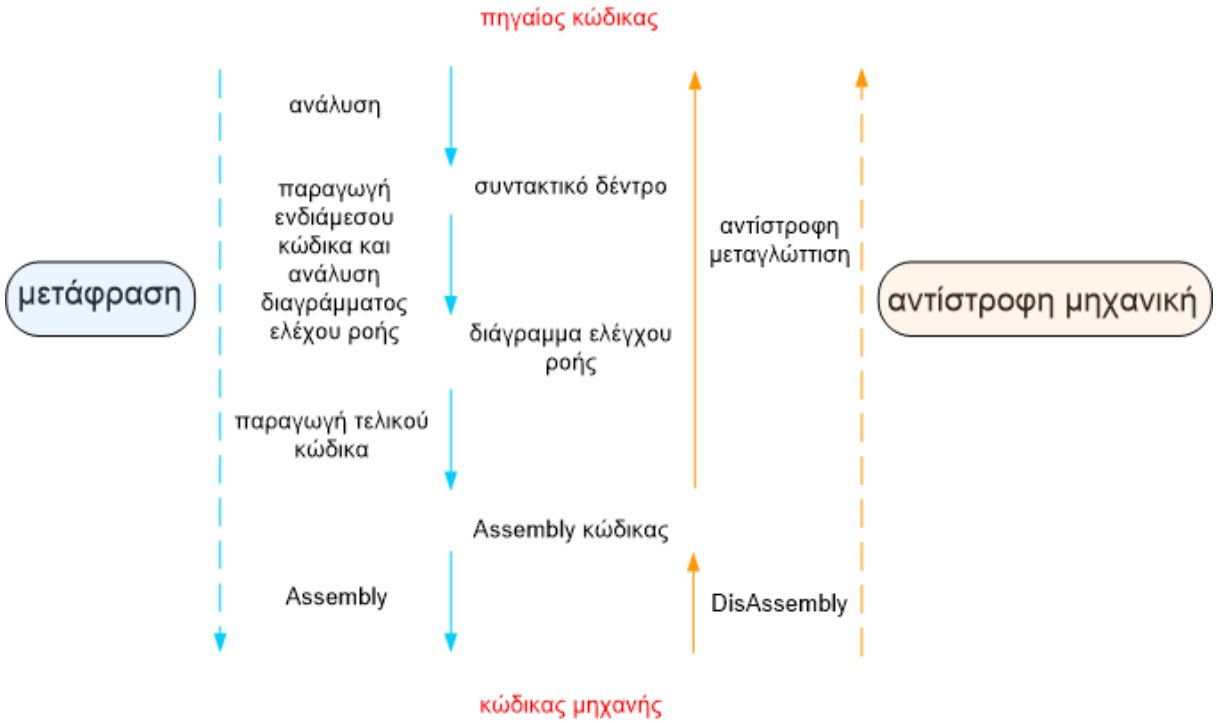
Ένα χρόνο αργότερα, το έτος 1999 οι ίδιοι συγγραφείς δημοσιεύουν άλλη μία εργασία με τίτλο: Software watermarking: models and dynamic embeddings. Στην εργασία αυτή είχαν επαναδιατυπώσει πιο δομημένα έννοιες και ορισμούς που είχαν εισάγει στην προηγούμενή τους εργασία καθώς επίσης είχαν επεκτείνει (αυτό ήταν και η συνεισφορά τους) και το τυπικό μοντέλο υδατογράφησης λογισμικού τους συμπεριλαμβάνοντας μετρικές σύμφωνα με τις οποίες μπορούσε να προσεγγιστεί σε καλό βαθμό η αποτελεσματικότητά του [28]. Οι μετρικές αυτές ύστερα από την παρουσίασή τους καθιερώθηκαν στον τομέα υδατογράφησης λογισμικού, ως βασικά στοιχεία αξιολόγησης τεχνικών και από άλλους ερευνητές.

Με την είσοδο στην καινούρια χιλιετία και αφού έχουν περάσει πέντε χρόνια από τη δημιουργία της γλώσσας προγραμματισμού Java, από τον James Gosling, άρχιζαν να έρχονται στην επιφάνεια όλο και περισσότερες ενδιαφέρουσες εργασίες προκειμένου να προτείνουν τεχνικές υδατογράφησης σε λογισμικά κωδικοποιημένα στη γλώσσα αυτή [42]. Ο λόγος για τον οποίο γίνονταν και συνεχίζει ακάθεκτα να γίνεται έως και σήμερα αυτό (να δημοσιεύονται εργασίες για υδατογράφηση σε λογισμικά γραμμένα σε γλώσσα Java), είναι το γεγονός πως η γλώσσα αυτή είναι κατασκευασμένη με τέτοιον τρόπο έτσι ώστε να καθίσταται ανεξάρτητη πλατφόρμας λογισμικού. Με απλά λόγια αν έχει γραφτεί ένα λογισμικό στη γλώσσα αυτή, έχει μεταγλωττιστεί σε μία μηχανή (π.χ., Sun) με ένα λογισμικό (π.χ., Linux) και το εκτελέσιμο αρχείο που έχει δημιουργηθεί μετά την μεταγλώττιση μπορεί και εκτελείται σε αυτό, τότε αυτό το εκτελέσιμο αρχείο θα εκτελείτε και σε οποιοδήποτε άλλο μηχάνημα με οποιοδήποτε λογισμικό (Write Once Run Anywhere). Η βασική προϋπόθεση βέβαια είναι πως και το ένα και το άλλο μηχάνημα θα πρέπει να έχουν εγκατεστημένο τον μεταγλωττιστή (compiler) της Java.

Επομένως η προκείμενη γλώσσα στηριζόμενη στο γεγονός ότι αποφεύγονταν οι ασυμβατότητες μεταξύ ηλεκτρονικών υπολογιστών διαδόθηκε πολύ σύντομα και χρησιμοποιείται έως και τις μέρες μας ευρέως σε πληθώρα εφαρμογών. Βέβαια το χαρακτηριστικό της ανεξαρτησίας έχει και τα μειονεκτήματά του, καθώς ο εκτελέσιμος κώδικας γίνεται ευάλωτος στην αντίστροφη μεταγλώττιση (μεταφορά από τον εκτελέσιμο κώδικα στον πηγαίο) γεγονός το οποίο χρειάζεται μεγάλη προσοχή κατά την ενσωμάτωση της δομής υδατογράφησης (να μην γίνεται ευκόλως ορατή σε κακόβουλους χρήστες κοιτάζοντας τον πηγαίο κώδικα).

Επομένως μεταβαίνοντας στο έτος 2000, δημοσιεύεται η εργασία των A. Monden, H. Iida, K. Matsumoto, K. Inoue και K. Torii με τίτλο: A practical method for watermarking Java programs. Στην εργασία αυτή είχε προταθεί μία μέθοδος σύμφωνα με την οποία, αποθαρρύνονταν η κλοπή λογισμικού το οποίο ήταν κωδικοποιημένο σε γλώσσα Java, ενσωματώνοντας σε αυτό ψηφιακό υδατογράφημα. Οι συγγραφείς αυτό που είχαν προτείνει στην ουσία, ήταν μία μέθοδος η οποία ενσωμάτωνε μέσα σε ένα λογισμικό τα πνευματικά δικαιώματα αυτών που το είχαν δημιουργήσει, με την μορφή υδατογραφήματος μέσα σε Java κλάσεις. Απώτερος σκοπός της ιδέας αυτής ήταν να εξασφαλίζεται η νομική ιδιοκτησία των κλάσεων αυτών. Επομένως κάνοντας χρήση της μεθόδου αυτής, μπορούσε

να καταστεί δυνατή η αναγνώριση ενός παράνομου λογισμικού το οποίο περιέχει στον κώδικά του υποκλεμμένες κλάσεις.



Σχήμα 1.8: Τα διαφορετικά στάδια της μεταγλώττισης και της αντίστροφης μηχανικής.

Συγκεκριμένα οι συγγραφείς είχαν χωρίσει την τεχνική υδατογράφησής τους στις ακόλουθες φάσεις:

- Εισαγωγή κίβδηλων μεθόδων(dummy method injection) οι οποίες πότε δεν εκτελούνταν μέσα στο λογισμικό, νεκρός κώδικας (π.χ., στο εσωτερικό συνθηκών ελέγχου με τις εντολές: $if(condition)\ call\ dummy\ method();$).
- Μεταγλώττιση του τροποποιημένου πλέον πηγαίου κώδικα με στόχο την παραγωγή εκτελέσιμου κώδικα με ενσωματωμένο το υδατογράφημα.
- Εξαγωγή του ενσωματωμένου υδατογραφήματος από το υδατογραφημένο λογισμικό.

Προκειμένου οι συγγραφείς της εργασίας να αποδείξουν την αποτελεσματικότητας της μεθόδου τους, είχαν πραγματοποιήσει και πειράματα σύμφωνα με τα οποία 20 από τα 23 ενσωματωμένα υδατογραφήματα μέσα σε κλάσεις παρέμειναν αναλλοίωτα ύστερα από την εφαρμογή δύο ειδών επιθέσεων. Στην πρώτη επίθεση είχαν εφαρμοστεί κάποιοι από τους μετασχηματισμούς συσκότισης (σε χαμηλό επίπεδο κώδικα), με στόχο να στρεβλώσουν τον κώδικα του εκτελέσιμου αρχείου, ενώ στη δεύτερη είχε γίνει αντίστροφη μεταγλώττιση και

εν συνεχεία ξανά μεταγλώττιση (decompile recompile) του εκτελέσιμου αρχείου με στόχο αυτήν την φορά την στρέβλωση του πηγαίου κώδικα (σε υψηλό επίπεδο κώδικα) [47].

Συνεχίζοντας στην καινούρια χιλιετία και μάλιστα στο ίδιο έτος, δημοσιεύτηκε και η εργασία των J. Stern, G. Hachez, F. Keoune και J.Quisquater με τίτλο: Robust object watermarking: application to code. Στην εργασία αυτή είχε γίνει έρευνα πάνω σε ένα στάδιο της διαδικασίας υδατογράφησης το οποίο είχε αγνοηθεί έως τότε. Συγκεκριμένα οι συγγραφείς είχαν βασιστεί στο γεγονός πως ήταν χρονοβόρο και επίπονο στο επίπεδο του κώδικα μηχανής να μετατραπούν / επεξεργαστούν / στρεβλωθούν δεδομένα. Με αποτέλεσμα η δημιουργία μίας εύρωστης τεχνικής ενσωμάτωσης υδατογραφήματος στο χαμηλό αυτό επίπεδο να θεωρείτο μία αποτελεσματική λύση, καθώς δύσκολα θα πραγματοποιούσε κάποιος εισβολέας επιθέσεις στο επίπεδο αυτό.

Έχοντας ως υπόθεση το γεγονός αυτό, είχαν προταθεί απλές μετατροπές δεδομένων (προσθήκη και αφαίρεση εντολών αντικαθιστώντας ομάδες εντολών από σημασιολογικά και προγραμματιστικά ισοδύναμες), προκειμένου να ενσωματώσουν το υδατογράφημα σε χαμηλό επίπεδο χωρίς να αλλοιώσουν την λειτουργικότητα του λογισμικού. Συγκεκριμένα τα στάδια της τεχνικής σύμφωνα με την οποία γίνονταν η ενσωμάτωση ήταν τα εξής:

- Δήλωση ενός συνόλου στο οποίο θα εμπεριέχονται όλες οι διαφορετικές ομάδες εντολών που βρίσκονται στον κώδικα.
- Αρχικοποίηση μίας μεταβλητής, έστω δ $(0 \leq \delta \leq 1)$, η οποία θα παίζει το ρόλο του ορίου εντοπισμού σήμανσης (mark) μέσα στον κώδικα του λογισμικού.
- Για κάθε ομάδα του συνόλου υπολογίζεται η συχνότητα εμφάνισής της μέσα στον κώδικα του λογισμικού.
- Κατασκευή με βάση τα αποτελέσματα των συχνοτήτων, ενός πινάκα έστω c, ο οποίος θα περιέχει τις συχνότητες εμφάνισης των ομάδων αυτών $(c_1, c_2, \ldots, c_n)$.
- Κατασκευή ενός πίνακα έστω w, ο οποίος θα είναι ίδιου μεγέθους με τον c και θα περιέχει στοιχεία (αριθμούς) τυχαία κατανεμημένους, σύμφωνα με συγκεκριμένους περιορισμούς $(w_1, w_2, \ldots, w_n)$.
- Με βάσει τους πίνακες c και w γίνεται η υδατογράφηση του λογισμικού, μετατρέποντας τις ήδη υπάρχουσες εντολές (ομάδες εντολών) με το τύπο: $c' = c + w$.

Οι εν λόγω μετατροπές από την μία πλευρά δυσκόλευαν τον κακόβουλο χρήστη να εφαρμόσει τεχνικές με στόχο εντοπίσει και να εξάγει το υδατογράφημα από το γεγονός ότι οι αλλαγές είχαν γίνει στο επίπεδο του κώδικα μηχανής, ενώ από την άλλη βοηθούσαν αυτόν που υδατογραφεί το λογισμικό να κάνει την ενσωμάτωση καθώς προσέθετε ή αφαιρούσε απλές εντολές χωρίς να αλλοιώνει την λειτουργικότητα του λογισμικού μετατρέποντας απλά τις συχνότητες εμφάνισες.

Επιπλέον οι συγγραφείς είχαν παρουσιάσει και ένα νέο παράδειγμα εξαγωγής πίνακα (vector extracting paradigm - VEP) στο οποίο εφάρμοσαν την παραπάνω τεχνική. Το

παράδειγμα αυτό ήταν ένα σημαντικό βήμα μεταξύ των ακατέργαστων ψηφιακών δεδομένων και του αντικειμένου στο οποίο λάμβανε χώρα η σήμανση. Η τεχνική της προκειμένης εργασίας διαφοροποιούνταν από προηγούμενες δημοσιευμένες εργασίες καθώς λάμβανε χώρα στο επίπεδο του κώδικα μηχανής, χειρίζοντάς τον ως στατιστικό αντικείμενο. Αυτό προέκυπτε από το γεγονός ότι η τεχνική υδατογράφησης λογισμικού στηρίζονταν στις συχνότητες εμφάνισης εντολών του κώδικα. Σημειώνεται επίσης πως ακόμη και αν ένας κακόβουλος χρήστης άλλαζε (π.χ., προσέθετε ή αφαιρούσε) κάποιες εντολές με αποτέλεσμα να μεταβάλει την συχνότητα εμφάνισής τους μέσα στον κώδικα μηχανής (π.χ., πρόσθετε απλά εντολές όπου δεν κάνουν τίποτα όπως η nop), τότε αν η αλλαγή ήταν μικρότερη από το όριο εντοπισμού δ το υδατογράφημα συνέχιζε να καθίσταται εντοπίσιμο. Εν τούτης ο καθορισμός του εν λόγω ορίου έχριζε ιδιαίτερης προσοχής (αποτελούσε ένα trade off) καθώς η τιμή που έπρεπε κάθε φορά να λάβει ήταν ανάλογη του εκάστοτε κώδικα και του εκάστοτε υδατογραφήματος [59].

Επίσης στο ίδιο έτος, δημοσιεύτηκε και η εργασία των J. Palsberg, S. Krishnaswamy, M. Kown, D. Ma, Q. Shao και Y. Zhang με τον τίτλο: Experience with software watermarking. Στην προκειμένη εργασία οι συγγραφείς είχαν εφαρμόσει μία τεχνική υδατογράφησης, από μία προηγούμενη εργασίας των C. Collberg και C. Thomborson, πάνω σε λογισμικό κωδικοποιημένο σε γλώσσα Java. Συγκεκριμένα είχαν παρουσιάσει την ενσωμάτωση του υδατογραφήματος, την εξαγωγή του καθώς και πειράματα αποτελέσματα επιθέσεων, με σκοπό να δείξουν πόσο αποτελεσματική ήταν η προστασία λογισμικού με την τεχνική αυτή.

Σύμφωνα με την τεχνική αυτή η ενσωμάτωση του υδατογραφήματος λάμβανε χώρα επιλέγοντας ήδη υπάρχουσες κλάσεις από τον κώδικα του λογισμικού, μετατρέποντάς τες σε κόμβους κλάσεις απλά προσθέτοντας κάποια επιπλέον πεδία (fields) σε αυτές. Κάθε ένα από τα επιπρόσθετα πεδία κρατούσε μία εξερχόμενη ακμή σε έναν άλλο κόμβο του αντικειμένου / υδατογραφήματος αναθέτοντας απλά στο πεδίο την κατάλληλη τιμή. Με τον τρόπο αυτόν ο κόμβος κλάση γίνονταν ένα δομικό στοιχείο για την κατασκευή του υδατογραφήματος (planted plane cubic tree - PPCT) και άτυπα παράγονταν κώδικας ευθείας γραμμής (straight line code) με απώτερο σκοπό να κατασκευάσει / συνδέσει κατάλληλα το PPCT (γράφημα το οποίο κωδικοποιεί έναν αριθμό) με το υπόλοιπο μη υδατογραφημένο λογισμικό [51].

Ένα χρόνο αργότερα και συγκεκριμένα το έτος 2001, δημοσιεύτεαι άλλη μία εργασία των C. Collberg, S. Jha, D. Tomko και H. Wang με τίτλο: UWStego: A general architecture for software watermarking. Στην εν λόγω εργασία οι συγγραφείς είχαν παρουσιάσει ποικίλες τεχνικές υδατογράφησης λογισμικού καθώς και μετρικές σύμφωνα με τις οποίες αξιολογήθηκαν εν συνεχεία οι τεχνικές αυτές. Επιπλέον είχαν παρουσιάσει τον σχεδιασμό και την εφαρμογή μίας αρχιτεκτονικής ονόματι *UWStego*, έχοντας ως πρωτεύοντα στόχο τη δυναμική υδατογράφηση λογισμικών γραμμένων στην γλώσσα προγραμματισμού Java [30].

Συγκεκριμένα η συνολική συνεισφορά της εργασίας αυτής στον τομέα υδατογραφημάτων αλλά και γενικά στον τομέα προστασίας λογισμικού ήταν η εξής:

- Ερευνούσε και κατηγοριοποιούσε τις ήδη υπάρχουσες ως τότε τεχνικές στον τομέα

υδατογράφησης λογισμικού, αναφέροντας σε κάθε μία τις αδυναμίες της. Επιπλέον μελετούσε ποικίλα μοντέλα απειλών γύρο από τα συστήματα υδατογραφημάτων λογισμικού.

- Παρουσίασε μετρικές προκειμένου να προσεγγιστεί η αποτελεσματικότητα μοντέλων υδατογραφημάτων λογισμικού που είχαν παρουσιαστεί ως τότε, από διάγορες οπτικές γωνίες αλλά και να χρησιμοποιηθούν ως βάση των μεταγενέστερων.

- Πρότεινε μία αρχιτεκτονική, ονόματι *UWStego*, για υδατογράφηση λογισμικών σε Java. Τα συστατικά της αρχιτεκτονικής αυτής είχαν σχεδιαστεί, βάσει τις αρχές της απόκρυψης πληροφορίας. Επίσης είχε γίνει αναφορά στο τρόπο με τον οποίον μπορούσε να χρησιμοποιηθεί η εν λόγω τεχνική προκειμένου να αυξηθεί η ανθεκτικότητα και η δυσκολία εντοπισμού του υδατογραφήματος.

Το επόμενο έτος και συγκεκριμένα το έτος 2002, έρχονται στην επιφάνεια άλλες δύο εργασίες σχετικές με υδατογράφηση και προστασία λογισμικού. Οι συγγραφείς της πρώτης κατά σειρά εργασίας ήταν οι J. Nagra, C. Collberg και C. Thomborson με τίτλο: A functional taxonomy for software watermarking και της δεύτερης ήταν οι C. Collberg και C. Thomborson με τίτλο: Watermarking, tamper-proofing, and obfuscation tools for software protection. Οι εργασίες αυτές είχαν ως απώτερο σκοπό να συγκεντρώσουν και να ομαδοποιήσουν ήδη υπάρχουσες πληροφορίες στον τομέα της υδατογράφησης άλλα και στον ευρύτερο τομέα της προστασίας λογισμικού. [49] [29]

Στην πρώτη κατά σειρά εργασία, οι συγγραφείς J. Nagra, C. Collberg και C. Thomborson είχαν συγκεντρώσει πληροφορίες, οι οποίες είχαν ήδη δημοσιευτεί στο παρελθόν, γύρω από τα υδατογραφήματα λογισμικού κατηγοριοποιώντας τα υδατογραφήματα σε επιμέρους κατηγορίες βάσει του τρόπου με τον οποίον ενσωματώνονται στο λογισμικό. Στην δεύτερη κατά σειρά εργασία, οι συγγραφείς οι οποίοι ήταν και πάλι ο C. Collberg και C. Thomborson, (αυτήν την φορά χωρίς την συμβολή του J. Nagra) και αυτήν την φορά είχαν συγκεντρώσει πληροφορίες γενικά για την προστασία του λογισμικού συμπεριλαμβανομένου της υδατογράφησης, δίνοντας με τον τρόπο αυτό έναυσμα για έρευνα σε ανεξερεύνητες πτυχές της προστασίας λογισμικού. Συγκεκριμένα, είχαν προτείνει / περιγράψει τις εξής τεχνικές προστασίας:

- Υδατογράφηση λογισμικού με σκοπό να προστατευτούν τα πνευματικά δικαιώματα από ξένες ιδιοποιήσεις.

- Συσκότιση λογισμικού με στόχο να δυσκολευτούν οι κακόβουλοι χρήστες να κατανοήσουν των κώδικα και να πάρουν με αντίστροφη μεταγλώττιση στα χέρια τους τον πηγαίο κώδικα.

- Θωράκιση λογισμικού με σκοπό να τεθεί ένα εμπόδιο στην εκτέλεση του κώδικα μετά από κάποια ενδεχόμενη τροποποίησή του.

Εν ολίγης είχαν προτείνει προστασία λογισμικού με τρεις διαφορετικούς και ταυτόχρονα αλληλένδετους τρόπους, με στόχο την αποτελεσματικότερη προστασία από τυχόν επιθέσεις

κακόβουλων χρηστών. Οι συγγραφείς προκείμενου να καλύψουν ενδεχόμενες ασάφειες και κενά σχετικά με τις τεχνικές αυτές είχαν αναφέρει σε κάθε μία από αυτές κάποια αντιπροσωπευτικά παραδείγματα και πιθανές απειλές που θα μπορούσαν να τις αλλοιώσουν.

Ένα χρόνο αργότερα το έτος 2003, δημοσιεύεται η εργασία των D. Couran, N. Hurley και M. Cinneide με τίτλο: Securing Java through software watermarking. Στην εργασία αυτή είχε προταθεί ένα καινούριο μοντέλο υδατογράφησης λογισμικού, του οποίου η πηγή έμπνευσης ήταν η θεωρία εντοπισμού σήματος καθώς επίσης και μία ήδη δημοσιευμένη εργασία των J. Stern, G. Hachez, F. Keoune και J.Quisquater. Συγκεκριμένα η ιδέα ήταν να εξαχθεί ένας πίνακας έστω r με N διαστάσεις από ένα λογισμικό s, κάνοντας χρήση μίας συνάρτησης εξαγωγής, έστω $X(s) = r$ (η διαδικασία εξαγωγής του πίνακα r είχε παρουσιαστεί ήδη σε προηγούμενη εργασία).

Συγκεκριμένα προκειμένου να εφαρμόσουν την ιδέα τους, οι συγγραφείς είχαν προτείνει την αποθήκευση διακριτών σημείων στο βάθος κλήσεων που εδημιουργείτο σε ένα λογισμικό κατά την διάρκεια εκτέλεσής του, έχοντας δοθεί σε αυτό μία συγκεκριμένη ακολουθία εισόδου. Ύστερα προκειμένου να αναγνωριστούν οι εν λόγω κλήσεις στις μεθόδους κατά την διάρκεια εκτέλεσης του λογισμικού, είχε γίνει χρήση αποσφαλματωτή (debugger). Εν συνεχεία στοχεύοντας στην ενίσχυση προστασίας του πίνακα r, είχε γίνει μετατροπή του πίνακα αυτού σε μία μετάθεση r' χρησιμοποιώντας ένα κλειδί μετάθεσης το οποίο ήταν γνωστό και κατά την διαδικασία της ενσωμάτωσης και κατά την διαδικασία της εξαγωγής [32]. Ο τρόπος σύμφωνα με τον οποίον είχε γίνει η εν λόγω μετατροπή είχε στηριχτεί στο ίδιο σκεπτικό με αυτό που είχε ήδη είχε λάβει χώρα σε μία προηγούμενη εργασία ($r' = r+w$, όπου ο πίνακας w ήταν ένα ψευδοτυχαίο υδατογράφημα σήματος / πίνακας ίδιας διάστασης με τον r. Οι δύο αυτοί πίνακες r, w προστήθονταν και έδιναν ως τελικό αποτέλεσμα έναν καινούριο πίνακα έστω r' σύμφωνα με τον οποίον θα πραγματοποιούνταν η ενσωμάτωση).

Ύστερα από έναν χρόνο το έτος 2004 είχε παρουσιαστεί η μεταπτυχιακή εργασία της S. Thaker με τίτλο: Software watermarking via Assembly code transformations. Στην εργασία αυτή είχε προταθεί ένα μοντέλο υδατογράφησης λογισμικού βασιζόμενο σε μετασχηματισμούς στο επίπεδο της γλώσσας Assembly. Σημειώνεται πως η τεχνική αυτής της υδατογράφησης ήταν εμπνευσμένη από τους μεταμορφικούς ιούς σε ηλεκτρονικούς υπολογιστές.

Αυτού του είδους οι ιοί μεταφέρονται σε ξεχωριστό τμήμα κώδικα του λογισμικού κάθε φορά που αντιγράφονται. Αυτό κάνει τον εντοπισμό τους εξαιρετικά δύσκολη υπόθεση, καθώς από τη στιγμή που δεν υπάρχουν σταθερά σημάδια τα οποία να δίνουν κάποια ερεθίσματα / σήματα σε λογισμικά εντοπισμού ιών, είναι εξαιρετικά δύσκολο να εντοπιστούν. Από την σκοπιά των υδατογραφημάτων λογισμικού τώρα ο μεταμορφισμός επιτρέπει στο υδατογράφημα να ενσωματώνεται με έναν μοναδικό τρόπο σε κάθε αντίγραφο του ίδιου λογισμικού. Με τον τρόπο αυτό, η εξαγωγή καθίσταται ιδιαίτερη δύσκολη και απαιτητική λειτουργία για να αυτοματοποιηθεί (πλεονέκτημα). Αλλά αν κάποιος εισβολέας έχει στα χέρια του δύο εκδόσεις του ίδιου λογισμικού και είτε χειροκίνητα είτε αυτόματα εντοπίσει τα σημεία που οι κώδικές διαφέρουν, τότε θα μπορεί να εντοπίσει και τα σημεία στα οποία έχει ενσωματωθεί το υδατογράφημα (μειονέκτημα).

Η S. Thaker λοιπόν στην εργασία της είχε παρουσιάσει αρχικά κάποιους ορισμούς προκειμένου να κάνει μία εισαγωγή γενικά για τα υδατογραφήματα λογισμικού αλλά και συγκεκριμένα για την δική της τεχνική υδατογράφησης. Η εισαγωγή περιείχε ορισμούς όπως για παράδειγμα τι είναι το υδατογράφημα και ποια χαρακτηριστικά πρέπει να το διέπουν. Επιπλέον είχε παρουσιάσει πως λειτουργεί η κεντρική μονάδα επεξεργασίας ενός ηλεκτρονικού υπολογιστή εν συντομία, δηλαδή ποιες εντολές χρησιμοποιεί και ποια τα τμήματα που την αποτελούν προκειμένου στη συνέχεια να βασιστεί σε αυτά τα στοιχεία και να παρουσιάσει την τεχνική της. Συγκεκριμένα η διαδικασία της ενσωμάτωσης του υδατογραφήματός της χωρίζονταν στα εξής στάδια:

- Μεταγλώττιση του πηγαίου κώδικα με απώτερο σκοπό την παραγωγή κώδικα Assembly (καθώς η ενσωμάτωση του υδατογραφήματος θα λάβει χώρα στο επίπεδο της Assembly).

- Εισαγωγή επιπρόσθετων Assembly εντολών με στόχο να ενσωματωθεί το επιθυμητό υδατογράφημα, χρησιμοποιώντας τρία βοηθητικά αρχεία. Το πρώτο αρχείο ήταν ένα αρχείο κεφαλίδας το οποίο εμπεριείχε για παράδειγμα τα ονόματα από τις μεταβλητές του κώδικα. Το δεύτερο αρχείο εμπεριείχε διάφορους μετασχηματισμούς οι οποίοι μπορούσαν να χρησιμοποιηθούν δίχως να αλλοιωθεί η λειτουργικότητα του λογισμικού, αριθμημένους με αύξουσα σειρά και εν συνεχεία το τρίτο εμπεριείχε μετασχηματισμούς εικονικούς οι οποίοι εφαρμοζόντουσαν και αυτοί με τη σειρά τους προκειμένου να προσδώσουν περισσότερη συσκότιση στο κώδικα του λογισμικού.

Η εξαγωγή του υδατογραφήματος τώρα, λάμβανε χώρα επίσης στο επίπεδο της Assembly με την διάφορα πως για να γίνει η μετάβαση σε αυτό το επίπεδο χρειάζονταν αντίστροφη μεταγλώττιση και όχι απλή. Επίσης αυτήν την φορά τώρα τα αρχεία που περιείχαν τους μετασχηματισμούς χρησιμοποιούνταν για να γίνει αναζήτηση στον κώδικα σημείων που έχουν υποστεί μετατροπή [61].

Το έτος 2004 δημοσιεύτηκε επίσης άλλη μία εργασία των C. Collberg, E. Carter, S. Debray, A. Huntwork, J. Kececioglu, C. Linn και M. Stepp με τίτλο: Dynamic path based software watermarking. Στην εργασία αυτή είχε παρουσιαστεί μία πρωτοποριακή με βάση τα ως τότε δεδομένα, τεχνική στον τομέα της υδατογράφησης λογισμικού. Η εν λόγω τεχνική στηρίζονταν στη δυναμική διακλάδωση συμπεριφοράς του λογισμικού. Το κύριο πλεονέκτημα της τεχνικής αυτής ήταν πως οι τεχνικές διόρθωσης λαθών και θωράκισης μπορούσαν να συνδυαστούν προκειμένου να δημιουργήσουν υδατογράφημα βασισμένο στα μονοπάτια εκτέλεσης και να το ενσωματώσουν καθιστώντας το ανθεκτικό σε πολλά είδα επιθέσεων [21].

Συνεχίζοντας την ανασκόπηση των τεχνικών υδατογράφησης ενώ ταυτόχρονα μένοντας στο ίδιο έτος δημοσιεύεται η εργασία των C. Collberg, S. Kouborov, E. Carter και C. Thomborson με τίτλο: Error correcting graphs for software watermarking. Οι συγγραφείς της εργασίας αυτής είχαν προτείνει μία γραφοθεοριτική προσέγγιση στις υδατογραφήσεις λογισμικού. Συγκεκριμένα η εν λόγω εργασία είχε επικεντρωθεί σε δύο αλγορίθμους οι οποίοι κωδικοποιούσαν πληροφορία μέσα στο λογισμικό, χρησιμοποιώντας μία δομή

γραφήματος. Εν συνεχεία, είχαν εφαρμόσει διαφορετικά μοντέλα επιθέσεων, εξάγοντας συμπεράσματα για το κάθε ένα από αυτά. Επιπλέον στην εργασία αυτή είχαν παρουσιαστεί και κάποιες κατηγορίες γραφημάτων τα οποία κάλλιστα μπορούσαν να χρησιμοποιηθούν ως υδατογραφήματα λογισμικών, αναλύοντας παράλληλα τις ιδιότητές τους [23].

Έναν χρόνο μετέπειτα και συγκεκριμένα το έτος 2005, δημοσιεύτηκε η εργασία των C. Collberg, A. Huntwork, E. Carter και E. Townsend με τον τίτλο: Graph theoretic software watermarks: implementation, analysis, and attacks. Στην εργασία αυτή είχε προταθεί μία τεχνική υδατογράφησης η οποία στηρίζονταν σε γραφήματα, δομές δηλαδή οι οποίες είχαν έρθει πρόσφατα στην επιφάνεια στον τομέα των υδατογραφημάτων λογισμικού και έδειχναν πολλά υποσχόμενες. Συγκεκριμένα οι συγγραφείς είχαν προτείνει ένα υδατογράφημα το οποίο ενσωματώνονταν στο διάγραμμα ελέγχου ροής του μη υδατογραφημένου κώδικα έχοντας εκ κατασκευής (το γράφημα αυτό) κάποιες ιδιότητες που το χαρακτήριζαν (π.χ., συγκεκριμένο αριθμό ακμών ανά κόμβο). Τα χαρακτηριστικά αυτά διευκόλυναν σε αρκετά μεγάλο βαθμό την εξαγωγή του υδατογραφήματος από τον κώδικα του λογισμικού καθώς επίσης χρησίμευαν και στον εντοπισμό τυχόν επιθέσεων. Η ενσωμάτωση λάμβανε χώρα με προσθήκη εικονικών μεθόδων οι οποίες δεν επηρέαζαν καθόλου την λειτουργικότητα του λογισμικού, απλά το μόνο που έκαναν ήταν να υπάρχουν μέσα στον κώδικα. Στην εργασία αυτή, είχε αναφερθεί επίσης και σημαντικό και μέχρι σήμερα ανοικτό πρόβλημα της σήμανσης λογισμικού προτάσσοντας κάποιες τεχνικές οι οποίες έδιναν λύσεις, όχι όμως ικανοποιητικά αποτελεσματικές έναντι επιθέσεων εντοπισμού σημείων σήμανσης από κακόβουλους χρήστες. Εν συνεχεία στην εργασία αυτή είχαν παρουσιαστεί παραδείγματα εφαρμογής της εν λόγω τεχνικής, επιθέσεις στο υδατογράφημα καθώς επίσης είχαν λάβει χώρα και αναλύσεις προκειμένου να εξαχθούν αντιπροσωπευτικά συμπεράσματα για την αποτελεσματικότητα της τεχνικής [22].

Ένα χρόνο ύστερα, δημοσιεύεται η εργασία των W. Zhu και C. Thomborson με τίτλο: Algorithms to watermark software through register allocation. Οι συγγραφείς είχαν παρουσιάσει τους αλγορίθμους QP και QPS οι οποίοι έκαναν υδατογράφηση λογισμικού μέσω μητρώο κατανομής. Συγκεκριμένα σχολίαζαν κάποιες δυσκολίες σχετικά με τη διαδικασία της εξαγωγής του υδατογραφήματος το οποίο είχε ενσωματωθεί μέσω των δύο αυτών αλγορίθμων, δίνοντας κάποια χαρακτηριστικά παραδείγματα [68].

Εν συνεχεία, στο ίδιο έτος είχε έρθει στην επιφάνεια και η εργασία των G. Myles και C. Collberg με τίτλο: Software watermarking via opaque predicates: implementation, analysis, and attacks [48]. Στην εργασία αυτή είχε αναλυθεί ένας ήδη δημοσιευμένος αλγόριθμός του G. Arboit (τι είναι τα αδιαφανή κατηγορήματα και πως κατασκευάζονται, κατασκευή υδατογραφήματος, ενσωμάτωση και εξαγωγή αυτού καθώς και αξιολόγηση υδατογραφήματος), τον οποίο τον είχε δημοσιεύσει στην εργασία του με τίτλο: A method for watermarking Java programs via opaque predicates [2].

Συγκεκριμένα ο G. Arboit είχε προτείνει μία τεχνική υδατογράφησης λογισμικού μέσω αδιαφανών κατηγορημάτων (opaque prdicates), η οποία ύστερα από εμπεριστατωμένη μελέτη από τους G. Myles και C. Collberg βρέθηκε πως αντιστέκεται σε ένα μεγάλο ποσοστό στα εξής είδη επιθέσεων:

- **Προσθετικές:** Επιθέσεις κατά τις οποίες προστίθεται κώδικας ο οποίος απευθύνεται σε νέο υδατογράφημα είτε επικαλύπτοντας το παλαιό είτε όχι.

- **Στρεβλωτικές:** Επιθέσεις κατά τις οποίες γίνονται τροποποιήσεις στον υδατογραφημένο κώδικα με σκοπό να αλλοιωθεί η δομή υδατογράφησης.

Μεταβαίνοντας στον επόμενο χρόνο και συγκεκριμένα στο έτος 2007, παρουσιάζεται η εργασία του J. Nagra με τίτλο: Threading software watermarking την οποία την είχε καταθέσει με σκοπό να αποκτήσει το διδακτορικό του δίπλωμα. Στην ουσία αυτό που είχε δημιουργήσει ο J. Nagra ήταν μία πρωτότυπη τεχνική υδατογράφησης λογισμικού βασισμένη σε λειτουργίες νημάτων (threads). Συγκεκριμένα στην εν λόγω εργασία είχε προταθεί η ενσωμάτωση ενός εύρωστου υδατογραφήματος σε λογισμικό χρησιμοποιώντας ανταγωνισμό μεταξύ νημάτων και αμοιβαίο αποκλεισμό τμημάτων του κώδικα. Είχε δημιουργήσει δηλαδή το υδατογράφημα προσθέτοντας καινούρια νήματα στα τμήματα του κώδικα τα οποία χρησιμοποιούσαν μονάχα ένα νήμα και με την βοήθεια του αμοιβαίου αποκλεισμού χειραγωγούσε την σειρά με την οποία εκτελούνταν με σκοπό να λειτουργήσει ορθά η ροή εκτέλεσης του κώδικα. Τα σημεία στα οποία ενσωμάτωνε αυτά τα καινούρια νήματα προκειμένου να επιλεγούν, έχρηζαν ιδιαίτερης προσοχής (δεν έπρεπε να γινόντουσαν άμεσα αντιληπτά από εισβολείς, ούτε η προσθήκη νημάτων στα σημεία αυτά να αύξανε σημαντικά τον χρόνο εκτέλεσης του λογισμικού). Επιπλέον όταν το λογισμικό εκτελούνταν με μία συγκεκριμένη ακολουθία εισόδου η δυναμική συμπεριφορά των νημάτων έπρεπε να ήταν χαρακτηριστική και να κωδικοποιούσε το υδατογράφημα. Εν συνεχεία προκειμένου να γίνονταν η εξαγωγή / εντοπισμός του υδατογραφήματος κατά την διάρκεια εκτέλεσης του λογισμικού γίνονταν χρήση παρακολουθητή (profiler).

Ως αποτέλεσμα της τεχνικής αυτής ήταν η αύξηση της πολυπλοκότητας του λογισμικού, καθώς η ανάλυση νημάτων καθίσταται πολύ δύσκολη και επίπονη εργασία μέσω στατικής ανάλυσης του κώδικα. Πάρα ταύτα όμως, το μειονέκτημα της εν λόγω τεχνικής ήταν πως σε περίπτωση προσθήκης πολλών νημάτων, πέραν από το επιτρεπτό όριο, ο χρόνος εκτέλεσης του λογισμικού αυξάνονταν ραγδαία και ο κώδικας γίνονταν ιδιαίτερα δύσχρηστος.

Συγκεκριμένα, η τεχνική του J. Nagra είχε στηριχτεί στην κατασκευή αδιαφανών κατηγορημάτων τα οποία είχαν προταθεί στο παρελθόν από δύο εργασίες του G. Arboit και C. Collberg αντίστοιχα. Η συνεισφορά του όμως ήταν το γεγονός ότι είχε προτείνει μία καινούρια χρήση των αδιαφανών κατηγορημάτων σε συνδυασμό με νήματα, η οποία αυτό που έκανε επί της ουσίας ήταν να συνενώνει δύο διαφορετικά τμήματα κώδικα τα οποία εμφανίζονταν πανομοιότυπα ύστερα από στατική ανάλυση. Εν ολίγοις η προσέγγισή του εμπεριείχε τα εξής στάδια:

- Παρακολούθηση μέσω ενός παρακολουθητή του λογισμικού το οποίο εκτελείται με μία συγκεκριμένη ακολουθία εισόδου.

- Ενσωμάτωση / προσθήκη / δήλωση νημάτων στα τμήματα εκείνα του προγράμματος όπου εκτελούνται από μονάχα ένα νήμα. Το μοτίβο (σειρά) σύμφωνα με το οποίο εκτελούνται τα νήματα είναι ξεχωριστό και κωδικοποιεί το υδατογράφημα.

- Η εξαγωγή του υδατογραφήματος γίνεται κατά την διάρκεια εκτέλεσης του λογισμικού δίνοντας μία συγκεκριμένη ακολουθία εισόδου κλειδί και αναλύοντας εν συνεχεία τις ιδιότητες (ποια νήματα εκτελέστηκαν μία συγκεκριμένη χρονική περίοδο, με ποια σειρά και από ποια τμήματα του κώδικα πέρασε η ροή εκτέλεσης) του ίχνος εκτέλεσης.

Εν τέλει στην εργασία αυτή είχε αποδειχτεί σε θεωρητικό υπόβαθρο η ορθότητα της εν λόγω τεχνικής, καθώς επίσης είχε δοθεί και μία θεωρητική προσέγγιση σύμφωνα με την οποία ήταν εξαιρετικά δύσκολη η επίθεση στον κώδικα του λογισμικού χρησιμοποιώντας στατική ανάλυση, εξαιτίας της χρήσης νημάτων [50].

Πλησιάζοντας στην ολοκλήρωση της ανασκόπισης το έτος 2008, δημοσιεύεται η εργασία των A. Mishra, R. Kumar και P. P. Chakrabarti με τίτλο: A method based whole program watermarking scheme for Java class files [46]. Στην παρούσα εργασία είχε παρουσιαστεί ένα μοντέλο στατικής υδατογράφησης λογισμικού το οποίο υδατογραφούσε Java κλάσεις, είχαν παρουσιαστεί επιθέσεις σε αυτό καθώς επίσης είχε συγκριθεί και με άλλα ήδη δημοσιευμένα μοντέλα υδατογράφησης προκειμένου να εξάγονταν αντιπροσωπευτικά συμπεράσματα για την ανθεκτικότητά του. Συγκεκριμένα το προκείμενο μοντέλο στηρίζονταν εξ ολοκλήρου στην δομή του λογισμικού και προκειμένου να ενσωματωθεί το υδατογράφημα σε κάθε μία από τις κλάσεις λάμβαναν χώρα οι εξής λειτουργίες:

- Δημιουργείται το υδατογράφημα (σύμφωνα με ορισμένο αλγόριθμο) το οποίο θα ενσωματωθεί μετέπειτα στη κλάση του λογισμικού (π.χ., το υδατογράφημα μπορεί να είναι είτε ένας αριθμός, είτε μία ακολουθία αριθμών).

- Για κάθε μία μέθοδο ξεχωριστά, εξάγονταν το διάγραμμα ελέγχου ροής, γίνονταν ανάθεση τιμών σε δομές οι οποίες είχαν προστεθεί κατά τη διαδικασία της υδατογράφησης σε συγκεκριμένα σημεία του κώδικα σύμφωνα με συγκεκριμένους τύπους, εν συνεχεία δημιουργούνταν το νέο διάγραμμα ελέγχου ροής (για το συγκεκριμένο τμήμα κώδικα) το οποίο εμπεριείχε μία ακολουθία με τμήματα με ψεύτικου κώδικα και ψεύτικων μεταβλητών των οποίων οι τιμές ήταν όμοιες με τις τιμές υδατογράφησης. Εν τέλει το καινούριο διάγραμμα ελέγχου ροής (που κωδικοποιούσε το υδατογράφημα), ενσωματώνονταν στο αρχικό διάγραμμα ελέγχου ροής ολόκληρου του λογισμικού στα κατάλληλα σημεία.

Συνεχίζοντας στο ίδιο έτος και συγκεκριμένα στο έτος 2008, δημοσιεύτηκε και η εργασία των D. Gong, F. Liu, B. Lu, P. Wang και L. Ding με τίτλο: Hiding information on Java class files. Στην εργασία αυτή οι συγγραφείς είχαν προτείνει ένα καινούριο μοντέλο υδατογράφησης λογισμικού το οποίο ενσωμάτωνε το υδατογράφημα στο εκτελέσιμο αρχείο με τον εξής τρόπο:

- Αναλύοντας την μορφή του εκτελέσιμου αρχείου σε Java.

- Αναδιατάσσοντας τους δείκτες των σταθερών στο κατάλληλο αρχείο όπου βρίσκονταν αποθηκευμένες με σκοπό την ενσωμάτωση του υδατογραφήματος.

Εν συνεχεία, στην εργασία αυτή είχαν πραγματοποιηθεί και πειράματα τα οποία είχαν δείξει πως με την εν λόγω εφαρμογή υδατογράφησης το μέγεθος του λογισμικού δεν αυξάνονταν (καθώς δεν προστήθονταν επιπλέον κώδικας) και αντιστέκονταν σθεναρά σε ποικίλες επιθέσεις κακόβουλων χρηστών [38].

Το έτος 2008 είχε επίσης δημοσιεύεται και η εργασία των M. D. Preda, R. Giacobazzi και E. Visentini με τίτλο: Hiding software watermarks in loop structures. Σε αυτήν την εργασία λοιπόν οι συγγραφείς είχαν εστιάσει στην ιδέα ενσωμάτωσης υδατογραφήματος μέσα σε συγκεκριμένα σημασιολογικά στιγμιότυπα καθώς επίσης είχαν εστιάσει και στην ιδέα εξαγωγής / εντοπισμού κάνοντας χρήση ενός κλειδιού προκειμένου να ανακτηθεί η πληροφορία που είχε χαθεί στις δομές που είχαν τροποποιηθεί, ανακατασκευάζοντας εν συνεχεία το υδατογράφημα. Συγκεκριμένα αυτό που πραγματοποίησαν οι συγγραφείς ήταν να ερευνήσουν τον τρόπο λειτουργίας των επαναληπτικών δομών και να βασίσουν τις διαδικασίες της ενσωμάτωσης και εξαγωγής στην κατανόηση της σημασιολογίας των ξεδιπλωμένων επαναλήψεων [53].

Στη συνέχεια μεταβαίνοντας στο έτος 2009 έρχεται στην επιφάνεια η εργασία των C. Zhang, J. Wang, C. Thomborson, C. Wang και C. Collberg με τίτλο: A semi dynamic multiple watermarking scheme for Java applications. Στην εργασία αυτή είχε προταθεί ένα μοντέλο υδατογράφησης, στο οποίο γίνονταν χρήση ενός αλγορίθμου των Huntwork, Carter και Townsend Collberg, προκείμενου να μετατραπούν σε ακολουθίες ακεραίων οι τιμές του υδατογραφήματος και του κλειδιού. Συγκεκριμένα στην εργασία αυτή είχαν δοθεί αρχικά κάποιοι ορισμοί σχετικά με το τι είναι η ενσωμάτωση, η εξαγωγή, ο εντοπισμός και η αναγνώριση του υδατογραφήματος λογισμικού. Στη συνέχεια είχε παρουσιαστεί ο αλγόριθμος σύμφωνα με τον οποίο μετατρέπονταν / κωδικοποιούνταν οι ακέραιοι σε μεταθέσεις, δίνοντας και παραδείγματα για καλύτερη κατανόηση του θέματος, ύστερα είχε παρουσιαστεί η τεχνική σύμφωνα με την οποία ενσωματώνονταν μέσα στον κώδικα του λογισμικού οι εν λόγω μεταθέσεις των ακεραίων αυτών καθώς επίσης είχε παρουσιαστεί και η τεχνική εντοπισμού των μεταθέσεων αυτών μέσα στον κώδικα. Αξίζει να σημειωθεί πως η ενσωμάτωση πραγματοποιούνταν με την βοήθεια του διαγράμματος ελέγχου ροής. Συγκεκριμένα, αφού είχε δημιουργηθεί το εν λόγω διάγραμμα από το μη υδατογραφημένο λογισμικό γινόταν επιλογή σημείων στα οποία θα ενσωματώνονταν ο επιπλέον κώδικας.

Το σκεπτικό σύμφωνα με το οποίο γίνονταν πράξη το μοντέλο των C. Zhang, J. Wang, C. Thomborson, C. Wang και C. Collberg ήταν να επιλέγεται ένα σημείο στο διάγραμμα ελέγχου ροής και το διάγραμμα να χωρίζεται σε δύο επιμέρους σημεία. Στην συνεχεία η ροή εκτέλεσης περνούσε είτε κατευθείαν από το πρώτο σημείο στο δεύτερο και ύστερα να συνέχιζε κανονικά (όπως και στο μη υδατογραφημένο λογισμικό), είτε να περνούσε (μετέβαινε) από το πρώτο σημείο σε ένα ενδιάμεσο όπου σε εκείνο θα έχει προστεθεί και ο επιπλέον κώδικας του υδατογραφήματος και εν συνεχεία διέρχονταν στο δεύτερο σημείο όπου από εκεί και ύστερα η ροή θα συνεχίζονταν όπως και στην αρχική έκδοση του μη υδατογραφημένου λογισμικού. Εν κατακλείδι προκειμένου να ολοκληρώσουν την εργασία τους οι συγγραφείς, ανέφεραν και την τεχνική σύμφωνα με την οποία το υδατογράφημα γίνονταν εξαγωγή από το λογισμικό καθώς επίσης είχαν παρουσιάσει και τα κριτήρια

σύμφωνα με τα οποία είχαν αξιολογήσει το μοντέλο τους παρουσιάζοντας πειραματικά αποτελέσματα [67].

Στη συνέχεια το επόμενο έτος και συγκεκριμένα το έτος 2011 δημοσιεύεται η εργασία των Z. Yu, C. Wang, C. Thomborson, J. Wang, S. Lian και A. V. Vasilakos με τίτλο: A novel watermarking method for software protection in the cloud. Στην εν λόγω εργασία οι συγγραφείς, είχαν προτείνει μία τεχνική στοχεύοντας στην υδατογράφηση αρχιτεκτονικών που υποστήριζαν εφαρμογές μέσω διαδικτύου καθώς τέτοιες εφαρμογές λογικά θα είναι και το μέλλον της επιστήμης των υπολογιστών. Αρχικά είχαν παρουσιάσει την εν λόγω αρχιτεκτονική, στη συνέχεια είχαν παρουσιάσει την διαδικασία ενσωμάτωσης και εξαγωγής του υδατογραφήματος και εν τέλει είχαν αξιολογήσει την τεχνική τους παρουσιάζοντας πειραματική αποτελέσματα [66].

Το έτος 2011 επίσης δημοσιεύεται η εργασία των J. Hamilton και S. Danicic με τίτλο: A survey of static software watermarking. Σε αυτήν την εργασία οι συγγραφείς είχαν προτείνει έναν γενικό διαχωρισμό των υδατογραφημάτων λογισμικού, σε στατικά και σε δυναμικά υδατογραφήματα, επικεντρώνοντας την έρευνά τους στη κατηγορία των στατικών. Στην πραγματικότητα αυτό που στην ουσία έκαναν ήταν να παρουσιάσουν μία έρευνα γύρο από τις γνωστές τεχνικές στατικής υδατογράφησης λογισμικού περιλαμβάνοντας επεξηγήσεις και εποικοδομητικά σχόλια για κάθε μία από αυτές [39].

Συνεχίζοντας στο ίδιο έτος και συγκεκριμένα στο έτος 2011, δημοσιεύτηκε και η εργασία των M. Chroni και S.D. Nikolopoulos με τίτλο: Efficient encoding of watermark numbers as eeducible permutation graphs. Η εργασία αυτή έδωσε και το εναρκτήριο λάκτισμα προκειμένου ο συγγραφέας να ερευνά στον τομέα της υδατογράφησης λογισμικού και να προσφέρει τα θεμέλια στο μοντέλο υδατογράφησης $WaterRpg$. Οι συγγραφείς είχαν προτείνει έναν αλγόριθμο μετατροπής / κωδικοποίησης ενός ακεραίου αριθμού σε ένα γράφημα μέσω αυτοαναστρέφουσας μετάθεσης και το αντίστροφο (αποκωδικοποίηση). Το εν λόγω γράφημα χαρακτηρίζονταν από συγκεκριμένες ιδιότητες (π.χ., συγκεκριμένος αριθμός των εξερχόμενων ακμών ανά κόμβο και περιττό πλήθος κόμβων) και αποτελούσε το έναυσμα για την επόμενη εργασία τους, σύμφωνα με την οποία το γράφημα αυτό που προέκυπτε από τον αλγόριθμό τους, το ενσωματώνονταν μέσα στον κώδικα λογισμικού με την μορφή υδατογραφήματος [13].

Ένα χρόνο αργότερα, οι ίδιοι συγγραφείς, δημοσιεύουν άλλη μία εργασία με τίτλο: An embedding graph based model for software watermarking. Στην εργασία αυτή είχε προταθεί μία τεχνική κωδικοποίησης υδατογραφήματος λογισμικού εντός του πηγαίου κώδικα, χρησιμοποιώντας το γράφημα κλήσεων. Το υδατογράφημα ενσωματώνονταν με την προσθήκη και διαγραφή κλήσεων μεταξύ συναρτήσεων του κώδικα χειραγωγώντας την ροή εκτέλεσης με συνθήκες ελέγχου προκειμένου να διατηρηθεί η ορθότητα λειτουργίας.

Το σκεπτικό πίσω από την εν λόγω τεχνική είναι η ένα προς ένα αντιστοιχία κάθε κόμβου του υδατογραφήματος με μία ήδη υπάρχουσα μέθοδο του λογισμικού και κάθε εξερχόμενης ακμής του υδατογραφήματος με μία επιπρόσθετη / εικονική κλήση μεθόδου του λογισμικού. Από μία αφηρημένη οπτική γωνία αυτό που στην ουσία πραγματοποιούσε η τεχνική τους ήταν η σύνδεση ήδη υπαρχόντων συναρτήσεων του λογισμικού με εικονικές

κλήσεις συναρτήσεων προκειμένου η εκτέλεση αυτών να κωδικοποιηθεί το υδατογράφημα μέσα σε αυτό [14].

Αξίζει να σημειωθεί επίσης πως το έτος 2013, οι συγγραφείς αυτοί αξιολογούν με πειραματική μελέτη επιθέσεων στην εργασία τους με τίτλο: Design and evaluation of a graph codec system for software watermarking, τα γραφήματα τα οποία παράγονται από τον αλγόριθμο της εργασίας τους [13] και καταλήγουν σε αποτελέσματα τα οποία δείχνουν πως αντιστέκονται σε αρκετά είδη επιθέσεων [16].

1.4.3 Επισκόπιση Υδατογραφημάτων Βασισμένα σε Γράφους

Τα υδατογραφήματα που έχουν ως βάση τα γραφήματα, από τις απαρχές της υδατογράφησης λογισμικού χαίρουν ιδιαίτερης προτίμησης καθώς μπορούν και προσομοιώνουν ή αλλιώς μιμούνται σε πολύ καλό βαθμό την ροή εκτέλεσης των λογισμικών (πολύ περισσότερο δε μία κατηγορία εξ αυτών ονόματι μεταθετικά αναγώγιμα γραφήματα) ενώ παράλληλα είναι δύσκολα αναλύσιμος ο κώδικας που δημιουργείται με την ενσωμάτωσή τους. Πάρα ταύτα, πολύ ερευνητές έχουν διαφορετική άποψη καθώς πιστεύουν πως αν από την εκτέλεση ενός λογισμικού στο ίχνος εκτέλεσης προκύψει ένα γράφημα το οποίο πληρεί γνωστές και αυστηρές ιδιότητες τότε αυτό αποτελεί σοβαρό κίνητρο για κακόβουλους χρήστες για να ψάξουν ενδελεχώς τον κώδικα.

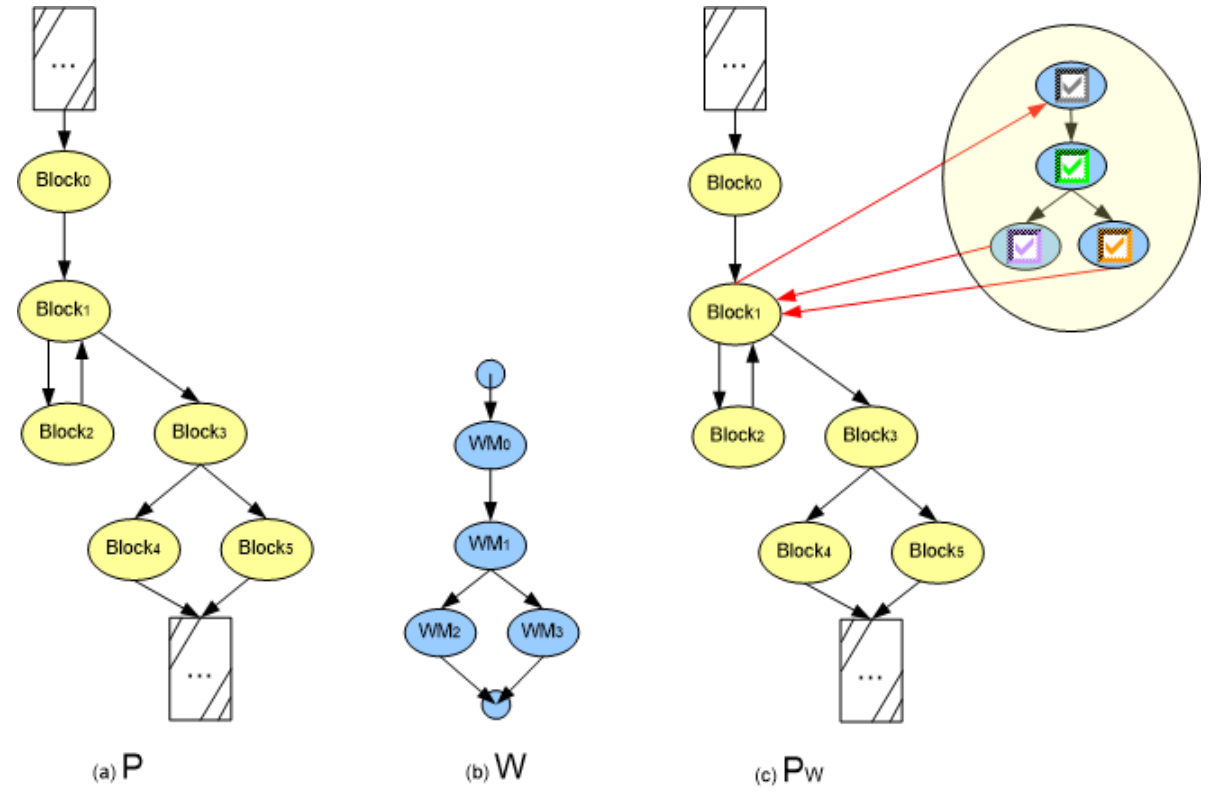
Εστιάζοντας σε αυτού τους είδους τα υδατογραφήματα γίνεται φανερό πως χωρίζονται σε δύο βασικές κατηγορίες, τα στατικά γραφήματα και τα δυναμικά γραφήματα. Τα στατικά ενσωματώνονται στο διάγραμμα ελέγχου ροής του εκάστοτε κώδικα, ενώ τα δυναμικά σε δομές δεδομένων ή σε ακολουθίες εντολών οι οποίες δημιουργούνται κατά την διάρκεια εκτέλεσης του κώδικα. Όπως και άλλες στατικές μεθόδους υδατογράφησης λογισμικού, έτσι και η στατική μέθοδος ενσωμάτωσης γραφημάτων είναι ιδιαίτερα ευπαθής και ευάλωτη στις τροποποιήσεις σημασιολογικών δομών του κώδικα (π.χ., εμφώλευση συνθηκών ελέγχου, ξεδίπλωμα δομών επανάληψης).

Ως επί το πλείστον η βασική ιδέα σε αυτά τα γραφήματα είναι πως κωδικοποιούν έναν ακέραιο αριθμό. Δηλαδή ένας αριθμός, μέσω ενός αλγορίθμου κωδικοποιείται στη δομή ενός γραφήματος το οποίο εν συνεχεία θα ενσωματωθεί σε ένα λογισμικό. Η οικογένεια γραφημάτων αυτή ονομάζεται graph enumeration και τα γραφήματα τα οποία ανήκουν σε αυτήν είναι τα εξής:

- **Κατευθυνόμενα Δέντρα με Δείκτη στον Πατέρα:** (Directed Parent - Pointer Trees) Στα γραφήματα αυτά υπάρχει μία και μόνο ακμή μεταξύ κόμβου και πατέρα.
- **Επιπεδικά Κυβικά Δέντρα** (Planted Planar Cubic Trees): Τα γραφήματα αυτά είναι δυαδικά δέντρα όπου κάθε εσωτερικός κόμβος τους εκτός της ρίζας έχει 2 παιδιά.
- **Γραφήματα Ρίζας** (Radix Graphs): Τα γραφήματα αυτά προσθέτουν ένα επιπλέον πεδίο δείκτη σε κάθε κόμβο μίας κυκλικής συνδεδεμένης λίστας μεγέθους k προκειμένου να κωδικοποιήσουν ένα $base - k$ ψηφίο.

- **Μεταθετικά Γραφήματα** (Permutation Graphs): Όπως και τα γραφήματα ρίζας, έτσι και αυτά τα γραφήματα χρησιμοποιούν την ίδια μονή συνδεδεμένη κυκλική λίστα με την διαφορά όμως πως έχουν ιδιότητες αυτοδιόρθωσης (error - correcting).
- **Αναγώγιμα Μεταθετικά Γραφήματα** (Reducible Permutation Graphs): Τα γραφήματα αυτά μοιάζουν αρκετά με τα μεταθετικά γραφήματα με την μόνη διαφορά πως είναι μοιάζουν σε μεγάλο βαθμό με τα διαγράμματα ελέγχου ροής (control flow graphs).

Ανάλογα με τα χαρακτηριστικά του εκάστοτε κώδικα προς υδατογράφηση (π.χ., αν υπάρχουν πολλές κλήσεις συναρτήσεων ανά συνάρτηση, αν ο κώδικας αποτελείται από κλάσεις από τις οποίες δεν δημιουργούνται εξαρτήσεις) επιλέγεται και η αντίστοιχη κατηγορία γραφημάτων. Στόχος πάντα είναι η δομή του κώδικα αν αναπαρασταθεί με κάποιον τρόπο ως γράφημα να μοιάζει όσο το δυνατόν περισσότερο με το γράφημα που πρόκειται να αποτελέσει το υδατογράφημα.



Σχήμα 1.9: Γενική περίπτωση ενσωμάτωσης γραφοθεωριτικών υδατογραφημάτων.

Τοποθετώντας όμως ένα γράφημα μέσα ένα λογισμικό, πρέπει με κάποιον τρόπο αυτό να μπορεί να εξαχθεί σε μελλοντικό χρόνο. Εν ολίγοις, να μπορούν να αναγνωριστούν τα σύνολα των ακμών και των κόμβων του στο εσωτερικό του κώδικα. Το πρόβλημα αυτό

λύνονταν είτε με σημάνσεις (marks) μέσα στον κώδικα, είτε με κάποιους αυτοσχέδιους μαθηματικούς τύπους οι οποίοι λάμβαναν υπόψη συχνότητες (π.χ., εμφάνισης ομάδων εντολών) πάντα με την βοήθεια ενός κατωφλίου (ορίου). Το όριο αποτελούσε την ασφαλιστική δικλίδα η οποία ήταν εκείνη που αποφάσιζε πότε γίνεται εντοπισμός και πότε όχι.

Από την μία πλευρά οι σημάνσεις, στην ουσία ήταν επιπρόσθετος κώδικας ο οποίος δήλωνε πως σε εκείνο το σημείο υπήρχε ένα σημάδι (μία δομή υδατογράφησης ή ένα τμήμα αυτής), όμως έδιναν στόχο σε κακόβουλους χρήστες καθώς δεν είχαν ιδιαίτερη συνοχή με τον υπόλοιπο κώδικα. Ενώ από την άλλη πλευρά, οι μαθηματικοί τύποι δεν ενσωματωνόντουσαν στο εσωτερικό του κώδικα, αλλά στις μεταβλητές που εμπεριείχαν έμπαιναν τιμές οι οποίες εξάγονταν από πληροφορίες του κώδικα (π.χ., ύστερα από την εκτέλεση του κώδικα με μία ακολουθία εισόδου πόσες συνθήκες ελέγχου εκτελέστηκαν σε κάθε συνάρτηση).

Το μεγαλύτερο μειονέκτημα των σημείων σημάνσεων ήταν πως μπορούσαν εύκολα να αφαιρεθούν δίχως να επηρεάζεται η λειτουργικότητα του προγράμματος. Με απλά λόγια, το υδατογράφημα και ο αρχικός κώδικας του λογισμικού παραμένουν ανέπαφα, αλλά από τη στιγμή που δεν υπάρχουν σημεία σήμανσης δεν μπορεί να γίνει η διάκριση μεταξύ τους. Επομένως ό,τι κέρδιζε κάποιος χρησιμοποιώντας ως υδατογράφηματα γραφήματα το έχανε από τις σημάνσεις που χρησιμοποιούσε μέσα στον κώδικα. Αντιπροσωπευτικά παραδείγματα σημάνσεων είναι τα εξής:

- Προσθήκη κώδικα ο οποίος δεν επιτελεί κάποια λειτουργία η οποία να επηρεάζει τον μη υδατογραφημένο κώδικα (π.χ., φορτώνοντας μία τιμή σε μία μεταβλητή η οποία ποτέ δεν χρησιμοποιείται, ή προσθέτοντας συνθήκες ελέγχου - διακλαδώσεις οι οποίες δεν σχετίζονται με τον υπόλοιπο κώδικα).

- Η καταμέτρηση των εντολών ενός τμήματος κώδικα και η χρησιμοποίηση της τιμής αυτής ως σημείο σήμανσης (π.χ., αν οι εντολές είναι 22 σε αριθμό και στόχος είναι να τεθεί στο σημείο σήμανσης το 24, προστίθενται 2 εντός *nop* οι οποίες δεν κάνουν καμία ενέργεια).

- Καταμέτρηση των προσβάσεων σε στατικές μεταβλητές προκειμένου να καθοριστεί η σήμανση.

- Ο υπολογισμός ενός checksum των όλων εντολών και η χρήση ενός η περισσοτέρων bits αυτού ως σήμανση.

- Μετασχηματισμός σε κανονική μορφή της ακολουθίας εντολών σε κάθε τμήμα κώδικα και μετέπειτα η τροποποίηση αυτού με στόχο να κωδικοποιηθούν οι σημάνσεις.

- Προσθήκη σημάνσεων στα meta data τα οποία συνδέονται σε κάθε τμήμα κώδικα (π.χ., αλλαγή πληροφοριών οι οποίες συνδέονται με την αποσφαλμάτωση προκειμένου να κωδικοποιηθούν οι σημάνσεις).

Τα δυναμικά υδατογραφήματα γραφημάτων είναι ανθεκτικά στις περισσότερες τροποποιήσεις σημασιολογικών δομών και δεν δίνουν ερεθίσματα σε κακόβουλους χρήστες αν το

γράφημα το οποίο είναι ενσωματωμένο έχει επιλεγεί σωστά (να ταιριάζει σε μεγάλο βαθμό με τις δομές που δημιουργούνται κατά την διάρκεια εκτέλεσης). Πάρα ταύτα δεν έχει σχεδιαστεί ως τώρα αλγόριθμος ο οποίος παίρνει ως είσοδο ένα ίχνος εκτέλεσης ενός λογισμικού και να παράγει ως έξοδο κάποιο βέλτιστο ή σχεδόν βέλτιστο (optimal) γράφημα το οποίο να ενδείκνυται για ενσωμάτωση.

Κεφαλαιο 2

Υδατογραφηση Λογισμικου

2.1 Υδατογραφήματα

2.2 Αλγοριθμικές Τεχνικές Υδατογράφησης Λογισμικού

2.3 Μεθοδολογία Κατασκευής και Ενσωμάτωσης Υδατογραφημάτων

2.4 Επιθέσεις σε Υδατογραφήματα Λογισμικού

2.5 Γραφήματα Παραγόμενα από Κώδικες Λογισμικών

2.6 Ανάλυση Κώδικα Λογισμικού

2.7 Ιδανική Κλάση Γραφοθεωρητικών Υδατογραφημάτων

2.8 Η Σημασία της Διαφάνειας

2.1 Υδατογραφήματα

Ένα από τα σημαντικότερα, δύσχρηστα και άλυτα προβλήματα στην υδατογράφηση λογισμικού είναι το υδατογράφημα αφού ενσωματωθεί στον κώδικα του λογισμικό να είναι διαφανές (stealthy) τόσο όσο να μην γίνεται αντιληπτό από στα μάτια κακόβουλων χρηστών, ενώ παράλληλα να μπορεί ανά πάσα στιγμή να εξαχθεί αποτελεσματικά από τον πραγματικό ιδιοκτήτη. Το γεγονός αυτό αποτελεί ίσως τον μεγαλύτερο συμβιβασμό (tradeoff) της υδατογράφησης λογισμικού καθώς αν το υδατογράφημα είναι πολύ εύκολο να εξαχθεί από το λογισμικό κάποιος εισβολέας θα μπορέσει γρήγορα να το βρει, ενώ αν το υδατογράφημα είναι πολύ δύσκολο να εξαχθεί από το λογισμικό ο ιδιοκτήτης του ίσως να μην είναι σε θέση πάντα να το εξάγει.

Βέβαια, θα έλεγε κανείς πως από τη στιγμή που προστίθεται κώδικας μέσα στον ήδη υπάρχον κώδικα, αυτός όπως και να γίνει θα φαίνεται. Το θέμα στην όλη υπόθεση είναι

πρώτον, ο κώδικας αυτός να μην δίνει κανένα κίνητρο για ψάξιμο σε εισβολείς που τον κοιτούν και δεύτερον, να είναι τόσο καλά συνδεδεμένος με τον υπόλοιπο μη υδατογραφημένο κώδικα έτσι ώστε να χρειάζεται πολύ ώρα προκειμένου κάποιος να διεκπεραιώσει την αποσύνδεσή του. Με αποτέλεσμα να είναι ασύμφορα γι αυτόν να ασχοληθεί με αυτού του είδους τις ενέργειες.

2.1.1 Κατηγοριοποίηση

Προκειμένου να κατασκευαστεί ένα υδατογράφημα λογισμικού πρέπει προηγουμένως να καταστεί σαφές ποιο θα είναι το είδος προστασίας που θα προσφέρει με την ενσωμάτωσή του στον κώδικα. Έχοντας ως βάση το γεγονός αυτό τα υδατογραφήματα λογισμικού κατηγοριοποιούνται ως εξής:

- **Σήματα συντάκτη** (Authorship marks): Τα υδατογραφήματα αυτής της κατηγορίας ενσωματώνονται με την μορφή ενός μοναδικού αναγνωριστικού μέσα στον κώδικα ενός λογισμικού το οποίο χαρακτηρίζει τον κάτοχο των πνευματικών δικαιωμάτων. Συγκεκριμένα αν κάποιος έχει κατασκευάσει ένα λογισμικό και επιθυμεί να το διαθέσει στην αγορά, αυτό που πρέπει να πράξει πρώτα είναι να ενσωματώσει σε κάθε αντίγραφο του λογισμικού το ίδιο υδατογράφημα προκειμένου σε μελλοντικό χρόνο αν χρειαστεί να το εξάγει και να αποδείξει πως αυτός είναι ο πραγματικός ιδιοκτήτης. Τέτοια υδατογραφήματα θα μπορούσε να είναι παραδείγματος χάριν γραφήματα τα οποία φέρουν συγκεκριμένες ιδιότητες.

- **Σήματα δακτυλικού αποτυπώματος** (Fingerprinting marks): Τα υδατογραφήματα αυτά, σε αντίθεση με τα authorship marks, κάθε φορά που ενσωματώνονται στο αντίγραφο ενός κώδικα λογισμικού παίρνουν και μοναδική μορφή. Απώτερος σκοπός του γεγονότος αυτού είναι σε μελλοντικό χρόνο να μπορεί να εντοπιστεί ο αγοραστής του λογισμικού (όχι αυτός που έχει τα πνευματικά δικαιώματα, αλλά αυτός που έκανε την αγορά από αυτόν που έχει τα πνευματικά δικαιώματα) ο οποίος θα είναι και ο πραγματικός υπαίτιος που άφησε να διαρρεύσει το υδατογραφημένο λογισμικό σε άτομα που δεν έχουν το νόμιμο δικαίωμα. Τέτοια υδατογραφήματα θα μπορούσε να είναι συμβολοσειρές (π.χ., δυαδικές) οι οποίες αφού αναγνωριστούν και εξαχθούν προδίδουν τον αγοραστή. Υδατογραφήματα αυτού του είδους χρησιμοποιούνται για παράδειγμα σε λογισμικά τα οποία είναι ακόμα σε πρώιμο στάδιο (beta copies) από εταιρίες οι οποίες τα παρέχουν σε κάποιες άλλες εταιρίες για εντοπιστούν τυχόν προβλήματα ασυμβατότητας. Επομένως στην περίπτωση κατά την οποία οι εταιρίες οι οποίες έλαβαν τα υδατογραφημένα λογισμικά τα διαθέσουν μετέπειτα στο κοινό παρανόμως, οι εταιρίες οι οποίες τα έστειλαν να μπορούν να αποδείξουν τα αυτονόητα.

- **Σήματα άδειας** (Licensing marks): Αυτά τα υδατογραφήματα, χρησιμοποιούνται προκειμένου να πιστοποιηθεί η αυθεντικότητα ενός λογισμικού μέσω ενός κλειδιού το οποίο συνήθως είναι μία ακολουθία αποτελούμενη από γράμματα ή αριθμούς η οποία ζητείται μία φορά μόνο κατά την εγκατάσταση του λογισμικού σε έναν ηλεκτρονικό

υπολογιστή για να αρχίσει να εκτελείται το υδατογραφημένο λογισμικό. Το κλειδί αυτό, πρέπει να είναι αναποτελεσματικό σε περίπτωση όπου το ενσωματωμένο υδατογράφημα έχει υποστεί αλλοιώσεις. Με τον όρο αναποτελεσματικό εννοείται πως αν ο κώδικας του υδατογραφήματος έχει τροποποιηθεί, δίνοντας το κλειδί στο λογισμικό, αυτό να μην λειτουργεί σωστά. Τέτοια υδατογραφήματα για παράδειγμα έχουν τα λογισμικά της Microsoft και της Adobe τα οποία αν ο υπολογιστής είναι συνδεδεμένος στο διαδίκτυο, πραγματοποιείται ο κατάλληλος έλεγχος και αν είναι θετικός το λογισμικό παύει να λειτουργεί σωστά βγάζοντας προειδοποιητικά μηνύματα.

- **Σήματα έγκρισης** (Validation marks): Αυτού του είδους τα υδατογραφήματα χρησιμοποιούνται με σκοπό να καθίσταται εφικτή η αναγνώριση ενός λογισμικού ως αυθεντικό και μη αλλοιωμένο. Τέτοια υδατογραφήματα για παράδειγμα, αποτελούν οι ψηφιακές υπογραφές εκείνων των προγραμμάτων που έχουν κωδικοποιηθεί στη γλώσσα προγραμματισμού Java προκειμένου οι χρήστες που τα κατεβάζουν από το διαδίκτυο και τα χρησιμοποιούν, μόλις τα κατεβάσουν να μπορούν να ελέγξουν αν αυτά έχουν υποστεί ενδεχόμενη τροποποίηση.

Ενα σοβαρό μειονέκτημα των υδατογραφημάτων τύπου fingerprinting marks είναι πως αν κάποιος εισβολέας καταφέρει να αποκτήσει πρόσβαση σε 2 ή και περισσότερα αντίγραφα ενός λογισμικού το οποίο προστατεύεται με αυτά, τότε από τη στιγμή που αυτά είναι διαφορετικά σε κάθε αντίγραφο αυτός μπορεί εύκολα να συγκρίνει τους κώδικές τους, να εντοπίσει τα σημεία στα οποία βρίσκονται, και στη συνέχεια να τα αλλοιώσει. Για τον λόγο αυτό, συνήθως όταν γίνεται χρήση τέτοιων υδατογραφημάτων αυτά συνδυάζονται με τεχνικές συσκότισης, ούτως ώστε οι κώδικες να μην διαφέρουν μονάχα στα σημεία όπου λαμβάνει χώρα η υδατογράφηση αλλά και σε διάφορα άλλα τυχαία σημεία. Βέβαια ο συνδυασμός αυτός επιφέρει μία δυσφορία στα άτομα τα οποία συντηρούν λογισμικά με τέτοιου είδους προστασία, αλλά δεν μπορεί να γίνει και κάπως διαφορετικά ακολουθώντας αυτόν τον τρόπο προστασίας.

Αξίζει επιπλέον να σημειωθεί πως σε ένα λογισμικό μπορούν να ενσωματωθούν και πολλαπλά υδατογραφήματα τύπου authorship ή fingerprinting marks προκειμένου να προστατεύονται τα πνευματικά δικαιώματα όλων όσων έχουν συνεισφέρει για να κατασκευαστεί αυτό. Το πρόβλημα όμως σε αυτήν την περίπτωση είναι πως με την εισαγωγή περισσοτέρων του ενός υδατογραφήματα στον κώδικα ενός λογισμικού υπάρχει σοβαρός κίνδυνος όταν εισάγεται κάποιο, να αλλοιώνονται τα προηγούμενα.

Με μία πρώτη ανάγνωση επίσης κάποιος θα έλεγε πως η φιλοσοφία των υδατογραφημάτων τύπου licensing marks μοιάζει πολύ με αυτήν που χαρακτηρίζει την τεχνική θωράκισης κώδικα, κατά την οποία υπάρχει η φάση του ελέγχου και η φάση της αντίδρασης. Και πως η μόνη διαφορά των υδατογραφημάτων τύπου licensing marks και validation marks είναι πως στα δεύτερα αν το κλειδί που ζητείται για εισαγωγή είναι λανθασμένο το λογισμικό δεν εκτελείται. Οι διαφορές μεταξύ των διαφορετικών τύπων υδατογραφημάτων μπορεί να είναι μικρές και για κάποιους αμελητέες, αλλά είναι αυτές που καθορίζουν τον τύπο προστασίας για τα πνευματικά δικαιώματα που παρέχεται σε κάποιο λογισμικό.

2.1.2 Ιδιότητες Υδατογραφημάτων Λογισμικού

Τα υδατογραφήματα λογισμικού πέραν από τις κατηγορίες στις οποίες αυτά κατατάσσονται, φέρουν και ιδιότητες οι οποίες τους παρέχουν ένα είδος ξεχωριστής ταυτότητας. Η κάθε μία από τις ιδιότητες προσεγγίζει τις δομές υδατογράφησης από μία δική της οπτική γωνία, χωρίζοντάς τες κατά κάποιον τρόπο σε 2 συμπληρωματικά είδη. Επίσης οι ιδιότητες αυτές, είναι αλληλένδετες με τις τεχνικές υδατογράφησης καθώς τις δίνουν συγκεκριμένο χαρακτήρα και κουλτούρα προκειμένου να λειτουργούν προσαρμόζοντας παράλληλα τις ενέργειές τους με σκοπό να ενσωματώσουν την εκάστοτε δομή στο εσωτερικό του λογισμικού. Οι ιδιότητες αυτές δίνονται ανά ζεύγη και είναι οι ακόλουθες:

- **Στατικά / Δυναμικά** (Static / Dynamic): Οι ιδιότητες αυτές αφορούν τον τρόπο με τον οποίον τα υδατογραφήματα ενσωματώνονται και εξάγονται από το εκάστοτε λογισμικό. Αν ένα υδατογράφημα έχει την ιδιότητα του στατικού αυτό σημαίνει πως για να εξαχθεί από το λογισμικό στο οποίο έχει ενσωματωθεί δεν χρειάζεται να γίνει εκτέλεση, η εξαγωγή του μπορεί να προβεί για παράδειγμα κατευθείαν από τον πηγαίο κώδικα μέσω ενός αναλυτή (parser). Ενώ αν ένα υδατογράφημα έχει την ιδιότητα του δυναμικού, αυτό σημαίνει πως για να εξαχθεί από το λογισμικό χρειάζεται η εκτέλεσή του μέσω μίας συγκεκριμένης ακολουθίας εισόδου I, η οποία θα αποτελεί και το κλειδί υδατογράφησης, και η μετέπειτα καταγραφή όλων των ενεργειών που πραγματοποιούνται για παράδειγμα μέσω ενός παρακολουθητή (profiler).

- **Εύθραυστα / Εύρωστα** (Fragile / Robust): Οι ιδιότητες αυτές αφορούν το κατά πόσο εύκολα ή δύσκολα τα υδατογραφήματα μπορούν να υποστούν αλλοιώσεις στη δομή τους και να μην μπορούν να εξαχθούν. Συγκεκριμένα αν ένα υδατογράφημα έχει την ιδιότητα του εύθραυστου, αυτομάτως αυτό σημαίνει πως σχεδόν με οποιαδήποτε μικρή αλλαγή το υδατογράφημα αλλοιώνεται. Υδατογραφήματα με αυτήν την ιδιότητα συνήθως συνδυάζονται με την τεχνική της θωράκισης καθώς είναι ιδιαίτερα βολικά στο να επιτυγχάνεται ορθός έλεγχος παραβίασης κώδικα από τρίτους. Ενώ αν ένα υδατογράφημα έχει την ιδιότητα του εύρωστου, αυτό σημαίνει πως ακόμα και με τροποποιήσεις στον κώδικα που το κατασκευάζει αυτό διατηρείται αναλλοίωτο.

- **Ορατά / Αόρατα** (Visible / Invisible): Οι ιδιότητες αυτές αφορούν την ορατότητα των υδατογραφημάτων μέσα στον κώδικα του λογισμικού. Επί της ουσίας, τα υδατογραφήματα τα οποία έχουν την ιδιότητα των ορατών περιγράφουν έναν αλγόριθμο υδατογράφησης όπου η συνάρτηση αναγνώρισης είναι γνωστή. Με αποτέλεσμα οποιοσδήποτε να είναι σε θέση να δει το υδατογράφημα και είτε να το καταστρέψει εντελώς, είτε να το αλλάξει με όποιον τρόπο αυτός επιθυμεί, δημιουργώντας έτσι ένα καινούργιο υδατογράφημα μη αναγνωρίσιμο από τον νόμιμο ιδιοκτήτη του προγράμματος. Από την άλλη πλευρά, η ιδιότητα των αόρατων υδατογραφημάτων περιγράφει μία συνάρτηση αναγνώρισης, η οποία δεν είναι γνωστή, παρά μόνο σε αυτόν που κατασκευάζει το υδατογράφημα καθιστώντας με αυτόν τον τρόπο δύσκολη και χρονοβόρα την δουλεία κάποιου να το εντοπίσει και να το βγάλει ή ακόμα και να το μεταλλάξει.

- **Εστιασμένα / Διάχυτα** (Focused / Spread spectrum): Οι ιδιότητες αυτές αφορούν το πλήθος και τα σημεία του κώδικα στα οποία έχουν λάβει χώρα οι τροποποιήσεις προκειμένου το υδατογράφημα να ενσωματωθεί. Αν ένα υδατογράφημα έχει την ιδιότητα των εστιασμένων, αυτό σημαίνει πως ο κώδικας ο οποίος κατασκευάζει το υδατογράφημα δεν είναι εκτενής και βρίσκεται σε ένα συγκεκριμένο σημείο μέσα στον κώδικα του λογισμικού, ενώ αν ένα υδατογράφημα έχει την ιδιότητα των διάχυτων είτε ο κώδικας που το κατασκευάζει είναι εκτενής, είτε δεν είναι οι τροποποιήσεις έλαβαν χώρα σε πολλά σημεία μέσα στον κώδικα.

- **Ενημερωμένα / Τυφλά** (Informed / Blind): Οι ιδιότητες αυτές σχετίζονται με την απαραίτητη πληροφορία που χρειάζεται προκειμένου να πραγματοποιηθεί η εξαγωγή του υδατογραφήματος από τον κώδικα. Συγκεκριμένα, αν ένα υδατογράφημα έχει την ιδιότητα των ενημερωμένων αυτό σημαίνει πως για να πραγματοποιηθεί η εξαγωγή του από τον κώδικα απαιτείται συν της άλλης και ο μη υδατογραφημένος κώδικας ή το υδατογράφημα, πράγμα που για υδατογράφημα το οποίο έχει την ιδιότητα των τυφλών δεν ισχύει.

Σε πολλές εργασίες που έχουν δημοσιευτεί σε διεθνή επιστημονικά συνέδρια και περιοδικά, άτυπα, για λόγους αποφυγής συγχύσεων και γενικά ευκολίας στη συγγραφή, οι ερευνητές χρησιμοποιούν την ιδιότητα των υδατογραφημάτων ως στατικά / δυναμικά προκειμένου να υπάρχει ένας απλός διαχωρισμός μεταξύ των τεχνικών υδατογράφησης λογισμικού. Δηλαδή στη γενική περίπτωση όταν μία τεχνική ενσωματώνει υδατογράφημα το οποίο για να εξαχθεί δεν απαιτείται η εκτέλεση του λογισμικού κατατάσσεται στις στατικές τεχνικές, ενώ όταν μία τεχνική ενσωματώνει υδατογράφημα το οποίο για να εξαχθεί απαιτεί την εκτέλεση του λογισμικού κατατάσσεται στις δυναμικές τεχνικές.

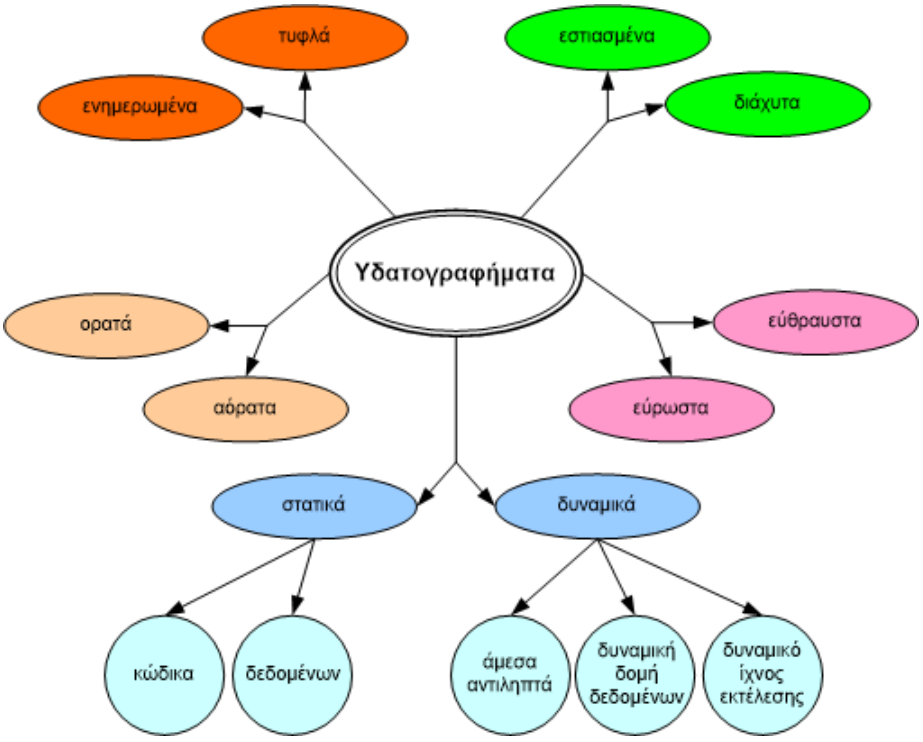
Επί της ουσίας όταν ένα υδατογράφημα έχει τη στατική ιδιότητα ενσωματώνεται άμεσα στον κώδικα ενός λογισμικού χωρίς να τροποποιείται κατά την διάρκεια εκτέλεσής του, ενώ όταν ένα υδατογράφημα έχει την δυναμική ιδιότητα ενσωματώνεται έμμεσα στον κώδικα ενός λογισμικού. Δηλαδή υπάρχει στο λογισμικό μέσα ο κώδικας ο οποίος το κωδικοποιεί, αλλά μονάχα στην περίπτωση όπου δοθεί η ακολουθία εισόδου κλειδί αυτό κατασκευάζεται.

Στην περίπτωση κατά την οποία ένα υδατογράφημα έχει την ιδιότητα των ενημερωμένων έχει ένα σοβαρό μειονέκτημα καθώς χρειάζεται περισσότερη πληροφορία, συγκριτικά με ένα υδατογράφημα που έχει την ιδιότητα των τυφλών, προκειμένου να εξαχθεί από ένα λογισμικό. Μία απλή και συνάμα πολύ συχνή περίπτωση κατά την οποία ένα ενημερωμένο υδατογράφημα δημιουργεί προβλήματα είναι εκείνη η περίπτωση κατά την οποία το λογισμικό έχει υδατογραφηθεί και στη συνέχεια επεκτείνεται ξανά και ξανά προκειμένου να του προστεθούν επιμέρους λειτουργίες. Τότε πρέπει αναγκαστικά να οι αλλαγές να γίνονται και στο μη υδατογραφημένο και στο υδατογραφημένο λογισμικό. Γεγονός το οποίο δυσκολεύει σε πολύ μεγάλο επίπεδο την συγγραφή κώδικα.

Εστιάζοντας στην στατική. δυναμική ιδιότητα των υδατογραφημάτων γίνεται φανερό πως τα δυναμικά υδατογραφήματα έχουν ένα μεγάλο πλεονέκτημα έναντι των στατικών. Καθώς από την στιγμή που δεν βασίζονται σε ιδιότητες του κώδικα (π.χ., συχνότητες

εμφάνισης εντολών) δύσκολα αλλοιώνεται η δομή τους ύστερα από τροποποιήσεις των σημασιολογικών δομών του κώδικα (π.χ., συγχώνευση συνθηκών ελέγχου). Η μόνη δίοδος που μπορεί να ακολουθηθεί από κακόβουλους χρήστες προκειμένου να επιφέρουν επιβλαβείς τροποποιήσεις στην δομή τους είναι να προκαλέσουν αλλαγές όχι στις ιδιότητες του κώδικα, αλλά στις δομές που δημιουργεί ο κώδικας αφότου αυτός εκτελεστεί. Πράγμα που από τη φύση του χρειάζεται περισσότερη εμπειρία και χρόνο.

Στο σημείο αυτό, τονίζεται πως κανένας συνδυασμός ιδιοτήτων δεν επαρκεί προκειμένου ένα υδατογράφημα με την εισαγωγή του σε κώδικα λογισμικού να καθίσταται αναλλοίωτο έναντι τροποποιήσεων και να περνά απαρατήρητο από τα μάτια κακόβουλων χρηστών. Κάθε μια από τις ιδιότητες προσφέρει ταυτοχρόνως και πλεονεκτήματα αλλά και μειονεκτήματα. Σημαντικό ρόλο όμως στην επίτευξη αποτελεσματικής υδατογράφησης παίζει η εμπειρία του χρήστη στον προγραμματισμό, καθώς αυτός πρέπει να έχει στο μυαλό του και τις πιθανές τροποποιήσεις κακόβουλων χρηστών, αλλά και τις τυχόν επεκτάσεις οι οποίες κατά πάσα πιθανότητα λάβουν χώρα στο μέλλον.



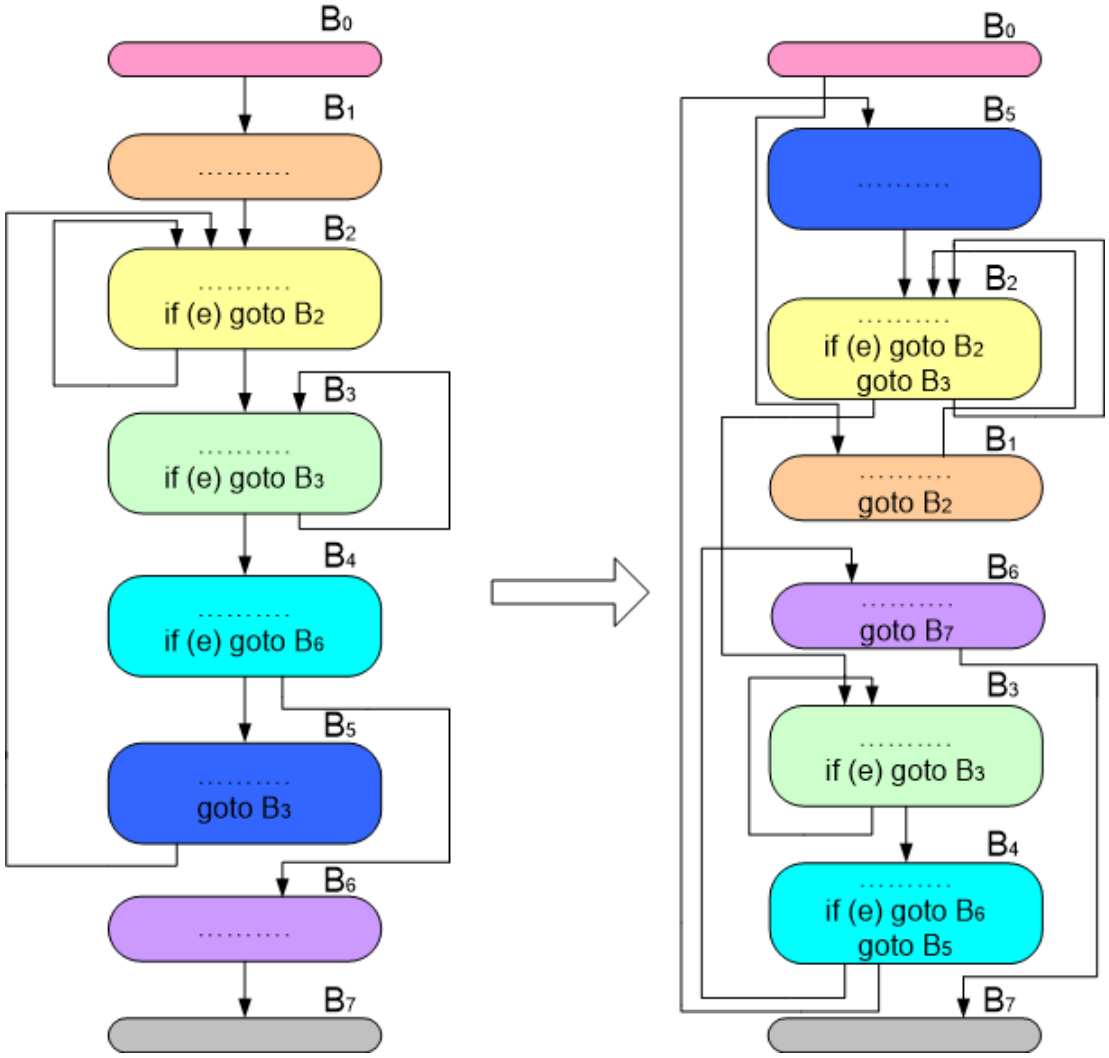
Σχήμα 2.1: Ιδιότητες υδατογραφημάτων λογισμικού.

Όμως η στατική / δυναμική ιδιότητα των υδατογραφημάτων χωρίζεται σε επιπλέον ιδιότητες οι οποίοι προσθέτουν με την σειρά τους σε αυτά επιπλέον χαρακτηριστικά. Τα χαρακτηριστικά αυτά εστιάζουν επί της ουσίας στον τρόπο με τον οποίο το εκάστοτε υδατογράφημα κωδικοποιείται μέσα στον κώδικα προκειμένου σε μελλοντικό χρόνο να εξαχθεί. Αυτές είναι οι ακόλουθες:

- Στατικά Υδατογραφήματα
  - **Υδατογραφήματα κώδικα:** (code watermarks) Τα υδατογραφήματα αυτά ενσωματώνονται στον κώδικα τροποποιώντας ορισμένες από τις ιδιότητές του. Ένα απλό παράδειγμα είναι η αλλαγή στη σειρά εμφάνισης των τμημάτων κώδικα προσθέτοντας εντολές τύπου goto.
  - **Υδατογραφήματα δεδομένων:** (data watermarks) Τα υδατογραφήματα αυτά ενσωματώνονται στα δεδομένα που χρησιμοποιεί ο κώδικας, τα οποία βρίσκονται εξ αρχής μέσα σε αυτόν. Για παράδειγμα στις μεταβλητές οι οποίες εμπεριέχουν συμβολοσειρές ή στα δεδομένα τα οποία είναι απαραίτητα για την αποσφαλμάτωση (debugging).
- Δυναμικά Υδατογραφήματα
  - **Αμεσα αντιληπτά υδατογραφήματα:** (Easter eggs watermarks) Τα υδατογραφήματα αυτά δοθείσας μίας συγκεκριμένης ακολουθίας εισόδου, η οποία αποτελεί και το κλειδί του υδατογραφήματος, πραγματοποιούνται ενέργειες απευθείας αντιληπτές από τον χρήστη. Δεν υπάρχει όμως κάποιο συγκεκριμένο αντικείμενο μέσα στον κώδικα του λογισμικού που αυτά ενσωματώνονται. Τέτοιου είδους ενέργειες είναι για παράδειγμα αφού πληκτρολογηθεί η ακολουθία κλειδί να εμφανιστεί ένα μήνυμα το οποίο να λέει πως το λογισμικό είναι πνευματικά κατοχυρωμένο.
  - **Υδατογραφήματα δυναμικών δομών δεδομένων:** (dynamic data structure watermarks) Αυτά τα υδατογραφήματα ενσωματώνονται στις δομές που δημιουργούνται κατά την διάρκεια εκτέλεσης του κώδικα δοθείσας μίας συγκεκριμένης ακολουθίας εισόδου. Το υδατογράφημα εξάγεται, εξετάζοντας τις τρέχουσες τιμές ορισμένων για παράδειγμα μεταβλητών κατά την εκτέλεση του λογισμικού με τη βοήθεια είτε ενός παρακολουθητή (profiler), είτε ενός αποσφαλματωτή (debugger).
  - **Υδατογραφήματα δυναμικού ίχνους εκτέλεσης:** (dynamic execution trace watermarks) Αυτά τα υδατογραφήματα ενσωματώνονται στο ίχνος που δημιουργεί η εκτέλεση του κώδικα, ύστερα από μία ακολουθία εισόδου κλειδί. Συγκεκριμένα, οι διευθύνσεις μνήμης ή οι εντολές που εκτελέστηκαν εξετάζονται στο σύνολό τους προκειμένου από μέσα αυτές να αναγνωριστεί το κρυμμένο υδατογράφημα. Ένα ωραίο παράδειγμα υδατογραφημάτων με αυτήν την ιδιότητα, είναι εκείνα τα υδατογραφήματα τα οποία κωδικοποιούνται σε μία αλληλουχία βασικών εντολών κατά την διάρκεια εκτέλεσης οι οποίες είναι ανεξάρτητες από την γλώσσα προγραμματισμού που χρησιμοποιείται. Όπως είναι οι κλήσεις συναρτήσεων, ή οι συνθήκες ελέγχου.

Ύστερα από μελέτη ετών η οποία έχει διεξαχθεί στον τομέα της υδατογράφησης λογισμικού, έχει αποδειχθεί πως τα υδατογραφήματα τα οποία έχουν την στατική ιδιότητα υπολείπονται

αυτών που έχουν την δυναμική καθώς είναι ιδιαιτέρως ευαίσθητα σε μετασχηματισμούς σημασιολογικών δομών του κώδικα. Από τη στιγμή όπου τέτοιου είδους μετασχηματισμούς μπορεί να εφαρμόσει ο καθένας σε ένα λογισμικό, απλά με την χρήση ενός αυτόματου εργαλείου, είναι αρκετό για να μην χρησιμοποιούνται μόνα τους δίχως κάποιον συνδυασμό (π.χ., θωράκιση).



Σχήμα 2.2: Υδατογράφημα το οποίο φέρει τη στατική ιδιότητα και συγκεκριμένα ανήκει στα υδατογραφήματα κώδικα.

Από την άλλη πλευρά, τα υδατογραφήματα που έχουν την δυναμική ιδιότητα, ναι μεν είναι βρίσκονται τοποθετημένα μέσα στο λογισμικό όπως και τα στατικά, αλλά προκειμένου να κατασκευάσουν το υδατογράφημα πρέπει να γίνει η εκτέλεση του λογισμικού με μία συγκεκριμένη ακολουθία εισόδου. Και πάλι όμως, εξαιρώντας τα υδατογραφήματα που έχουν την ιδιότητα των Easter eggs, τα υδατογραφήματα με την δυναμική ιδιότητα δεν παράγουν ως αποτέλεσμα κάτι διαφορετικό από αυτό που θα παρήγαγε το μη υδατογραφημέ-

νο λογισμικό με την ίδια είσοδο. Το μόνο πράγμα που αλλάζει, είναι πως το επιδιωκόμενο αποτέλεσμα έρχεται εις πέρας με ενέργειες οι οποίες στο εσωτερικό τους κωδικοποιούν τη δομή υδατογράφησης (π.χ., το διάγραμμα ελέγχου ροής κάθε μίας από τις συναρτήσεις που εκτελούνται δημιουργεί συγκεκριμένο γράφημα τύπου μεταθετικού). Με απλά λόγια δηλαδή, η στατική ιδιότητα των υδατογραφημάτων έχει τα θεμέλιά της στις ιδιότητες που χαρακτηρίζουν τον κώδικα, ενώ η δυναμική ιδιότητα των υδατογραφημάτων έχει τα θεμέλιά της στα χαρακτηριστικά και στις ιδιότητες των δομών που δημιουργούνται κατά την διάρκεια εκτέλεσης.

Η δυναμική ιδιότητα με τα ως τώρα πεπραγμένα στην προστασία των πνευματικών δικαιωμάτων στα λογισμικά φαίνεται να είναι μία πρόκληση καθώς προσφέρει περισσότερη ασφάλεια έναντι επιθέσεων από κακόβουλους χρήστες, αλλά από τη στιγμή που χειρίζεται ενέργειες οι οποίες πραγματοποιούνται αφού εκτελεστεί η κώδικας και οι οποίες αφού τροποποιηθούν προφανώς πρέπει να διατηρούν την λειτουργικότητα του λογισμικού καθιστά την εφαρμογή υδατογράφησης χρονοβόρα διαδικασία. Αυτό διότι, πριν ενσωματωθεί ένα δυναμικό υδατογράφημα πρέπει προηγουμένως ο κώδικας να αναλυθεί από πολλές οπτικές γωνίες με στόχο να εντοπιστούν τα σημεία τα οποία προσφέρονται για τροποποίηση καθώς επίσης πρέπει να αναλυθούν και οι δομές που δημιουργούνται κατά την διάρκεια εκτέλεσης προκειμένου να επιλεγεί υδατογράφημα το οποίο δεν τους επιφέρει την μικρότερη δυνατή αναστάτωση.

2.1.3 Χαρακτηριστικά Αξιολόγησης Υδατογραφημάτων Λογισμικού

Εφαρμόζοντας μία τεχνική υδατογράφησης προκειμένου να ενσωματωθεί ένα υδατογράφημα μέσα σε ένα λογισμικό, δημιουργούνται κάποια ερωτήματα σχετικά με την ορθότητα, την ασφάλεια και την επιβάρυνση που επιφέρει μία τέτοια ενέργεια. Είναι δεδομένο, πως αν ένας εισβολέας, έχει απεριόριστο χρόνο και εργαλεία στη διάθεσή του μετά βεβαιότητας μπορεί να αλλοιώσει οποιαδήποτε προστασία έχει δοθεί σε ένα λογισμικό. Το γεγονός αυτό όμως μοιάζει κάπως ουτοπικό, καθώς και σπανίως κάποιος έχει απεριόριστο χρόνο και σπανίως έχει και γνωρίζει να χρησιμοποιεί καταλλήλως τέτοιου είδους εργαλεία. Εν τούτης, μία και μόνο επιτυχής επίθεση κακόβουλου χρήστη αρκεί για να αφαιρέσει τα πνευματικά κατοχυρωμένα δικαιώματα που είχαν ενσωματωθεί σε ένα λογισμικό και να το διαθέσει (π.χ., μέσω του διαδικτύου) ελεύθερα στο κοινό.

Το γεγονός αυτό, απαιτεί την ενδελεχή αξιολόγηση των τεχνικών υδατογράφησης λογισμικού σύμφωνα με χαρακτηριστικά, τα οποία το κάθε ένα βάζει το δικό του λιθαράκι σε μία σφαιρική προστασία λογισμικού απαλλαγμένη από τρωτά σημεία τα οποία θα μπορούσαν να αποβούν μοιραία σε χέρια κακόβουλων χρηστών. Αυτά τα χαρακτηριστικά αξιολόγησης είναι τα εξής:

- **Ρυθμός δεδομένων** (Data rate): Αυτό το χαρακτηριστικό εκφράζει την αναλογία μεγέθους του υδατογραφήματος συγκριτικά με τον κώδικα προς υδατογράφηση. Ουσιαστικά είναι το κλάσμα με αριθμητή το μέγεθος σε bits του μη υδατογραφημένου

λογισμικού και παρονομαστή το μέγεθος σε bits του υδατογραφήματος. Όσο μεγαλύτερο είναι το κλάσμα αυτό, τόσο καλύτερη είναι η υδατογράφηση.

- **Αντίσταση στην Αναγνώριση** (Resistance to detection): Το χαρακτηριστικό αυτό αναφέρεται στις ενδεχόμενες στατιστικές ιδιότητες που έχει το υδατογραφημένο πρόγραμμα, τις οποίες τα συνήθη προγράμματα δεν τις έχουν.

- **Αντίκτυπο ενσωμάτωσης** (Embedding overhead): Το χαρακτηριστικό αυτό εκφράζει την επιπρόσθετη επιβάρυνση που έχει το υδατογραφημένο λογισμικό συγκριτικά με το μη υδατογραφημένο. Η επιβάρυνση αυτή αφορά τον μέσο χρόνο εκτέλεσης, τον χώρο μόνιμης αποθήκευσης (π.χ., στον σκληρό δίσκο) και τον χώρο προσωρινής φόρτωσης κατά την διάρκεια εκτέλεσης στην μνήμη τυχαίας προσπέλασης.

- **Αντίσταση έναντι μετασχηματισμών** (Resistance against transformations): Αυτό το χαρακτηριστικό δηλώνει την δυνατότητα ενός υδατογραφήματος να αντιστέκεται ύστερα από μετασχηματισμούς κώδικα όπως είναι η συσκότιση και η βελτιστοποίηση.

- **Τμηματική προστασία** (Part protection): Αυτό το χαρακτηριστικό αναφέρεται στην έκταση που καταλαμβάνει ο κώδικας του υδατογραφήματος. Όσο περισσότερο έχει διασκορπιστεί ο κώδικας αυτός τόσο δυσκολότερα κάποιος κακόβουλος χρήστης θα μπορέσει να τον εντοπίσει.

- **Αντίσταση έναντι αλλαγής της γλώσσας προγραμματισμού** (Resistance against changing programming language): Το χαρακτηριστικό αυτό αναφέρεται στην ανεξαρτησία του υδατογραφήματος από την γλώσσα προγραμματισμού με την οποία έχει κωδικοποιηθεί το λογισμικό. Ένα υδατογράφημα είναι καλό και απολύτως επιθυμητό να είναι ανεξάρτητο, όχι μόνο από το υλικό ενός ηλεκτρονικού υπολογιστή, αλλά και από την γλώσσα που χρησιμοποιείται.

- **Αξιοπιστία** (Credibility): Αυτό το χαρακτηριστικό εκφράζει τα ποσοστά της λανθασμένης αναγνώρισης (false positive) και της λανθασμένης μη αναγνώρισης (false negative) από τον κώδικα ενός υδατογραφημένου λογισμικού. Όσο πιο χαμηλά κυμαίνονται αυτά τα ποσοστά τόσο αποτελεσματικότερη είναι η υδατογράφηση.

- **Πιστότητα** (Fidelity): Το χαρακτηριστικό αυτό δηλώνει το ποσοστό της λειτουργικότητας του υδατογραφημένου κώδικα. Ένα από τα προαπαιτούμενα της υδατογράφησης λογισμικού είναι ο υδατογραφημένος κώδικας να παράγει τα ίδια ακριβώς αποτελέσματα για κάθε ακολουθία εισόδου με τον μη υδατογραφημένο κώδικα. Αυτό όμως σε περιπτώσεις κατά τις οποίες το ενσωματωμένο υδατογράφημα έχει την ιδιότητα των δυναμικών υδατογραφημάτων δεν είναι τετριμμένο πως ισχύει, καθώς οι ενέργειες οι οποίες πραγματοποιούνται προκειμένου να κατασκευαστεί το υδατογράφημα πιθανόν σε μερικές περιπτώσεις να δημιουργούν λογικά λάθη στον κώδικα.

Μέχρι στιγμής δεν έχει βρεθεί κάποια μαγική συνταγή, τέτοια ώστε να κατασκευάζει υδατογράφημα ενσωματώνοντάς το σε ένα λογισμικό και να έχει όλα τα χαρακτηριστικά

αξιολόγησης στο μέγιστο δυνατό. Όλες οι τεχνικές κάπου κερδίζουν και κάπου χάνουν, γι αυτό άλλωστε η προστασία λογισμικού και γενικότερα η προστασία πνευματικών δικαιωμάτων είναι μία σύνθεση ιδεών και συγκεκριμένα τεχνικών η οποία στόχο έχει να αποθαρρύνει την πειρατεία καθιστώντας την αποδέσμευση από τυχόν ενοχοποιητικά στοιχεία επίπονη και χρονοβόρα διαδικασία.

2.2 Αλγοριθμικές Τεχνικές Υδατογράφησης Λογισμικού

Ως αλγόριθμος ορίζεται μια σειρά πεπερασμένων ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, με στόχο την επίλυση ενός προβλήματος. Ο αλγόριθμος θα πρέπει να πληρεί κάποια πρότυπα και να διατυπώνεται με συγκεκριμένο τρόπο. Επομένως αυτός πρέπει να ικανοποιεί τα επόμενα κριτήρια:

- **Καθοριστικότητα:** Κάθε κανόνας του αλγορίθμου πρέπει να ορίζεται επακριβώς και η αντίστοιχη διεργασία είναι συγκεκριμένη. Κάθε εντολή πρέπει να καθορίζεται χωρίς καμία αμφιβολία για τον τρόπο εκτέλεσής της λαμβάνοντας υπόψη όλα τα πιθανά σενάρια.

- **Περατότητα:** Κάθε εκτέλεσή του είναι πεπερασμένη, τελειώνοντας ύστερα από έναν πεπερασμένο αριθμό διεργασιών ή βημάτων. Ίλλως, στην περίπτωση κατά την οποία η διαδικασία δεν τελειώνει μετά από συγκεκριμένο / πεπερασμένο αριθμό βημάτων θα είναι απλά μία υπολογιστική διαδικασία.

- **Αποτελεσματικότητα:** Ο αλγόριθμος πρέπει να είναι μηχανιστικά αποτελεσματικός. Ουσιαστικά δηλαδή όλες του οι διαδικασίες που περιλαμβάνονται σε αυτόν να μπορούν να πραγματοποιούνται με ακρίβεια και σε πεπερασμένο χρόνο. Κάθε μεμονωμένη εντολή του πρέπει να είναι απλή και όχι σύνθετη. Καθώς μία εντολή δεν αρκεί να έχει ορισθεί, αλλά πρέπει να είναι και εκτελέσιμη.

- **Επεκτασιμότητα:** Κατά την εκκίνηση εκτέλεσής του καμία, μία ή περισσότερες τιμές δεδομένων πρέπει να δίνονται ως είσοδοι σε αυτόν. Η περίπτωση όπου δε δίνονται τιμές δεδομένων εμφανίζεται όταν ο αλγόριθμος δημιουργεί και επεξεργάζεται κάποιες πρωτογενείς τιμές με τη βοήθεια συναρτήσεων παραγωγής τυχαίων αριθμών ή με τη βοήθεια άλλων εντολών.

- **Να έχει είσοδο δεδομένων, επεξεργασία και έξοδο αποτελεσμάτων:** Δίδει τουλάχιστον ένα μέγεθος ως αποτέλεσμα που εξαρτάται κατά κάποιο τρόπο από τις αρχικές εισόδους του.Ο αλγόριθμος πρέπει να δημιουργεί τουλάχιστον μία τιμή δεδομένων ως αποτέλεσμα προς το χρήστη ή προς έναν άλλο αλγόριθμο.

Προκειμένου να ενσωματωθεί μία δομή στο εσωτερικό ενός κώδικα λογισμικού πρέπει να ακολουθηθούν διακριτά βήματα τα οποία είναι ρητώς διατυπωμένα στα πλαίσια μίας τεχνικής αποφεύγοντας άσκοπες ενέργειες οι οποίες θα μπορούσαν να καταστρέψουν την

λειτουργικότητα του κώδικα. Κατά καιρούς έχει διατυπωθεί μία πληθώρα τέτοιων τεχνικών οι οποίες με τα πλεονεκτήματα και τα μειονεκτήματα που τις χαρακτηρίζουν έχουν προσπαθήσει να εκμεταλλευτούν, με την καλή έννοια της λέξης, συστατικά από τους κώδικες των λογισμικών με σκοπό να τα τροποποιήσουν ορθά και αποτελεσματικά κωδικοποιώντας ένα υδατόσημα σε αυτά. Οι τεχνικές αυτές λοιπόν είναι οι εξής:

- **Αφηρημένος αλγόριθμος διερμηνευτή** (Abstract interpretation algorithms): Η τεχνική αυτή εισάγει το υδατογράφημα σε ακέραιες τοπικά ορισμένες μεταβλητές. Συγκεκριμένα, εντοπίζεται μία συνάρτηση η οποία περιέχει δομή επανάληψης στην οποία και θα κωδικοποιηθεί το υδατογράφημα. Το υδατογράφημα μπορεί να είναι από μία σταθερά μέχρι και ένα πολυώνυμο. Απλά για να μην είναι το υδατογράφημα ευάλωτο σε τροποποιήσεις συνήθως προτιμάται το πολυώνυμο του οποίου η επιλογή γίνεται με τέτοιον τρόπο έτσι ώστε το υπόλοιπο της ακέραιας διαίρεσής του με το κλειδί υδατογράφησης να δίνει την τιμή του υδατογραφήματος [31].

- **Αλγόριθμος αναδιάταξης κώδικα** (Code reordering algorithms): Το υδατογράφημα με αυτήν την τεχνική ενσωματώνεται στο λογισμικό αναδιατάσσοντας την σειρά με την οποία εμφανίζονται τα τμήματα (blocks) του γραφήματος ελέγχου ροής του κώδικα προσθέτοντας εντολές τύπου goto προκειμένου να διατηρείται η λειτουργικότητα του κώδικα [33].

- **Αλγόριμος βασισμένος σε περιορισμούς** (Constraint based algorithms): Η τεχνική αυτή αναθέται έναν μεγάλο αριθμό μεταβλητών του προγράμματος σε ένα μικρό πλήθος καταχωρητών. Η διαδικασία αυτή μπορεί να γίνει πράξη ή πάνω σε ένα βασικό τμήμα όπου στην περίπτωση αυτή ονομάζεται local register allocation ή πάνω σε μία συνάρτηση όπου στην περίπτωση αυτή ονομάζεται global register allocation ή ακόμα και μεταξύ συναρτήσεων όπου σε αυτήν την περίπτωση ονομάζεται interprocedural register allocation [55].

- **Αλγόριθμος δυναμικού μονοπατιού** (Dynamic path based algorithms): Το υδατογράφημα με την τεχνική αυτή ενσωματώνεται στην εκτελέσιμη διακλαδώμενη δομή του κώδικα. Συγκεκριμένα, από τη στιγμή που οι διακλαδώσεις (συνθήκες ελέγχου) οι οποίες εκτελούνται σε ένα πρόγραμμα είναι εκ φύσεώς τους δυαδικές (αληθής ή ψευδής) μπορούν και κωδικοποιούνται σε αυτές δυαδικές συμβολοσειρές. Στην ουσία η τεχνική καθορίζει, δοθείσας μία ακολουθίας εισόδου, την δυναμική συμπεριφορά του μη υδατογραφημένου προγράμματος παρακολουθώντας τα μονοπάτια εκτέλεσης μέσω ενός παρακολουθητή (profiler) με σκοπό να εντοπίσει κατάλληλα σημεία ενσωμάτωσης και να προσθέσει ή να τροποποιήσει τις ήδη υπάρχουσες συνθήκες ελέγχου [21].

- **Αλγόριθμος βασισμένος σε γραφήματα** (Graph based algorithms): Η τεχνική αυτή ενσωματώνει γραφήματα στους κώδικες των λογισμικών τροποποιώντας τα διαγράμματα ελέγχου ροής. Ως επί το πλείστον, τέτοια γραφήματα είναι τα προσανατολισμένα με δείκτη στον πατέρα δέντρα (oriented parent pointer trees), τα γραφήματα με βάση k (radix - k graphs), τα μεταθετικά γραφήματα (permutation graphs),

τα επιπεδικά κυβικά δέντρα (planted planar cubic trees) καθώς και τα αναγώγιμα μεταθετικά γραφήματα (reducible permutation graphs) [22].

- **Αδιαφανές αλγόριθμος κατηγορήματος** (Opaque predicates algorithms): Το υδατογράφημα με αυτού του είδους την τεχνική, ενσωματώνεται στο κώδικα του λογισμικού προσθέτοντας συνθήκες ελέγχου οι οποίες αποτιμούν αδιαφανή κατηγορήματα. Συγκεκριμένα σε κάθε μία τέτοια συνθήκη ελέγχου κωδικοποιείται ένα τμήμα του υδατογραφήματος [25].

- **Αλγόριθμος εκτεταμένου φάσματος** (Spread spectrum algorithms): Η τεχνική αυτή, δεν χειρίζεται το λογισμικό ως μία ακολουθία εντολών (όπως κάνουν οι περισσότερες των τεχνικών) αλλά ως ένα συμπαγές αντικείμενο το οποίο περιέχει συχνότητες εμφάνισης ομάδων εντολών προσπαθώντας να τις τροποποιήσει για να πραγματοποιήσει την ενσωμάτωση. Η τροποποίηση των συχνοτήτων γίνεται αντικαθιστώντας ομάδες εντολών με άλλες ισοδύναμες [59].

- **Αλγόριθμος βασισμένος σε νήματα** (Thread based algorithms): Η τεχνική αυτή ενσωματώνει υδατογραφήματα προσθέτοντας νήματα ή τροποποιώντας τις συμπεριφορές των ήδη υπαρχόντων που χειρίζεται ο κώδικας του λογισμικού [50].

Η τεχνική υδατογράφησης ονόματι Code reordering algorithms χωρίζεται σε επιμέρους τεχνικές οι οποίες ανάλογα με τα αντικείμενα που χειρίζονται και αναδιατάσσουν προκειμένου να ενσωματώνουν το υδατογράφημα παίρνουν και το αντίστοιχο όνομα. Συγκεκριμένα, αν η τεχνική αναδιατάσσει τμήματα κώδικα ονομάζεται Basic block reordering, αν αναδιατάσσει εξισώσεις ονομάζεται Equation reordering, ενώ αν αναδιατάσσει σταθερές στον χώρο αποθήκευσής τους ονομάζεται Constant pool reordering. Όποια όμως από τις τεχνικές της αναδιάταξης να χρησιμοποιηθεί με σκοπό να ενσωματώσει κάποια δομή μέσα σε κώδικα, εύκολα μπορεί να αλλοιωθεί κάνοντας χρήση απλών σημασιολογικών μετασχηματισμών και μόνο.

Επιπλέον η τεχνική Thread based algorithms έχει ένα σοβαρό μειονέκτημα το οποίο έρχεται στην επιφάνεια όταν προκειμένου να ενσωματωθεί ένα μεγάλο υδατογράφημα, ενσωματώνονται πολλά νήματα μέσα στον κώδικα. Με τον τρόπο αυτόν ο κώδικας γίνεται ιδιαίτερα δύσχρηστος στην ανάλυση και στην επεκτασιμότητά του καθώς και φοβερά αργός καθώς τα νήματα από τη φύση τους λειτουργούν παράλληλα και δύσκολα χειραγωγούνται όπως ακριβώς κάποιος προγραμματιστής επιθυμεί. Έχουν βέβαια προτεραιότητες (priorities) όπου για παράδειγμα στη γλώσσα προγραμματισμού Java είναι από το 1 (χαμηλή) έως και το 10 (υψηλή) αλλά ως επί το πλείστον δεν τηρούνται πιστά από τα λειτουργικά συστήματα στα οποία αυτές, μέσω των νημάτων, λαμβάνουν χώρα.

2.3 Μεθοδολογία Κατασκευής και Ενσωμάτωσης Υδατογραφημάτων

Η υδατογράφηση λογισμικού είναι μία διαδικασία η οποία απαιτεί κατά κύριο λόγω φαντασία προκειμένου να γίνει σωστά και κυρίως αποτελεσματικά. Ο λόγος για τον οποίο συμβαίνει

αυτό είναι ιδιαίτερα προφανής καθώς για να προστατέψει κανείς, όχι μόνο κάποιο λογισμικό, αλλά και γενικά κάτι το οποίο θέλει να κρατήσει μακριά από την εκμετάλλευση τρίτων χρειάζεται άπλετη φαντασία περί των κινήσεων οι οποίες θα μπορούσαν να αποφέρουν βλάβες σοβαρές και μη σε αυτό. Πρέπει επί της ουσίας κάποιος να είναι σε θέσει να μαντεύει κατά κάποιον τρόπο τις κινήσεις οι οποίες εκμεταλλεύονται τρωτά σημεία του προγράμματος και δίνουν την δυνατότητα σε κακόβουλους χρήστες να αλωνίζουν σε βάρος των πνευματικών δικαιωμάτων των ιδιοκτητών.

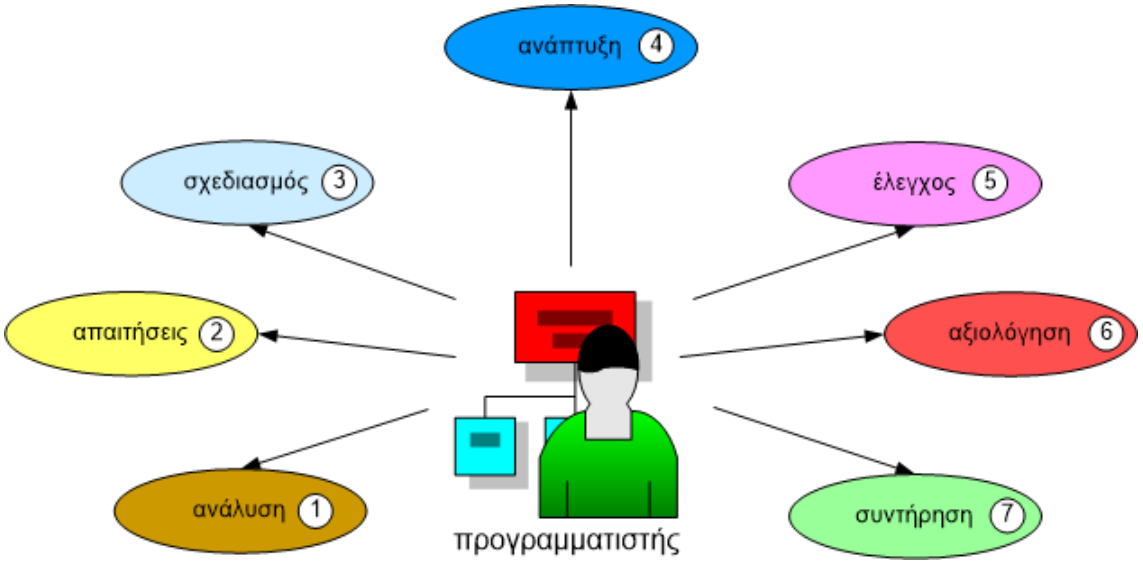
Από τη στιγμή όπου η υδατογράφηση λογισμικού χωρίζεται σε δύο κύριες φάσεις, την φάση της κατασκευής του υδατογραφήματος και την φάση της ενσωμάτωση του μέσα στον κώδικα του λογισμικού γίνεται φανερό πως δεν υπάρχει απαραίτητα κάποιου είδους εξάρτηση μεταξύ τους. Το τι δομή θα δημιουργήσει κάποιος θέλοντας να την προσθέσει μέσα σε ένα λογισμικό δεν συνεπάγεται και συγκεκριμένη τεχνική που θα πρέπει να χρησιμοποιήσει. Το σίγουρο είναι πως η δομή στην οποία κάποιος θα καταλήξει θα πρέπει να έχει ιδιότητες οι οποίες να τις δίνουν ταυτότητα και ελευθερία στις τροποποιήσεις χωρίς να αλλοιώνεται. Καθώς όταν θα βρίσκεται ενσωματωμένη μέσα στον κώδικα, με μεγάλη πιθανότητα θα υποστεί τροποποιήσεις είτε καλόβουλες από αυτούς που επιθυμούν να επεκτείνουν τον κώδικα του λογισμικού, είτε κακόβουλες από εισβολείς οι οποίοι επιθυμούν να απαλλαγούν από τυχόν προστασίες.

Εν αρχή, κανείς δεν είναι σε θέση να υδατογραφήσει κάποιο λογισμικό αν πρώτα δεν έχει καταλάβει πως αυτό λειτουργεί. Αυτό σημαίνει πως προηγουμένως αυτός που επιθυμεί να υδατογραφήσει κάποιο λογισμικό πρέπει να μελετήσει εις βάθος την φιλοσοφία του κώδικά του και να εξάγει πληροφορίες οι οποίες θα του φανούν χρήσιμες κατά την διαδικασία της υδατογράφησης. Οι πληροφορίες αυτές όσο περισσότερες και όσο πιο ευρύ φάσμα καλύπτουν είναι τόσο το καλύτερο για αυτόν. Χάρη σε αυτές θα μπορέσει να κατασκευάσει ένα υδατογράφημα το οποίο να ενδείκνυται για ενσωμάτωση σε ένα λογισμικό, καθώς δεν σημαίνει αναγκαστικά πως μία δομή υδατογράφησης μέσω μίας τεχνικής ότι μπορεί να προστεθεί σε οποιοδήποτε λογισμικό.

Ένα απλό παράδειγμα τέτοιων πληροφοριών είναι σε ποια γλώσσα προγραμματισμού είναι κωδικοποιημένος ο κώδικας, ποιος είναι ο μέσος χρόνος εκτέλεσης με ένα ορισμένο πλήθος διαφορετικών ακολουθιών εισόδου, πόσες συναρτήσεις περιέχει και πόσες από αυτές εκτελούνται κατά μέσο όρο με μία ακολουθία εισόδου, πόσο μεγάλες είναι σε όγκο (αριθμός εντολών) αυτές, πόσα και τι είδους ορίσματα αυτές δέχονται, τι επιστρέφουν, πόσες τοπικές μεταβλητές χρησιμοποιούν και πόσα διαφορετικά μονοπάτια εκτέλεσης υπάρχουν σε κάθε μία από αυτές τις συναρτήσεις. Εν συνεχεία πάνω στο σύνολο των πληροφοριών αυτών, αυτός που πρόκειται να εφαρμόσει την υδατογράφηση θα πατήσει με στόχο να δημιουργήσει ένα υδατογράφημα και να το ενσωματώσει όσο πιο κρυφά και έξυπνα γίνεται.

Μετέπειτα απαραίτητο είναι να διατυπωθεί πως η τεχνική ή αλλιώς το μοντέλο υδατογράφησης που έχει στο μυαλό του, αυτός που επιθυμεί να προστατέψει ένα λογισμικό, λειτουργεί σωστά στο πρόγραμμα που θέλει να το εφαρμόσει. Με την φράση λειτουργεί σωστά, δύναται να εννοηθεί πως το υδατογραφημένο πρόγραμμα να λειτουργεί πανομοιότυπα με το μη υδατογραφημένο, χωρίς να κρεμά για ένα σύνολο ακολουθιών εισόδου και είτε να

παράγει λανθασμένα αποτελέσματα, είτε να μην μπορεί να εκτελεστεί. Με απλά λόγια, και τα δύο προγράμματα να παράγουν τα ίδια ακριβώς αποτελέσματα για όλες τις πιθανές ακολουθίες εισόδου και στον ίδιο περίπου χρόνο. Στην περίπτωση όπου οι ακολουθίες εισόδου για ένα λογισμικό τείνουν στο άπειρο ή είναι πάρα πολλές (σύνηθες φαινόμενο για λογισμικά μεγάλης κλίμακας), αποδεικνύεται η ορθότητα του μοντέλου χωρίς να χρειαστεί να εκτελεστεί το υδατογραφημένο πλέον πρόγραμμα για όλες τις πιθανές ακολουθίες εισόδου του χάνοντας ενδεχομένως πολύτιμο χρόνο για εξαγωγή του στην αγορά (time to market).



Σχήμα 2.3: Μεθοδολογία ανάπτυξης και εφαρμογής μοντέλου υδατογράφησης λογισμικού.

Το τελικό στάδιο πριν αρχίσει η εφαρμογή του μοντέλου στον κώδικα, είναι να ψάξει αυτός που θα υδατογραφήσει το λογισμικό για πιθανές διορθώσεις του μοντέλου οι οποίες αν πραγματοποιηθούν, δώσουν καλύτερα αποτελέσματα προστασίας. Τα αποτελέσματα αυτά μπορεί να είναι λόγου χάρη λιγότερος αποθηκευτικό χώρος, γρηγορότερος χρόνος εκτέλεσης και λιγότερο εμφανές το ενσωματωμένο υδατογράφημα από πιθανούς εισβολείς. Πάρα ταύτα οι διορθώσεις αυτές δεν είναι απαραίτητο να συμβαίνουν πάντα, σε κάθε περίπτωση όμως αυτός που επιθυμεί να προσδώσει στο λογισμικό καλύτερη ασφάλεια και προστασία των πνευματικών δικαιωμάτων πρέπει να αφιερώνει χρόνο για τυχόν τέτοιες διορθώσεις. Για αυτό άλλωστε και οι πειρατεία λογισμικού συνεχώς αυξάνεται, καθώς τα μοντέλα υδατογράφησης λογισμικού που κατασκευάζονται όχι μόνο χρήζουν περισσότερης βελτίωσης, αλλά και γενικά ολόκληρος ο τομέας προστασίας λογισμικού χρήζει περισσότερης προσοχής.

Εν τέλει αφότου ολοκληρωθούν οι παραπάνω ενέργειες το μοντέλο εφαρμόζεται στο λογισμικό βήμα προς βήμα, χωρίς να θεωρείται τίποτα δεδομένο εξ αρχής και χωρίς να παραλείπονται ή να εφαρμόζονται με διαφορετική σειρά από αυτήν που επιβάλει το μοντέλο, τα βήματα της υδατογράφησης. Όταν ολοκληρωθεί η εφαρμογή του μοντέλου στο πρόγραμ-

μα, αυτό είναι έτοιμο πλέον να βγει στην αγορά και να δεχτεί ποικίλες επιθέσεις από κακόβουλους χρήστες, καθώς αυτός άλλωστε είναι και ο κύριος αντίπαλος της υδατογράφησης που πρέπει να αποκρούσει, οι επιθέσεις κακόβουλων χρηστών. Όσα περισσότερα είδη επιθέσεων μπορεί να αποκρούσει ένα μοντέλο υδατογράφησης, τόσο καλύτερο θεωρείται.

2.4 Επιθέσεις σε Υδατογραφήματα Λογισμικού

Ένα λογισμικό όταν βγει στη αγορά, είναι πολύ πιθανόν να δεχθεί διάφορες επιθέσεις από κακόβουλους χρήστες οι οποίοι θα προσπαθήσουν μέσα από αυτές να καταστρέψουν κάθε είδους ασφάλειας που πιθανόν να έχει. Οι λόγοι για τους οποίους γίνεται αυτό από χρήστες των ηλεκτρονικών υπολογιστών είναι είτε οικονομικοί προκειμένου να βγάλουν χρήματα μέσα από μία τέτοια διαδικασία, είτε ψυχολογικοί για λόγους επιβεβαίωσης.

Στην γενική περίπτωση οι κακόβουλοι χρήστες δεν λειτουργούν τυχαία όταν έχουν στα χέρια τους κάποιο λογισμικό. Απεναντίας δρουν μεθοδευμένα, ακολουθώντας διακριτά βήματα προκειμένου να πετύχουν τον στόχο τους. Στην αρχή πράττουν ενέργειες οι οποίες ολοκληρώνονται σχετικά γρήγορα, προκειμένου να πάρουν μία πρώτη άποψη περί τίνος πρόκειται να αντιμετωπίσουν. Μετά, αφού έχουν βάλει στο μυαλό τους κάποιες βασικές πληροφορίες (π.χ., ποια είναι η λειτουργία του λογισμικού, ποια είναι τα τρωτά του σημεία, ποιο είδος ασφάλειας το προαστεύει) διεξάγουν ενέργειες οι οποίες είναι πλέον στοχευμένες και διαρκούν περισσότερη ώρα από τις αρχικές προκειμένου να αλλοιώσουν την υποτιθέμενη προστασία. Και ύστερα δοκιμάζουν (εκτελούν) το λογισμικό για να καταλάβουν αν οι ενέργειές τους πέτυχαν ή όχι. Όλη αυτή η διαδικασία προφανώς γίνεται επαναληπτικά μέχρις ότου βεβαιωθούν σε βαθμό όπου θα είναι ικανοποιημένοι πως έχουν πετύχει αυτό που ήθελαν.

Όπως και στις τεχνικές υδατογράφησης λογισμικού, έτσι και στις επιθέσεις σε λογισμικά υπάρχουν διάφοροι τύποι επιθέσεων οι οποίοι μπορούν να εφαρμοστούν έχοντας ως απώτερο σκοπό να δημιουργήσουν προβλήματα ασφάλειας. Οι τύποι αυτοί είναι οι εξής:

- **Προσθετικές επιθέσεις** (Additive attacks): Με αυτήν την επίθεση ενσωματώνεται στον κώδικα ενός ήδη υδατογραφημένου λογισμικού νέο υδατογράφημα. Συνήθως αυτή η ενέργεια δεν δημιουργεί πρόβλημα στην περίπτωση κατά την οποία οι τροποποιήσεις που γίνουν δεν αφορούν τον κώδικα του παλιού υδατογραφήματος.

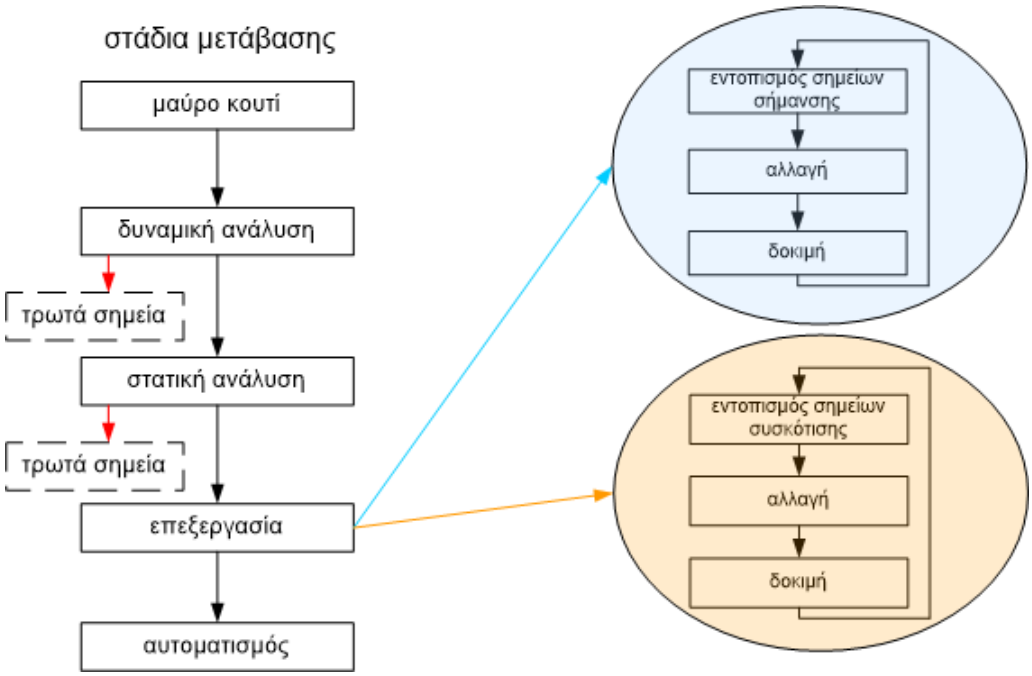
- **Καταστροφικές επιθέσεις** (Distortive attacks): Αυτή η επίθεση σχετίζεται με τροποποιήσεις σημασιλογικών δομών του κώδικα χρησιμοποιώντας είτε την τεχνική της συσκότισης είτε την τεχνική της βελτιστοποίησης. Οι ενέργειες αυτής της επίθεσης μπορούν να λάβουν χώρα σε συγκεκριμένα σημεία ή σε ολόκληρη την έκταση του κώδικα ανάλογα με το αν οι εισβολείς έχουν ή όχι γνώση για το σημείο εφαρμογής της υδατογράφησης.

- **Αφαιρετικές επιθέσεις** (Substractive attacks): Με αυτήν την επίθεση αφαιρούνται τα τμήματα του κώδικα που αντιστοιχούν στο υδατογράφημα διορθώνοντας τυχόν

λάθη που έχουν ως αντίκτυπο την αλλοίωση της λειτουργικότητας τα οποία και δημιουργούνται από την αφαίρεση αυτή.

- **Επιθέσεις πρωτοκόλλου** (Protocol attacks): Αυτή η επίθεση δεν πειράζει καθόλου τον κώδικα του λογισμικού, παρά μόνο κατασκευάζει έναν ψεύτικο αναγνωριστή (recognizer) ο οποίος εξάγει ένα διαφορετικό υδατογράφημα μέσα από το λογισμικό.
- **Συμπαιγνίακές επιθέσεις** (Collusive attacks): Αυτή η επίθεση είναι αποτελεσματική μονάχα στην περίπτωση κατά την οποία το υδατογράφημα είναι τύπου fingerprinting mark. Συγκεκριμένα, όπως και στην protocol attack, έτσι και στην collusive attack δεν πειράζεται καθόλου ο κώδικας του λογισμικού απλά συγκρίνονται δύο ή περισσότερες εκδόσεις του ίδιου κώδικα με σκοπό να εντοπιστούν τα διαφορετικά σημεία τα οποία είναι και αυτά που κωδικοποιούν το ενσωματωμένο υδατογράφημα.

Σε μία πιθανή διαμάχη στα δικαστήρια κατά την οποία κάποιος κακόβουλος χρήστης έχει προσθέσει ένα καινούριο υδατογράφημα σε ένα λογισμικό, χωρίς να έχει δημιουργήσει κάποιο πρόβλημα στο παλιό, τότε ο πραγματικός ιδιοκτήτης έχοντας την έκδοση που έχει μονάχα το δικό του υδατογράφημα μέσα στον κώδικα μπορεί να ισχυριστεί πως αυτός είναι ο νόμιμος κάτοχος των πνευματικών δικαιωμάτων καθώς και στις δύο εκδόσεις (την αρχική που έχει ένα υδατογράφημα και την νέα που έχει δύο υδατογραφήματα) υπάρχει το δικό του υδατογράφημα.



Σχήμα 2.4: Στάδια από τα οποία μεταβαίνει κάποιος κακόβουλος χρήστης με σκοπό να εντοπίσει και να εξάγει υδατογραφήματα καθώς και να εξαλείψει συσκοτίσεις.

Επιπλέον αν κάποιος κακόβουλος χρήστης κατασκευάσει έναν ψεύτικο δικό του αναγνωριστή και ισχυριστεί και αυτός με τη σειρά του πως αυτός είναι ο νόμιμος ιδιοκτήτης, τότε χρειάζεται η επέμβαση ενός ιδικού στον τομέα των υδατογραφημάτων ο οποίος θα αποφανθεί για το ποιος ισχυρίζεται την αλήθεια και ποιος όχι. Πάρα ταύτα αυτός ο τύπος επιθέσεων αποτελεί ένα μεγάλο πρόβλημα για την υδατογράφηση λογισμικού καθώς ο οποιοσδήποτε μπορεί να κατασκευάζει ένα πρόγραμμα το οποίο θα το ονομάζει αναγνωριστή προκειμένου να ισχυρίζεται πως κάποιος κώδικας είναι δικός του.

2.5 Γραφήματα Παραγόμενα από Κώδικες Λογισμικών

Στα μαθηματικά, ένα γράφημα είναι μια αφηρημένη αναπαράσταση ενός συνόλου στοιχείων, όπου μερικά ζευγάρια στοιχείων συνδέονται μεταξύ τους με δεσμούς. Τα διασυνδεδεμένα στοιχεία αναπαριστώνται με μαθηματικές έννοιες οι οποίες ονομάζονται κορυφές, ενώ οι δεσμοί που συνδέουν τα ζευγάρια των κορυφών ονομάζονται ακμές. Συνήθως, ένα γράφημα απεικονίζεται σε διαγραμματική μορφή ως ένα σύνολο κουκκίδων για τις κορυφές, ενωμένα μεταξύ τους με γραμμές ή καμπύλες για τις ακμές. Οι ακμές μπορούν να είναι είτε κατευθυνόμενες, είτε μη κατευθυνόμενες.

Στην πιο κοινή έννοια του όρου, ένα γράφημα είναι ένα διατεταγμένο ζεύγος $G = (V, E)$ αποτελούμενο από ένα σύνολο V των κορυφών ή αλλιώς κόμβων μαζί με ένα σύνολο V από γραμμές ή αλλιώς ακμές, οι οποίες είναι υποσύνολα δύο στοιχείων V (δηλαδή, μια ακμή σχετίζεται με δύο κορυφές και η σχέση απεικονίζεται ως ένα ταξινομημένο ή μη ζεύγος των κορυφών σε σχέση με τη συγκεκριμένη ακμή. Το ζεύγος αυτό είναι ταξινομημένο στην περίπτωση που το γράφημα είναι κατευθυνόμενο, ενώ το ζεύγος είναι μη ταξινομημένο στην περίπτωση που το γράφημα είναι μη κατευθυνόμενο.

Από έναν κώδικα λογισμικού μπορούν να παραχθούν διάφορα γραφήματα στα οποία βρίσκεται πληροφορία κωδικοποιημένη σχετικά με το πως αυτό λειτουργεί κάτω από γενικές ή συγκεκριμένες συνθήκες. Τέτοια γραφήματα είναι τα εξής:

- **Γραφήματα κλήσεων συναρτήσεων:** Τα γραφήματα συναρτήσεων είναι εκείνα τα κατευθυνόμενα γραφήματα $G = \{V, E\}$ τα οποία αναπαριστούν τη σχέση κλήσεων (calling relationship) μεταξύ των συναρτήσεων F ενός προγράμματος. Συγκεκριμένα κάθε κόμβος v αναπαριστά μία συνάρτηση και κάθε ακμή $e = (f_i, f_j)$ αναπαριστά μία κλήση από την συνάρτηση f_i, η οποία αποτελεί την καλούσα συνάρτηση (caller function) στην συνάρτηση f_j, η οποία την κλειθήσα συνάρτηση (callee function). Ένας κύκλος $e = (f_i, f_i)$ στο γράφημα κλήσεων CG δηλώνει την αναδρομική κλήση από την συνάρτηση f_i στον εαυτό της. Τα γραφήματα αυτά χωρίζονται σε δύο κύριες κατηγορίες, τα στατικά γραφήματα κλήσεων (static call graphs - SCG) και τα δυναμικά γραφήματα κλήσεων (dynamic call graphs - DCG).

- **Γραφήματα ιεραρχίας κλήσεων:** Αυτά τα γραφήματα δηλώνουν ποιες κλήσεις συναρτήσεων ενός προγράμματος έχει η κάθε μία συνάρτηση στον κώδικά της. Κάθε κόμβος αποτελεί μία συνάρτηση και κάθε ακμή αποτελεί μία κλήση συνάρτησης.

- **Διάγραμμα ελέγχου ροής:** Τα γραφήματα αυτά αναπαριστούν όλα τα πιθανά μονοπάτια εκτέλεσης ενός προγράμματος. Συγκεκριμένα τα γραφήματα αυτά είναι κατευθυνόμενα, όπου κάθε κόμβος αυτών αναπαριστά ένα τμήμα κώδικα (block) το οποίο έχει μονάχα μία είσοδο και μονάχα μία έξοδο και κάθε ακμή αυτών αναπαριστά μία μεταφορά (jump) από το ένα τμήμα σε ένα άλλο.

- **Διαγράμματα κλάσεων:** Τα γραφήματα αυτά ομαδοποιούν συναρτήσεις σε κλάσεις (π.χ., στην γλώσσα προγραμματισμού Java χρησιμοποιούνται classes) όπου η κάθε κλάση αποτελεί έναν κόμβο και κάθε ένας από τους κόμβους αυτούς έχει κατευθυνόμενη ακμή σε κάποιον άλλον αν υπάρχει κάποιου είδους εξάρτηση από τον έναν κόμβο στον άλλον. Οι ακμές των γραφημάτων δεν είναι ενός είδους, αντιθέτως υπάρχουν διάφορες ακμές ανάλογα με την εκάστοτε εξάρτηση που υπάρχει.

Τα γραφήματα κλήσεων συναρτήσεων είναι αντικείμενα τα οποία χρησιμοποιούνται κατά κόρον για ανάλυση προγραμμάτων προκειμένου να γίνουν τα προγράμματα πιο κατανοητά από του ανθρώπους. Αποτελούν επίσης και βάση για περαιτέρω αναλύσεις, όπως η παρακολούθηση των τιμών που μεταβιβάζονται μεταξύ συναρτήσεων. Μία απλή εφαρμογή τέτοιων γραφημάτων είναι για να εντοπιστούν συναρτήσεις οι οποίες ποτέ δεν εκτελούνται ή συναρτήσεις οι οποίες δεν τροποποιούν τις τιμές που χειρίζονται.

Επιπλέον τα γραφήματα κλήσεων συναρτήσεων χωρίζονται σε στατικά και σε δυναμικά γραφήματα. Τα δυναμικά παράγονται ύστερα από μία εκτέλεση ενός προγράμματος δοθείσας μίας ακολουθίας εισόδου, ενώ τα στατικά αναπαριστούν κάθε πιθανή εκτέλεση ενός προγράμματος. Από τη στιγμή όμως, που δεν είναι δυνατόν στη γενική περίπτωση των προγραμμάτων να γνωστοποιηθούν όλες οι πιθανές ακολουθίες εισόδου ενός προγράμματος, η ακριβής κατασκευή ενός στατικού γραφήματος κλήσεων συναρτήσεων αποτελεί ένα δυσεπίλυτο πρόβλημα (undecidable problem).

2.6 Ανάλυση Κώδικα Λογισμικού

Μία επίθεση στον κώδικα ενός λογισμικού χαρακτηρίζεται ως επιτυχής όταν έχει καταφέρει και έχει τροποποιήσει τα επίπεδα ασφαλείας του, όπου στη συγκεκριμένη περίπτωση είναι το ενσωματωμένο υδατογράφημα, διατηρώντας ταυτόχρονα αναλλοίωτη την ορθότητα του κώδικα. Προκειμένου αυτό να έχει αρκετές πιθανότητες να συμβεί, πρέπει αυτός ο οποίος θα διαπράξει αυτήν την επίθεση προηγουμένως, να έχει αναλύσει αποτελεσματικά και εις βάθος τον κώδικα του λογισμικού. Με τον τρόπο αυτόν, θα μπορέσει να είναι πιο σίγουρος πως οι ενέργειες που πράττει πάνω στον κώδικα σε πρώτο επίπεδο δεν θα τον αλλοιώσουν και σε ένα δεύτερο επίπεδο θα είναι πιο στοχευμένες από ότι θα ήταν αν δεν γίνονταν κανένα είδους ανάλυσης.

Από την σκοπιά της ανάλυσης του κώδικα λοιπόν υπάρχουν τεχνικές οι οποίες η κάθε μία από αυτές, στο επίπεδο που της αναλογεί, εξετάζει την συμπεριφορά του κώδικα. Οι αναλύσεις αυτές είναι οι εξής:

- **Στατική ανάλυση** (Static analysis): Εφαρμόζοντας κάποιος αυτήν την ανάλυση στόχο έχει να εξάγει κάποιο αξιόλογο συμπέρασμα από τα αποτελέσματα που δίνει κάθε μία συνάρτηση του λογισμικού (να καταλάβει τι ακριβώς κάνει και ποια δεδομένα επιστρέφει ανάλογα με τις παραμέτρους που δέχεται ως ορίσματα), δίχως όμως να εκτελεστεί ο κώδικας του λογισμικού. Ένα απλό παράδειγμα είναι, αν μία μέθοδος έχει κίβδηλο κώδικα (dummy code) συγκριτικά με τον υπόλοιπο κώδικα του λογισμικού ή αν επιστρέφει συνεχώς τα ίδια αποτελέσματα, οπότε και στις δύο περιπτώσεις οι συναρτήσεις αυτές μπορούν εύκολα να αφαιρεθούν.

- **Δυναμική ανάλυση** (Dynamic analysis): Με την εφαρμογή αυτής της ανάλυσης πραγματοποιείται η εκτέλεση του κώδικα του λογισμικού με διάφορες ακολουθίες εισόδου παρατηρώντας τον τρόπο με τον οποίο εξελίσσεται η ροή εκτέλεσης και μέσα από αυτό τα δεδομένα που δημιουργούνται σε κάθε χρονική στιγμή. Αυτού του είδους η ανάλυση είναι προφανώς ισχυρότερη και αποτελεσματικότερη από την προηγούμενη, αλλά παράλληλα πιο απαιτητική και χρονοβόρα λόγω του γεγονότος ότι η δημιουργία ενός συνόλου κάλυψης όλων των πιθανών ακολουθιών εισόδου για οποιοδήποτε πραγματικό κώδικα λογισμικού αποτελεί ακόμα και σήμερα ένα δύσκολο πρόβλημα. Επομένως σε λογισμικά μεγάλης έκστασης κώδικα, ένας εισβολέας ο οποίος διαπράττει την ανάλυση αυτή, δεν μπορεί ποτέ να είναι απόλυτα σίγουρος πως έχει εξετάσει πλήρως τον κώδικα δίνοντας μονάχα ακολουθίες εισόδου.

- **Διερμηνευτική ανάλυση** (Interpretative analysis): Εφαρμόζοντας αυτήν την ανάλυση, ουσιαστικά είναι σαν να εφαρμόζεται ενός συνδυασμός των προηγουμένων δύο. Αυτή η ανάλυση στόχο έχει να βρει ποιο είναι το αποτέλεσμα μίας συγκεκριμένης κλήσης συνάρτησης όπως γίνεται στην static analysis, με την διαφορά ότι αυτό γίνεται διερμηνεύοντας τον κώδικά της, σε μία άλλη εικονική μηχανή, γεγονός το οποίο μοιάζει με την dynamic analysis. Στη γενική των περιπτώσεων η διερμηνεία σε εικονική μηχανή είναι ακόμα πιο απαιτητική και χρονοβόρα λειτουργία από την δυναμική ανάλυση, αλλά παρέχει καλύτερες και πιο σαφείς πληροφορίες σχετικά με το πώς συμπεριφέρεται κάθε συνάρτηση ανάλογα με τα ορίσματα οποία δέχεται. Η προσπάθεια όμως που απαιτείται να καταβάλει κάποιος με σκοπό να κατασκευάσει μία τέτοια εικονική μηχανή και εν συνέχεια να δίνει ακολουθίες εισόδου και ορίσματα προκειμένου να διερμηνεύεται η συγκεκριμένη συνάρτηση απαιτεί υπομονή, φαντασία και πολύ χρόνο.

- **Στατιστική ανάλυση** (Statistical analysis): Η ανάλυση αυτή χρησιμοποιείται κατά κόρον από αυτόματα κακόβουλα εργαλεία τα οποία επιτελούν αποσυσκοτίσεις κώδικα. Συγκεκριμένα αυτή η ανάλυση επικεντρώνεται στις αποτιμήσεις των κατηγορημάτων του κώδικα, κατά την διάρκεια εκτέλεσής του, τα οποία εμπεριέχονται σε συνθήκες ελέγχου ή επανάληψης. Με αυτού του είδους την ανάλυση εντοπίζονται πρώτον κατηγορήματα τα οποία για ένα μεγάλο πλήθος ακολουθιών εισόδου αποτιμώνται πάντα ως αληθής ή ψευδής και δεύτερον κατηγορήματα τα οποία η αποτίμησή τους εξαρτάται από την ακολουθία εισόδου.

Η στατική ανάλυση χωρίζεται σε κατηγορίες, η κάθε μία από τις οποίες προσεγγίζει τον κώδικα του λογισμικού από μία διαφορετική οπτική γωνία και θέτει συγκεκριμένα ερωτήματα προς απάντηση. Αναφορικά οι κατηγορίες αυτές είναι η ανάλυση το ελέγχου ροής (control flow analysis), η ανάλυση ροής δεδομένων (data flow analysis), η ανάλυση εξάρτησης δεδομένων (data dependence analysis), η ανάλυση ψευδωνύμων (alias analysis), ο τεμαχισμός κώδικα (slicing) και η αφαιρετική διερμήνευση (abstract interpretation) [24]. Τα ερωτήματα που τίθενται για παράδειγμα από την ανάλυση ροής δεδομένων ενός προγράμματος είναι αν οι τιμές μίας μεταβλητής του, χρησιμοποιούνται ύστερα από ένα σημείο στον κώδικα ή αν οι τιμές κάποιων μεταβλητών παραμένουν καθ όλη την έκταση του κώδικα σταθερές. Όταν κάποιος αποφασίσει να εφαρμόσει στατική ανάλυση σε κώδικα λογισμικού, πρέπει προτού επιλέξει συγκεκριμένη κατηγορία αλγορίθμων που κάνουν αυτήν την ανάλυση, από τη μία πλευρά να καθορίσει την ακρίβεια των πληροφοριών που επιθυμεί να εξάγει και από την άλλη την προσπάθεια που χρειάζεται να καταβάλει προκειμένου να την εφαρμόσει (π.χ., πολυπλοκότητα αλγορίθμου στατικής ανάλυσης και απαιτούμενος χρόνος εκτέλεσης).

Η δυναμική ανάλυση όπως και η στατική χωρίζεται και αυτή σε κατηγορίες ή καλύτερα αυτήν την φορά σε τεχνικές οι οποίες χειρίζονται και αναλύουν τον κώδικα με διαφορετικό τρόπο η κάθε μία από αυτές. Επί της ουσίας οι τεχνικές αυτές εκτελούν τον κώδικα με σκοπό να αντλήσουν πληροφορίες εκτέλεσης από αυτόν. Οι τεχνικές αυτές είναι η αποσφαλμάτωση (debugging), η παρακολούθηση (profiling), η καταγραφή του ίχνους εκτέλεσης (tracing) και η εξομοίωση (emulation).

Ένα κακόβουλο αυτόματα εργαλείο εφαρμόζοντας στατιστική ανάλυση δεν μπορεί έτσι απλά, ελαφρά την καρδία, να αντικαταστήσει τα κατηγορήματα στον κώδικα τα οποία αποτιμώνται ύστερα από τις δοθείσες σε αυτό ακολουθίες εισόδου ως αληθή με $true$ και αυτά τα οποία αποτιμώνται ως ψευδή με $false$ καθώς είναι πολύ πιθανόν να δημιουργηθούν προβλήματα εκτέλεσης. Ένα απλό παράδειγμα και συνάμα πολύ συχνό είναι στην γλώσσα προγραμματισμού Java να υπάρχουν πολλές συνθήκες ελέγχου όπως οι try - catch οι οποίες να είναι υπεύθυνες για καταστάσεις οι οποίες συμβαίνουν σπάνια και κάτω από συγκεκριμένες συνθήκες (exeptional circumstances). Με απλά λόγια τέτοιου είδους κατηγορήματα συνήθως ή μάλλον σχεδόν πάντα αποτιμώνται ως ψευδή και μόνο αν συμβεί μία συγκεκριμένη συνθήκη αποτιμώνται ως αληθή. Επομένως αν ένα τέτοιο κατηγόρημα αντικατασταθεί με $false$, τότε ο κώδικας κινδυνεύει να κρεμάσει όταν συμβούν καταστάσεις οι οποίες χαρακτηρίζονται από τους προγραμματιστές ως επικίνδυνες ή επισφαλής.

Επιπλέον η στατιστική ανάλυση μπορεί να χρησιμοποιηθεί και για εμπεριστατωμένη αξιολόγηση κώδικα. Συγκεκριμένα, αν σε έναν κώδικα P ένα αδιαφανές κατηγόρημα της μορφής P^T αναγνωριστεί μέσα του (ύστερα βέβαια από ένα πλήθος ακολουθιών εισόδου) και αντικατασταθεί με $true$, τότε δημιουργείται μία νέα έκδοση του κώδικα P'. Η νέα αυτή έκδοση εκτελείται παράλληλα με την παλαιά για όλες εκείνες τις ακολουθίες εισόδου που είχαν δοθεί για να αναγνωριστεί τα κατηγόρημα του οποίου η αποτίμηση ήταν μονίμως $true$ και αν έχουν πανομοιότυπα αποτελέσματα τότε η αντικατάσταση αυτή ήταν επιτυχής και διατηρείται και το κατηγόρημα το οποίο και αφαιρέθηκε χαρακτηρίζεται ως εικονικό (bo-

gus). Όμως σε τέτοιου είδους καταστάσεις κατά τις οποίες αντικαθιστώνται εικονικά κατηγορήματα πρέπει να είναι σίγουρο πως έχουν καλυφθεί όλα τα πιθανά μονοπάτια εκτέλεσης με όλες τις διαφορετικές ακολουθίες εισόδου. Πράγμα το οποίο είναι εξαιρετικά δύσκολο, χρονοβόρο και σε περιπτώσεις μάλιστα όπου ο κώδικας είναι εκτενής και πολύπλοκος είναι ιδιαίτερα αμφίβολο αν μπορεί να επιτευχθεί μία τέτοιου είδους πράξη. Γι αυτό ακριβώς στις τεχνικές συσκότισης οι οποίες εφαρμόζουν μετασχηματισμούς με μεγάλη ανθεκτικότητα προτιμώνται τα κατηγορήματα τύπου $P^?$ τα οποία είναι και αληθή και ψευδή ανάλογα με την ακολουθία εισόδου για να μην μπορούν να αντικατασταθούν εύκολα με $true$ ή $false$ αλλά και ούτε και με κάποια απλούστερη μορφή.

2.7 Ιδανική Κλάση Γραφοθεωρητικών Υδατογραφημάτων

Τα γραφοθεωριτικά υδατογραφήματα αποτελούν ίσως την κλάση υδατογραφημάτων η οποία χαίρει ιδιαίτερης προτίμησης από την επιστημονική κοινότητα της υδατογράφησης λογισμικού από τις πρώτες κιόλας επιστημονικές εργασίες που δημοσιεύτηκαν [64]. Το γεγονός αυτό είναι απολύτως δικαιολογημένο καθώς ορισμένες κλάσεις γραφημάτων χαρακτηρίζονται από ιδιότητες οι οποίες τους βοηθούν να προσομοιώνουν σε πολύ καλό βαθμό την ροή εκτέλεσης ενός προγράμματος, με αποτέλεσμα η ενσωμάτωσή τους σε κώδικες να μην επιδρούν αρνητικά στην εκτέλεση ή τουλάχιστον να μην δίνουν εύκολα ερεθίσματα σε κακόβουλους χρήστες. Επί της ουσίας τα γραφήματα αυτά μοιάζουν αρκετά με τα διαγράμματα ελέγχου ροής.

Κατά καιρούς έχουν προταθεί διάφορες τεχνικές οι οποίες ενσωματώνουν γραφήματα στους κώδικες των λογισμικών τα οποία φέρουν είτε την στατική, είτε την δυναμική ιδιότητα. Συγκεκριμένα όλοι όσοι επιθυμούν να ενσωματώσουν μία δομή τύπου γραφήματος μέσα στον κώδικα ενός λογισμικού επικεντρώνονται σε γραφήματα με τις ακόλουθες ιδιότητες:

- Ικανότητα για την αποτελεσματική κωδικοποίηση και αποκωδικοποίηση ενός ακεραίου αριθμού σε ένα γράφημα μέσα από πολυωνυμικούς αλγορίθμους. Τα γραφήματα αυτά, είναι επιθυμητό να μιμούνται την ροή του κώδικα.

- Ύπαρξη ενός κόμβου ρίζα από τον οποίον όλοι οι κόμβοι του γραφήματος να είναι προσβάσιμοι από αυτόν.

- Υψηλό ρυθμό δεδομένων.

- Χαμηλό εξερχόμενο αριθμό ακμών ανά κόμβο προκειμένου να μοιάζουν με κοινές δομές όπως είναι οι λίστες και τα δέντρα καθώς επίσης και να βρίσκονται κοντά στην φιλοσοφία των συνθηκών ελέγχου, όπου από ένα σημείο τα πιθανά ενδεχόμενα είναι δύο, δηλαδή στη περίπτωση των γραφημάτων ένας κόμβος να έχει δύο εξερχόμενες ακμές.

- Δυνατότητα αυτοδιόρθωσης (error correcting) προκειμένου αν γίνουν τροποποιήσεις στη δομή τους, αυτή να μπορεί να γίνει όπως ήταν να λάβουν χώρα αυτές. Μία τέτοια δυνατότητα είναι η ύπαρξη hamiltonial μονοπατιού.

- Δυνατότητα θωράκισης (tamperproofing abilities) στοχεύοντας στον εντοπισμό πιθανών τροποποιήσεων. Μία τέτοια δυνατότητα θα μπορούσε να είναι ο συγκεκριμένος αριθμός εξερχόμενων ακμών ανά κόμβο του γραφήματος.

- Δυνατότητα υπολογισμού αλγορίθμου για ισομορφισμό γραφημάτων για χρήση κατά την διαδικασία αναγνώρισης τους.

Προκειμένου να ενσωματωθεί ένα γράφημα σε ένα λογισμικό, προηγουμένως πρέπει να τροποποιηθεί σε μορφή κατάλληλη προς ενσωμάτωση. Μία συνηθισμένη τέτοια μορφή είναι τα διαγράμματα ελέγχου ροής. Με μία τέτοια τροποποίηση όμως, δημιουργούνται δύο σοβαρά μειονεκτήματα τα οποία σιωπηλά δηλώνουν την στρέψη σε μία διαφορετική προσέγγιση. Το πρώτο μειονέκτημα είναι πως προκειμένου να μετατραπούν σε διαγράμματα ελέγχου ροής πρέπει για κάθε ένα από τα τμήματα (blocks) των διαγραμμάτων ροής να παράγεται και ο αντίστοιχος κώδικας ο οποίος δεν θα είναι κάτι άλλο πέραν από κίβδηλος (dummy). Το δεύτερο μειονέκτημα είναι πως ενσωματώνοντας τα παραγόμενα διαγράμματα ελέγχου ροής στον κώδικα προκειμένου αυτά μελλοντικά να αναγνωριστούν πρέπει να γίνει χρήση σημείων σήμανσης (marks).

Επομένως, λαμβάνοντας σοβαρά υπόψη αυτά τα δύο μειονεκτήματα καθώς επίσης και όλη τη θεωρία που διατυπώθηκε ως τώρα στην προκειμένη εργασία, στο επόμενο κεφάλαιο θα γίνει η εισαγωγή ενός νέου και πρωτοποριακού μοντέλου το οποίο εν μέρη ξεπερνά εμπόδια που ως τώρα έφερναν σύγχυση στον τομέα της υδατογράφησης λογισμικού.

2.8 Η Σημασία της Διαφάνειας

Η διαφάνεια (stealth) στην υδατογράφηση λογισμικού αποτελεί την αρχή και το τέλος της διαδικασίας. Είναι αυτή η οποία καθορίζει πόσα καλά έχει προστεθεί ο κώδικας που κωδικοποιεί το υδατογράφημα με τον υπόλοιπο κώδικα. Με την λέξη καλά εννοείται πως ο επιπλέον κώδικας του υδατογραφήματος έχει ενσωματωθεί με τέτοιον τρόπο έτσι ώστε να είναι και δύσκολο διακριθεί από τον αρχικό κώδικα, αλλά και να αφαιρεθεί χωρίς να δημιουργήσει προβλήματα λειτουργικότητας. Η έννοια αυτή δεν πρέπει να συγχέεται σε καμία των περιπτώσεων με την αορατότητα (invisibility). Η βασική τους διαφορά είναι πως το διαφανές υδατογράφημα βρίσκεται μπροστά στα μάτια κάποιου ο οποίος παρατηρεί τον κώδικα του λογισμικού περνώντας απαρατήρητο, ενώ το αόρατο υδατογράφημα είναι κρυμμένο μέσα στον κώδικα ενδεχομένως στο εσωτερικό δομών (π.χ., πίνακες, δέντρα, λίστες).

Ένα ωραίο παράδειγμα προκειμένου να γίνει αντιληπτή η έννοια της διαφάνειας είναι τα αεροπλάνα τύπου stealth, των οποίων η κατασκευή πραγματοποιήθηκε για πρώτη φορά στην Γερμανία κατά την διάρκεια του δευτέρου παγκοσμίου πολέμου, τα οποία πετούν

κανονικά όπως όλα τα αεροπλάνα στον ουρανό αλλά σε αντίθεση με τα υπόλοιπα δεν γίνονται αντιληπτά από ένα μεγάλο σύνολο ραντάρ. Τα αεροπλάνα αυτά, όπως και τα υδατογραφήματα τα οποία είναι αδιαφανή κάνουν χρήση ορισμένων τεχνικών οι οποίες από την πλευρά των αεροπλάνων μειώνουν σε πολύ μεγάλο βαθμό την αντανάκλαση / εκπομπή των ραντάρ ενώ από την πλευρά των υδατογραφημάτων μειώνουν σε πολύ μεγάλο βαθμό τις επικίνδυνες στατιστικές ιδιότητες του κώδικα (π.χ., σε κάθε συνάρτηση μέσα στο σύνολο των εντολών της εμφανίζεται δύο φορές ένα σύνολο εντολών) οι οποίες δημιουργούνται από τα συνήθη υδατογραφήματα.

Από τη στιγμή όπου ένα κακόβουλος χρήστης εντοπίσει ένα υδατογράφημα μέσα στον κώδικα ενός λογισμικού, είναι δεδομένο ότι μπορεί και να το αφαιρέσει. Αν έχει την ιδιότητα του εύρωστου υδατογραφήματος ίσως του πάρει κάποιον περισσότερο χρόνο, από ότι υα του έπαιρνε αν είχε την ιδιότητα του εύθραυστου, αλλά όπως και να έχει είναι κοινά αποδεκτό ότι είναι σε θέσει να το αλλοιώσει. Όπως έχει πει και ο C. Collberg (επιστήμονας στον τομέα της υδατογράφησης λογισμικού με πολλές δημοσιευμένες εργασίες σε επιστημονικά συνέδρια και περιοδικά στο ενεργητικό του) αν κάποιος κακόβουλος χρήστης έχει εντοπίσει την τοποθεσία του υδατογραφήματος δεν υπάρχουν περιθώρια αποφυγής [24].

Υπάρχει η στατική διαφάνεια και η δυναμική διαφάνεια [62]. Η στατική διαφάνεια αναφέρεται στις στατικές ιδιότητες ενός προγράμματος (π.χ., συχνότητες εμφάνισης εντολών μέσα στον κώδικα) ενώ η δυναμική διαφάνεια αναφέρεται στις δυναμικές ιδιότητες (π.χ., συχνότητες εμφάνισης εντολών μέσα στον ίχνος εκτέλεσης). Επιπλέον η διαφάνεια μπορεί να είναι είτε τοπική (local), είτε καθολική (global). Η τοπική διαφάνεια αναφέρεται στη σύγκριση της υδατογραφημένης έκδοσης ενός κώδικα και στη μη υδατογραφημένη, ενώ η καθολική διαφάνεια αναφέρεται στη σύγκριση μεταξύ ενός υδατογραφημένου κώδικα με πολλούς και ποικίλους κώδικες.

Κατά καιρούς έχουν δημοσιευτεί εργασίες που παρουσιάζουν αλγοριθμικές τεχνικές υδατογράφησης λογισμικού οι οποίες ενσωματώνουν υδατογραφήματα τα οποία δίνουν έναυσμα προς περαιτέρω ψάξιμο σε εισβολείς. Το γεγονός αυτό οφείλεται στην χαμηλή διαφάνεια που τις χαρακτηρίζει καθώς σύμφωνα με τους αλγορίθμους που ακολουθούν κωδικοποιούν το υδατογράφημα δημιουργώντας τρωτά σημεία και καταστάσεις μέσα στον κώδικα τα οποία είναι ιδιαίτερα εμφανή και ευκόλως εκμεταλλεύσιμα. Τέτοια σημεία είναι τα εξής:

- Σημεία στα οποία βρίσκονται ακολουθίες από συνθήκες ελέγχου οι οποίες αποτιμούν εκφράσεις που περιέχουν μεγάλο αριθμό από ασυνήθιστες σταθερές, μη συσχετισμένες με τον υπόλοιπο κώδικα.

- Ασυνήθιστα μεγάλο ποσοστό από εντολές τύπου goto συγκριτικά με το μέγεθος σε εντολές του κώδικα.

- Συναρτήσεις οι οποίες εμπεριέχουν κίβδηλο (dummy) κώδικα, ο οποίος είναι εξ ορισμού άσχετος με τον υπόλοιπο κώδικα που τον περιέχει.

- Περιοδικές εμφανίσεις συγκεκριμένων συνόλων εντολών (π.χ., εμφάνιση της εντολής

push k φορές συνεχόμενα και αμέσως μετά της εντολής pop με τον ίδιο αριθμό εμφανίσεων) οι οποίες δεν παρέχουν κάποια ουσιαστική λειτουργικότητα.

Όλο το θέμα στη διαφάνεια ενός ενσωματωμένου υδατογραφήματος είναι να φαίνεται φυσιολογικός και άρρηκτα αλληλένδετος με τον υπόλοιπο κώδικα, ο κώδικας που το κωδικοποιεί. Ουσιαστικά να μη δίνει κίνητρο σε κάποιον που πιθανόν να διαβάζει τον κώδικα. Το γεγονός αυτό βέβαια, προϋποθέτει έναν συνδυασμό άπλετης φαντασίας και εμπειρίας από την πλευρά του προγραμματιστή.

Κεφαλαιο 3

Το Μοντελο Υδατογραφησης Λογισμικου WaterRPG

3.1 Προγενέστερα Αποτελέσματα

3.2 Βασική Ιδέα του WaterRPG Μοντέλου

3.3 Διαδικασίες Ενσωμάτωσης και Εξαγωγής του Υδατογραφήματος

3.4 Συνιστώσες του Μοντέλου WaterRPG

3.5 Αλγόριθμος Υδατογράφησης και Απόδειξη Ορθότητας

3.1 Προγενέστερα Αποτελέσματα

Εχοντας παρουσιάσει και αναλύσει εις βάθος ένα μεγάλο τμήμα περί προστασίας κώδικα λογισμικού μέσω τεχνικών όπως η υδατογράφηση, η συσκότιση και η θωράκιση η παρούσα διατριβή στο σημείο αυτό βασίζεται σε ήδη δημοσιευμένα αποτελέσματα επιστημονικών εργασιών με σκοπό να προτείνει και εν συνεχεία να αξιολογήσει ένα νέο δυναμικό μοντέλο υδατογράφησης λογισμικού. Τα αποτελέσματα στα οποία αυτή βασίζεται αφορούν και την φάση της δημιουργίας υδατογραφήματος [13] [14] [15] και την φάση της ενσωμάτωσης υδατογραφήματος [16] έχοντας έτσι ένα πλήρες υπόβαθρο.

3.1.1 Αλγόριθμος Chroni & Nikolopoulos

Ο αλγόριθμος Chroni & Nikolopoulos μετατρέπει, απαιτώντας γραμμικό χρόνο και χώρο, έναν ακέραιο αριθμό τον οποίο τον δέχεται ως είσοδο σε ένα αναγώγιμο μεταθετικό γράφημα (reducible permutation graph - RPG) το οποίο το παράγει ως έξοδο μέσω μίας αυτοαναστρέφουσας μετάθεσης (self inverting permutation - SiP) και το αντίστροφο [13].

Επί της ουσίας ο αλγόριθμος αυτός χωρίζεται σε τέσσερις επιμέρους φάσεις, εκ των οποίων οι πρώτες δύο αποτελούν το ορθό της διαδικασίας κατασκευής και οι τελευταίες δύο αποτελούν το αντίστροφο της διαδικασίας κατασκευής, όπου γίνεται η μεταφορά από το αναγώγιμο μεταθετικό γράφημα πίσω στον ακέραιο αριθμό. Συγκεκριμένα, η πρώτη φάση κωδικοποιεί έναν φυσικό αριθμό μέσω μίας διτονικής (bitonic) μετάθεσης σε μία αυτοαναστρέφουσα μετάθεση, η δεύτερη φάση κωδικοποιεί την αυταναστρέφουσα μετάθεση μέσω ενός κατευθυνόμενου άκυκλου γραφήματος (directed acyclic graph - DAG) σε ένα αναγώγιμο μεταθετικό γράφημα, η τρίτη φάση αποκωδικοποιεί το αναγώγιμο μεταθετικό γράφημα μέσω ενός δέντρου στην αυατοαναστρέφουσα μετάθεση και η τέταρτη και τελευταία φάση αποκωδικοποιεί την παραχθείσα μετάθεση πίσω στον αρχικό ακέραιο αριθμό.

Παρακάτω παρατίθενται οι αλγόριθμοι κωδικοποίησης και αποκωδικοποίησης των τεσσάρων αυτών φάσεων καθώς επίσης και ένα παράδειγμα στον κάθε έναν από αυτούς προκειμένου να γίνει σαφέστερη και συνάμα πιο κατανοητή η διαδικασία.

Είσοδος: ακέραιος αριθμός w
Εξοδος: μετάθεση SiP

---

1. Υπολογισμός δυαδικής αναπαράστασης του w, έστω $B = b_1, b_2, \ldots, b_n$;
2. Κατασκευή του δυαδικού αριθμού έστω $B' = 00\ldots0||B||1$ μήκους $n' = 2 * n + 1$;
3. Κατασκευή του δυαδικού αριθμού έστω $B^* = \bar{B'}$;
4. Κατασκευή των ακολουθιών:

  4.1 $X = x_1, x_2, \ldots, x_k$ με τις θέσεις των 1 του B^*;
  4.2 $Y = y_1, y_2, \ldots, y_m$ με τις θέσεις των 0 του B^*;

5. Κατασκευή της διτονικής μετάθεσης $\pi^b = X||Y^R$;
6. Εστω $\pi_1, \pi_2, \ldots, \pi_x, \pi_{x+1}, \ldots, \pi_{n'} = x_1, x_2, \ldots, x_k, y_m, y_{m-1}, \ldots, y_1$, $i = 1$, $j = n'$;
Οσο $i < j$ **επανέλαβε**

  6.1 Κατασκευή του κύκλου $c_i = (\pi_i, \pi_j)$;
  6.2 $i = i + 1$;
  6.3 $j = j - 1$;

Τέλος επανάληψης
Αν $i = j$ **τότε** κατασκευή του κύκλου $c_i = (\pi_i, \pi_i)$;
7. Εστω $SiP = \pi_1, \pi_2, \ldots, \pi_{n'}$, έτσι ώστε $\pi_i = i$, $1 \leq i \leq n'$;
8. **Για κάθε** κύκλο $c_i = (\pi_i, \pi_j) \in C$

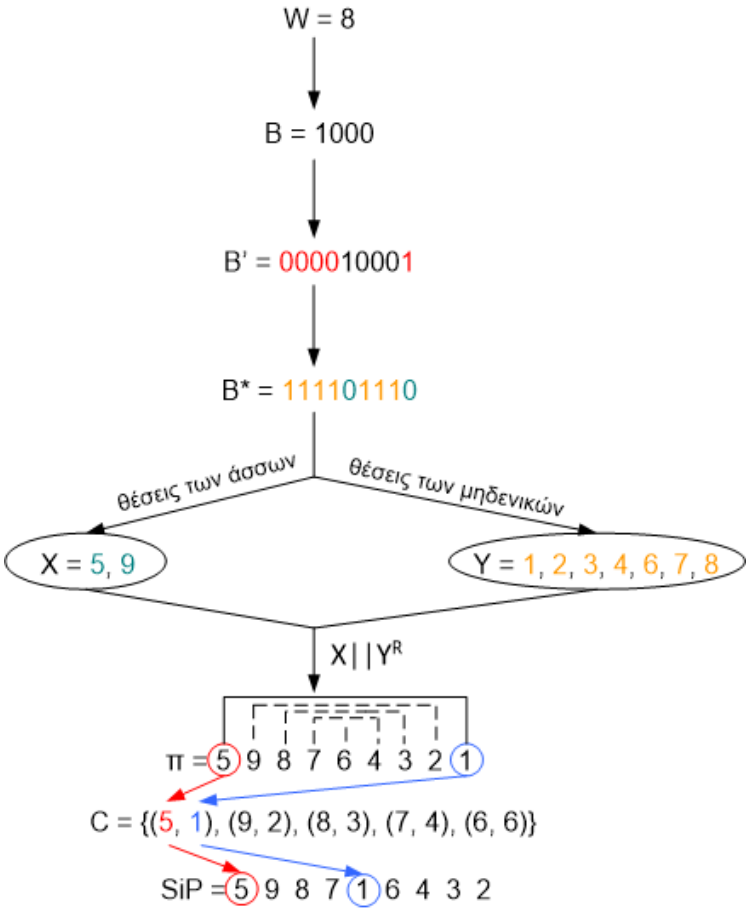
  6.1 Το π_i εισέρχεται στη θέση που δηλώνει το π_j στην SiP και το αντίστροφο;

Τέλος επανάληψης
9. **Επιστροφή** της SiP;

Αλγόριθμος 1: Πρώτη φάση κωδικοποίησης *Encode_W_to_SiP*

Ο αλγόριθμος αυτός της πρώτης φάσης ο οποίος κατασκευάζει από έναν ακέραιο αριθμό μία αυτοαναστρέφουσα μετάθεση μπορεί να χρησιμοποιηθεί και αυτούσιος για την παραγωγή δομών υδατογράφησης (μαζί βέβαια με τον αλγόριθμο της τέταρτης κατά σειράς φάσης για την αποκωδικοποίησή τους) οι οποίες μπορούν να ενσωματωθούν αποτελεσματικά και σε άλλα ψηφιακά αντικείμενα. Τέτοια αντικείμενα μπορεί να είναι, είτε εικόνες [9] [18] [19] [20] είτε ηχητικά σήματα [9].

Στο παρακάτω παράδειγμα ο ακέραιος αριθμός 8 μετατρέπεται, μέσω του αλγορίθμου της πρώτης φάσης, σε μία αυτοαναστρέφουσα μετάθεση. Τα κόκκινα και μπλε βέλη δείχνουν πως από τη διτονική μετάθεση κατασκευάζεται ένα ζεύγος στοιχείων καθώς επίσης και πως αυτό το ζεύγος στοιχείων τοποθετείται στις θέσεις της αυτοαναστρέφουσας μετάθεσης.



Σχήμα 3.1: Παράδειγμα πρώτης φάσης κωδικοποίησης.

Στο σημείο αυτό, προκειμένου να αποφευχθεί η δημιουργία κάποιου κενού σχετικά με τον αλγόριθμο της δεύτερης φάσης είναι σκόπιμο να οριστεί ο τρόπος με τον οποίον υπολογίζονται τα κυρίαρχα στοιχεία μίας μετάθεσης. Συγκεκριμένα σε μία μετάθεση έστω $\pi = \pi_1, \pi_2, \ldots, \pi_n$ ένα στοιχείο π_j κυριαρχείται (dominated) από ένα στοιχείο π_i αν το π_i (η τιμή του στοιχείου) είναι μεγαλύτερη από το π_j (η τιμή του του στοιχείου) και το

π_i^{-1} (η θέση του π_i στοιχείου στη μετάθεση) είναι μικρότερη από το π_j^{-1} (η θέση του π_j στοιχείου στη μετάθεση). Επίσης ένα στοιχείο π_j κυριαρχείται άμεσα (directly) από ένα στοιχείο π_i αν το π_i κυριαρχεί του π_j και δεν παρεμβάλλεται μεταξύ τους άλλο στοιχείο π_k στη μετάθεση τέτοιο ώστε το στοιχείο π_i να κυριαρχεί του στοιχείου π_k και το στοιχείο π_k να κυριαρχεί του στοιχείου π_j.

Έχοντας ορίσει λοιπόν τις δύο σχέσεις κυριαρχίας, την απλή και την άμεση, ορίζεται και το σύνολο κυριαρχίας $dom(i)$ ενός στοιχείου i. Το σύνολο αυτό περιέχει όλα εκείνα τα στοιχεία της μετάθεσης π τα οποία κυριαρχούνται από το στοιχείο i. Με τον ίδιο ακριβώς τρόπο ορίζεται και το σύνολο της άμεσης κυριαρχίας $didom(i)$, το οποίο περιέχει όλα εκείνα τα στοιχεία της μετάθεσης π τα οποία κυριαρχούνται άμεσα από το στοιχείο i.

Παρακάτω παρατίθεται ο αλγόριθμος της δεύτερης φάσης ο οποίος και ολοκληρώνει το ορθό της όλης διαδικασίας κατασκευής του αναγώγιμου μεταθετικού γραφήματος το οποίο θα αποτελέσει στη συνέχεια το υδατογράφημα.

Είσοδος: μετάθεση SiP
Εξοδος: γράφημα $F[SiP]$

---

1. Κατασκευή άκυκλου κατευθυνόμενου γραφήματος $D[SiP]$ με n κόμβους ως εξής:

- **1.1** Εστω $V(D[SiP]) = v_1, v_2, \ldots, v_n$;
- **1.2** Υπολογισμός συνόλου $didom(i)$ για κάθε στοιχείο i της SiP, $1 \leq i \leq n-1$;
- **1.3** $\forall\ j \in didom(i)$ προστίθεται η ακμή (v_i, v_j) στο σύνολο $E(D[SiP])$
- **1.4** Προσθήκη 2 κόμβων $s = v_{n+1}$ και $t = v_0$ στο σύνολο $V(D[SiP])$;
- **1.5** Προσθήκη ακμών (s, v_i) στο $E(D[SiP])$ $\forall$ κόμβο v_i με 0 εισερχόμενες ακμές;
- **1.6** Προσθήκη ακμών (v_i, t) στο $E(D[SiP])$ $\forall$ κόμβο v_i με 0 εξερχόμενες ακμές;

2. **Για κάθε** κόμβο $v_i \in V(D[SiP])$, $1 \leq i \leq n$ **επανέλαβε**

- **2.1** Υπολογισμός του συνόλου $P(v_i) = \{v_j \in V(D[SiP]) | (v_i, v_j) \in E(D[SiP])\}$;
- **2.2** Επιλογή κόμβου v_m με τη μεγαλύτερη ετικέτα $P(v_i)$;
- **2.3** Εστω $p(v_i) = v_m$;

3. Κατασκευή κατευθυνόμενου γραφήματος $F[SiP]$ ως εξής:

- **3.1** $V(F[SiP]) = \{t = v_0, v_1, \ldots, v_n, v_{n+1} = s\}$;
- **3.2** **Για** $i = n$ **μέχρι** 0 **επανέλαβε**
  - **3.2.1** Προσθήκη ακμής (v_{i+1}, v_i) στο $E(F[SiP])$;

  Τέλος επανάληψης

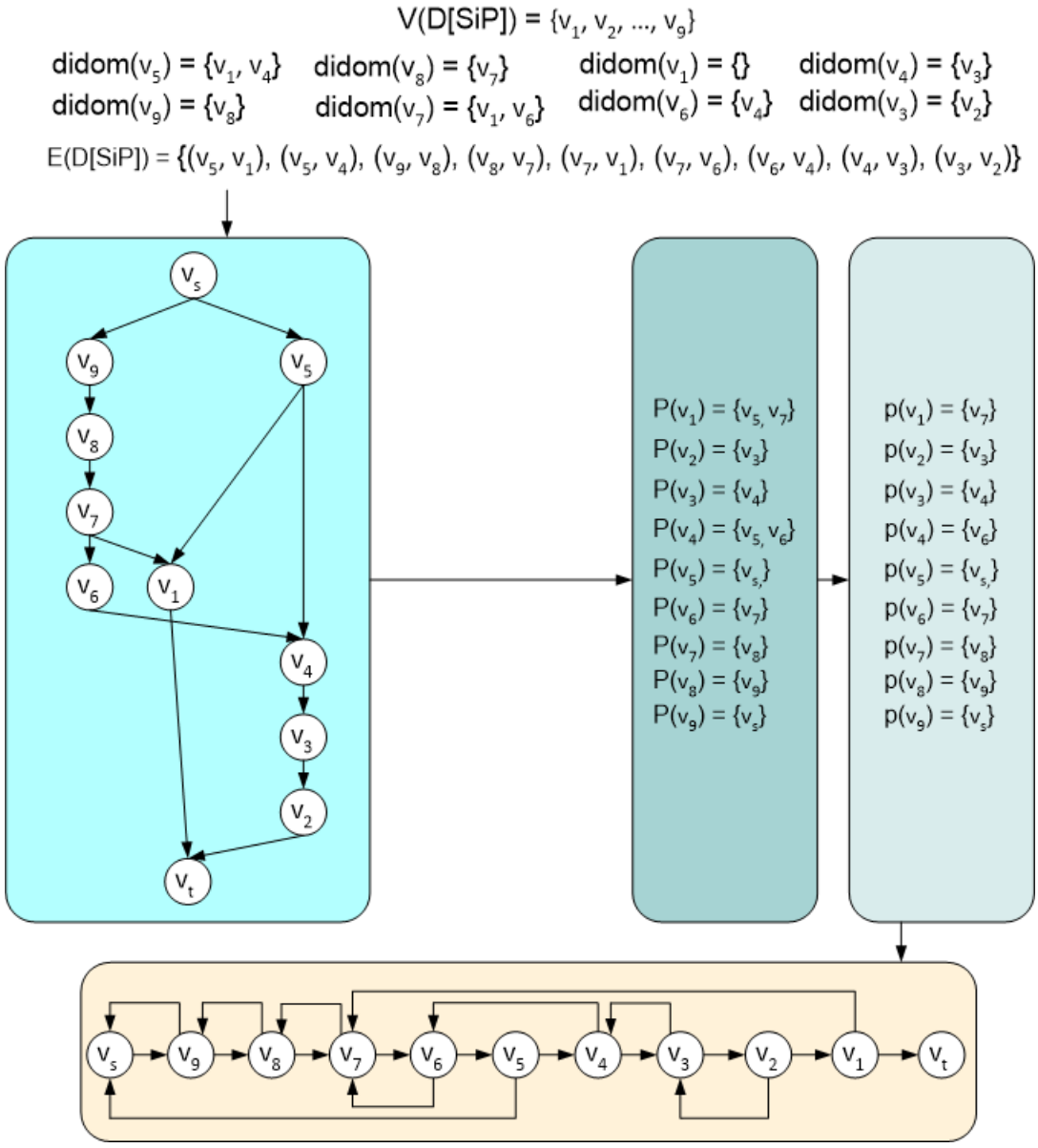
4. **Για κάθε** κόμβο $v_i \in V(F[SiP])$, $1 \leq i \leq n$ **επανέλαβε**

- **4.1** Προσθήκη ακμής (v_i, v_m) στο $E(F[SiP])$ αν $v_m = p(v_i)$;

5. **Επιστροφή** του $F[SiP]$;

Αλγόριθμος 2: Δεύτερη φάση κωδικοποίησης $Encode\_SiP\_to\_RPG$

Με τον τρόπο αυτόν ολοκληρώνεται το ορθό της διαδικασίας κατασκευής αναγώγιμου μεταθετικού γραφήματος από ακέραιο αριθμό. Παρακάτω ακολουθεί παράδειγμα στο οποίο η αυτοαναστρέφουσα μετάθεση η οποία έχει προέλθει από τον ακέραιο αριθμό 8, του προηγούμενου παραδείγματος προκειμένου να υπάρχει μία συνέχεια και συνοχή στη ροή της εν λόγω διατριβής, μετατρέπεται σε αναγώγιμο μεταθετικό γράφημα. Το γράφημα αυτό αποτελεί το υδατογράφημα το οποίο μετέπειτα επρόκειτο να ενσωματωθεί σε κώδικα λογισμικού βάση του μοντέλου WaterRpg. Προφανώς, το γράφημα αυτό προτού ενσωματωθεί σε λογισμικό μετατρέπεται σε κώδικα για να να μπορέσει να είναι λειτουργικό τόσο το υδατογράφημα όσο και ο κώδικας του λογισμικού.



Σχήμα 3.2: Παράδειγμα δεύτερης φάσης κωδικοποίησης.

Έχοντας παρουσιάσει το ορθό της διαδικασίας κωδικοποίησης ενός ακεραίου αριθμού σε αναγώγιμο μεταθετικό γράφημα, στο σημείο αυτό ακολουθεί η αντίστροφη διαδικασία κατά την οποία το αναγώγιμο μεταθετικό γράφημα, το οποίο κατασκευάστηκε από τις προηγούμενες δύο φάσεις, αποκωδικοποιείται στον ακέραιο αριθμό από τον οποίο και προήλθε. Παρακάτω παρουσιάζονται οι αλγόριθμοι των φάσεων αυτών με τα αντίστοιχα παραδείγματά τους.

Είσοδος: γράφημα $F[SiP]$
Εξοδος: μετάθεση SiP

---

1. Διαγραφή των ακμών (v_{i+1}, v_i) από το σύνολο $E(F[SiP])$, $1 \leq i \leq n$;
2. Διαγραφή του κόμβου $t = v_0$ από το σύνολο $V(F[SiP])$;
3. Αντιστροφή κατεύθυνσης στις ακμές του συνόλου $E(F[SiP])$;
4. Εφαρμογή αναζήτησης κατά βάθος επισκέπτοντας το παιδί με την μικρότερη ετικέτα;
5. Ταξινόμηση των κόμβων σε μετάθεση σύμφωνα με τη σειρά με την οποία επισκέφτηκαν;
6. Διαγραφή του κόμβου s από τη μετάθεση του προηγούμενου βήματος;
7. **Επιστροφή** της SiP;

Αλγόριθμος 3: Τρίτη φάση αποκωδικοποίησης *Decode_RPG_to_SiP*

Στο σημείο αυτό για λόγους καλύτερης κατανόησης του προηγούμενου αλγορίθμου καθώς επίσης και για λόγους πληρότητας γίνεται μία μικρή νύξη σε αλγορίθμους διάσχισης σε γραφήματα δίνοντας παράλληλα και τους αλγορίθμους σε μορφή ψευδοκώδικα τόσο της αναζήτησης κατά βάθος (depth first search - DFS) ή αλλιώς καθοδικής αναζήτησης όσο και της αναζήτησης κατά πλάτος (breadth first search - BFS) ή αλλιώς οριζόντιας αναζήτησης. Οι εν λόγω αλγόριθμοι αποτελούν ίσως τους πιο δημοφιλείς αλγορίθμους που εφαρμόζονται επάνω σε γραφήματα.

Όπως υποδηλώνει και η ονομασία της μεθόδου, στην αναζήτηση κατά βάθος, η διερεύνηση επεκτείνεται προς μεγαλύτερα βάθη στο γράφημα, οποτεδήποτε αυτό είναι εφικτό. Οι ακμές εξερευνούνται με αφετηρία τον πιο πρόσφατο κόμβο v από τον οποίο εκκινούν μη εξερευνημένες ακμές. Αφού εξερευνηθούν όλες οι ακμές του v, η διερεύνηση επιστρέφει στον κόμβο από τον οποίον εντοπίστηκε ο v και συνεχίζεται με τις τυχόν άλλες ακμές που εκκινούν από αυτόν. Η διαδικασία αυτή συνεχίζεται μέχρις ότου εντοπιστούν όλοι οι κόμβοι που είναι προσπελάσιμοι από τον αρχικό αφετηριακό κόμβο. Εάν εξακολουθούν να υπάρχουν μη εντοπισμένοι κόμβοι, τότε επιλέγεται ένας από αυτούς ως αφετηριακός κόμβος και η διερεύνηση επαναλαμβάνεται από αυτόν. Η όλη διαδικασία συνεχίζεται μέχρις ότου εντοπιστούν όλοι οι κόμβοι.

Κάθε φορά όπου εντοπίζεται ένας κόμβος v καθώς διατρέχεται ο κατάλογος γειτνίασης ενός ενός ήδη εντοπισμένου κόμβου u, ο αλγόριθμος καταγράφει αυτό το συμβάν θέτοντας

ως προκάτοχο του του v, $\pi[v]$, τον u. Σημειώνεται στο σημείο αυτό πως το υπογράφημα προκατόχων είναι πιθανόν να αποτελείται από διάφορα δέντρα, καθώς η διερεύνηση μπορεί να επαναληφθεί από πολλούς διαφορετικούς αφετηριακούς κόμβους. Το υπογράφημα προκατόχων ορίζεται ως εξής:

- $G_\pi = (V, E_\pi)$
- $E_\pi = \{(\pi[v], v) : v \in V και π[v] \neq \emptyset\}$

Το υπογράφημα προκατόχων αυτής της διερεύνησης σχηματίζει ένα καθοδικό δάσος το οποίο αποτελείται από διάφορα καθοδικά δέντρα. Οι ακμές του συνόλου E_π ονομάζονται δενδρικές. Κατά την πορεία της διερεύνησης αποδίδονται στους κόμβους χρώματα τα οποία υποδεικνύουν την κατάστασή τους. Κάθε κόμβος είναι αρχικά λευκός, στη συνέχεια, όταν εντοπίζεται, χρωματίζεται γκρίζος και όταν περατώνεται, όταν εν ολίγοις ολοκληρώνεται η εξέταση του καταλόγου γειτνίασής του, χρωματίζεται μελανός. Επομένως, με αυτόν τον τρόπο έχει εξασφαλιστεί ότι κάθε ένας από του κόμβους του γραφήματος θα ενταχθεί μονάχα σε ένα καθοδικό δέντρο.

Πέραν όμως του καθοδικού δάσους η διερεύνηση αυτή θέτει επίσης σε κάθε έναν κόμβο και χρονοασφραγίδες. Κάθε κόμβος v έχει δύο χρονασφραγίδες όπου η πρώτη, $d[v]$, καταγράφει τη χρονική στιγμή που εντοπίζεται για πρώτη φορά ο v, ενώ η δεύτερη $f[v]$ τη χρονική στιγμή που ολοκληρώνεται η εξέταση του καταλόγου γειτνίασης του v. Οι χρονοσφραγίδες αυτές χρησιμοποιούνται σε πληθώρα αλγορίθμων γραφημάτων με σκοπό να διευκολύνουν τις μελέτες συμπεριφορών και στη συγκεκριμένη περίπτωση, τη μελέτη της διερεύνησης κατά βάθος. Οι τιμές των χρονοσφραγίδων είναι ακέραιοι αριθμοί μεταξύ του 1 και του $2|V|$ καθώς για κάθε έναν από τους $|V|$ κόμβους του γραφήματος υπάρχει ένα συμβάν εντοπισμού και ένα συμβάν περάτωσης. Παρακάτω παρουσιάζεται υπό την μορφή ψευδοκώδικα η αναζήτηση κατά βάθος, χωρισμένη σε δύο συναρτήσεις.

ΚΑΘΟΔΙΚΗ ΔΙΕΡΕΥΝΗΣΗ(G)

---

1. για κάθε κόμβο $u \in V|G|$
2. χρώμα$[u] \longleftarrow$ ΛΕΥΚΟ
3. π$[u] \longleftarrow$ ΚΕΝΟ
4. χρόνος $\longleftarrow 0$
5. για κάθε κόμβο $u \in V|G|$
6. αν χρώμα$[u] =$ ΛΕΥΚΟ τότε
7. επίσκεψη ΚΑΘΟΔΙΚΗΣ ΔΙΕΡΕΥΝΗΣΗΣ(u)

ΚΑΘΟΔΙΚΗΣ ΔΙΕΡΕΥΝΗΣΗΣ(u)

1. χρώμα[u] $\longleftarrow$ ΓΚΡΙΖΟ
2. χρόνος $\longleftarrow$ χρόνος + 1
3. $d[u] \longleftarrow$ χρόνος
4. για κάθε $v \in Adj[u]$
5. αν χρώμα[v] = ΛΕΥΚΟ τότε
6. $\pi[v] \longleftarrow u$
7. επίσκεψη ΚΑΘΟΔΙΚΗΣ ΔΙΕΡΕΥΝΗΣΗΣ(v)
8. χρώμα[u] $\longleftarrow$ ΜΕΛΑΝΟ
9. f[u] $\longleftarrow$ χρόνος $\longleftarrow$ χρόνος + 1

Από την άλλη πλευρά η αναζήτηση κατά πλάτος είναι ένας από τους απλούστερους αλγορίθμους διερεύνησης ενός γραφήματος και αποτελεί το αρχέτυπο για πολλούς και συνάμα σημαντικούς αλγορίθμους γραφημάτων όπως είναι οι Prim (αλγόριθμος ελαφρύτερου συνδετικού δέντρου) και Dijkstra (αλγόριθμος ομοαφετηριακών ελαφρύτερων διαδρομών). Συγκεκριμένα, για ένα δεδομένο γράφημα $G = (V, E)$ και έναν δεδομένο αφετηριακό κόμβο s, η οριζόντια διερεύνηση συνίσταται στη συστηματική εξέταση των ακμών του G ώστε να εντοπιστούν όλοι οι κόμβοι οι οποίοι είναι προσπελάσιμοι από τον κόμβο s.

Στο πλαίσιο της εν λόγω μεθόδου υπολογίζεται η απόσταση (το μικρότερο πλήθος ακμών) ανάμεσα στον κόμβο s και σε κάθε προσπελάσιμο κόμβο. Δημιουργείται επίσης ένα οριζόντιο δέντρο με κόμβο ρίζα τον κόμβο s το οποίο και περιέχει όλους τους προσπελάσιμους κόμβους. Για οποιονδήποτε κόμβο v προσπελάσιμο από τον s η διαδρομή από τον s μέχρι τον v στο οριζόντιο δέντρο αντιστοιχεί σε μία βραχύτατη διαδρομή από τον s μέχρι τον v στο γράφημα G, δηλαδή σε μία διαδρομή που περιλαμβάνει το μικρότερο πλήθος ακμών. Η ονομασία του διάσχισης αυτής οφείλεται στο γεγονός ότι επεκτείνει το σύνορο μεταξύ εντοπισμένων και μη εντοπισμένων κόμβων ομοιόμορφα σε όλο το εύρος του συνόρου αυτού και στο γεγονός ότι το σύνορο έχει επικρατήσει να σχεδιάζεται κατά την οριζόντια διεύθυνση. Επί της ουσίας, εντοπίζονται όλοι οι κόμβοι σε απόσταση k από τον κόμβο s (αρχικός κόμβος) και μόνο αν εξαντληθούν αυτοί η αναζήτηση προχωρά στο επόμενο επίπεδο $k + 1$.

Όπως στην αναζήτηση κατά βάθος, έτσι και στην αναζήτηση κατά πλάτος οι κόμβοι παίρνουν τα χρώματα λευκό, γκρίζο και μελανό. Εν αρχή όλοι οι κόμβοι έχουν χρώμα λευκό και στη συνέχεια παίρνουν τα υπόλοιπα δύο χρώματα όπως γίνεται και στην αναζήτηση κατά βάθος. Η διερεύνηση αυτή κατασκευάζει σε αντίθεση με την προηγούμενη οριζόντιο δέντρο, το οποίο αρχικά περιέχει μόνο τον αφετηριακό κόμβο s. Οποτεδήποτε εντοπίζεται ένας κόμβος v κατά την διάρκεια σάρωσης του καταλόγου γειτνίασης ενός ήδη εντοπισμένου κόμβου u, ο κόμβος u και η ακμή (u, v) προστίθενται στο δέντρο. Επίσης σημειώνεται πως από τη στιγμή που ένας κόμβος εντοπίζεται ακριβώς μία φορά έχει το πολύ έναν προκάτοχο. Οι σχέσεις προγόνου και απογόνου στο οριζόντιο δέντρο ορίζονται ως προς τον ριζικό

κόμβο s με τον κλασσικό τρόπο: εάν ο u βρίσκεται σε μία διαδρομή του δέντρου από τον ριζικό κόμβο s μέχρι τον v, τότε ο κόμβος u είναι πρόγονος του v και ο v με τη σειρά του είναι απόγονος του u. Στη συνέχεια παρουσιάζεται ο ψευδοκώδικας της αναζήτησης κατά πλάτος.

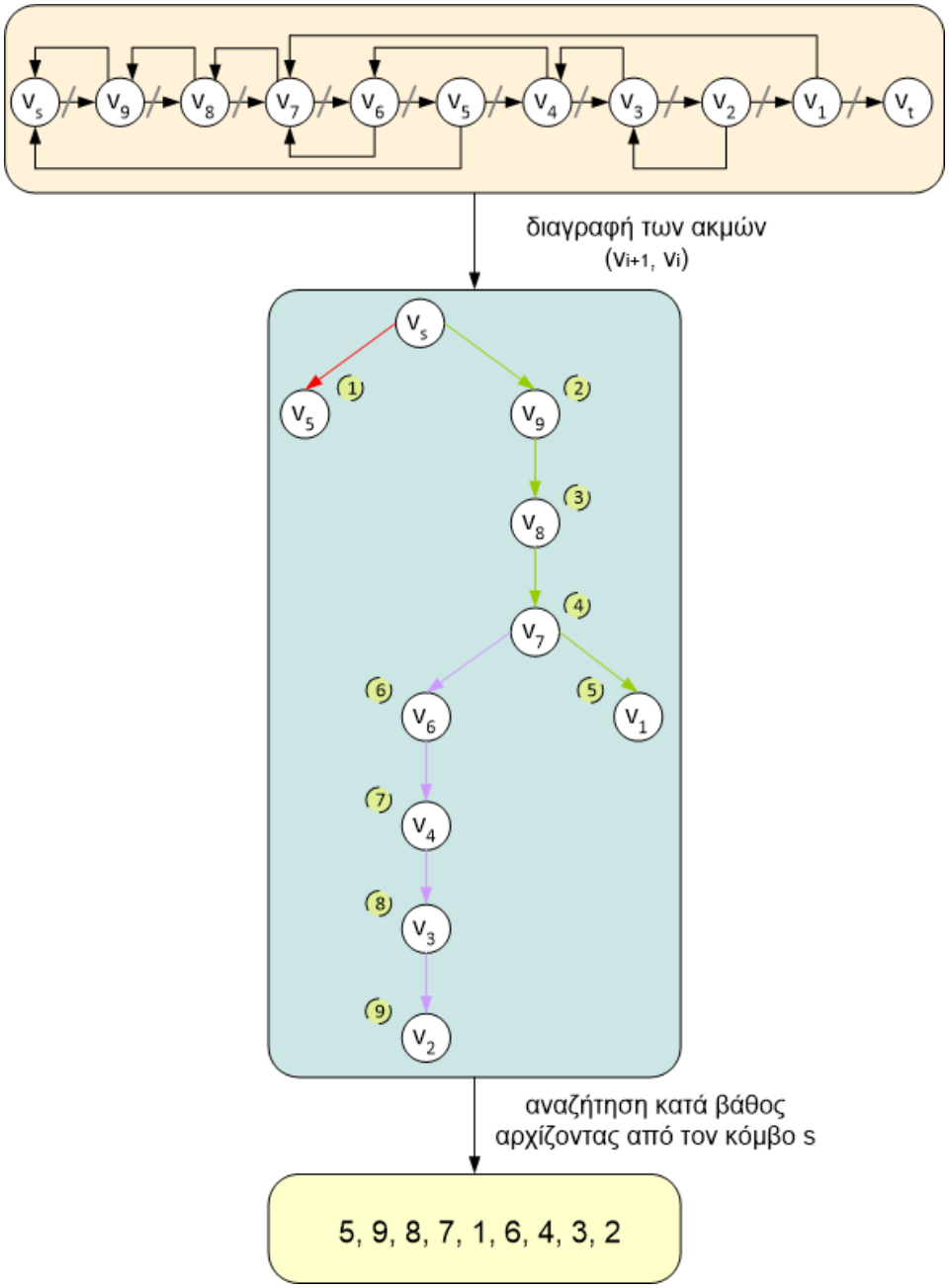
ΟΡΙΖΟΝΤΙΑ ΔΙΕΡΕΥΝΗΣΗΣ(G, s)

---

1. για κάθε κόμβο $u \in V|G| - S$
2. χρώμα$[u] \longleftarrow$ ΛΕΥΚΟ
3. $d[u] \longleftarrow \infty$
4. $\pi[u] \longleftarrow$ ΚΕΝΟ
5. χρώμα$[s] \longleftarrow$ ΓΚΡΙΖΟ
6. $d[s] \longleftarrow 0$
7. $\pi[s] \longleftarrow$ ΚΕΝΟ
8. $Q \longleftarrow \emptyset$
9. προσθήκη(Q, s)
10. όσο$(Q \neq \emptyset)$
11. $u \longleftarrow$ αφαίρεση(Q)
12. για κάθε $v \in Adj[u]$
13. αν χρώμα$[v] =$ ΛΕΥΚΟ
14. χρώμα$[v] \longleftarrow$ ΓΚΡΙΖΟ
15. $d[v] \longleftarrow d[u] + 1$
16. $\pi[v] \longleftarrow u$
17. προσθήκη(Q, v)
18. χρώμα$[u] \longleftarrow$ ΜΕΛΑΝΟ

Παρακάτω παρατίθενται παράδειγμα της τρίτης φάσης αποκωδικοποίησης όπως ακριβώς έγινε και με τις φάσης της κωδικοποίησης. Στοχεύοντας στην διατήρηση της συνοχής με τα προηγούμενα παραδείγματα το αναγώγιμο μεταθετικό γράφημα το οποίο χρησιμοποιείται σε αυτό το παράδειγμα είναι εκείνο το γράφημα το οποίο προήλθε από τον ακέραιο αριθμό 8, όπως ακριβώς έγινε και στα προηγούμενα παραδείγματα (φάση 1 και φάση 2), με στόχο την παράθεση μίας πλήρους διαδικασίας κωδικοποίησης και αποκωδικοποίησης ενός ακεραίου αριθμού σε αναγώγιμο μεταθετικό γράφημα και το αντίστροφο.

Σημειώνεται πως τα κόκκινα βέλη στο δέντρο το οποίο δημιουργείται κατά την διαγραφή των μπροστά ακμών (v_{i+1}, v_i) στο παράδειγμα παίζουν το ρόλο του πρώτου μονοπατιού διάσχισης κατά την διάσχιση κατά βάθος, τα πράσινα βέλη παίζουν τον ρόλο του δεύτερου μονοπατιού διάσχισης και τα μοβ βέλη παίζουν τον ρόλο του τρίτου και τελευταίου μονοπατιού διάσχισης. Επιπλέον, τα νούμερα πάνω από κάθε κόμβο δείχνουν την σειρά με την οποία επισκέφτηκαν για πρώτη φορά οι κόμβοι κατά την διάσχιση αυτή, η οποία είναι και αυτή σύμφωνα με την οποία οι ετικέτες των κόμβων εισέρχονται στην μετάθεση.



Σχήμα 3.3: Παράδειγμα τρίτης φάσης αποκωδικοποίησης.

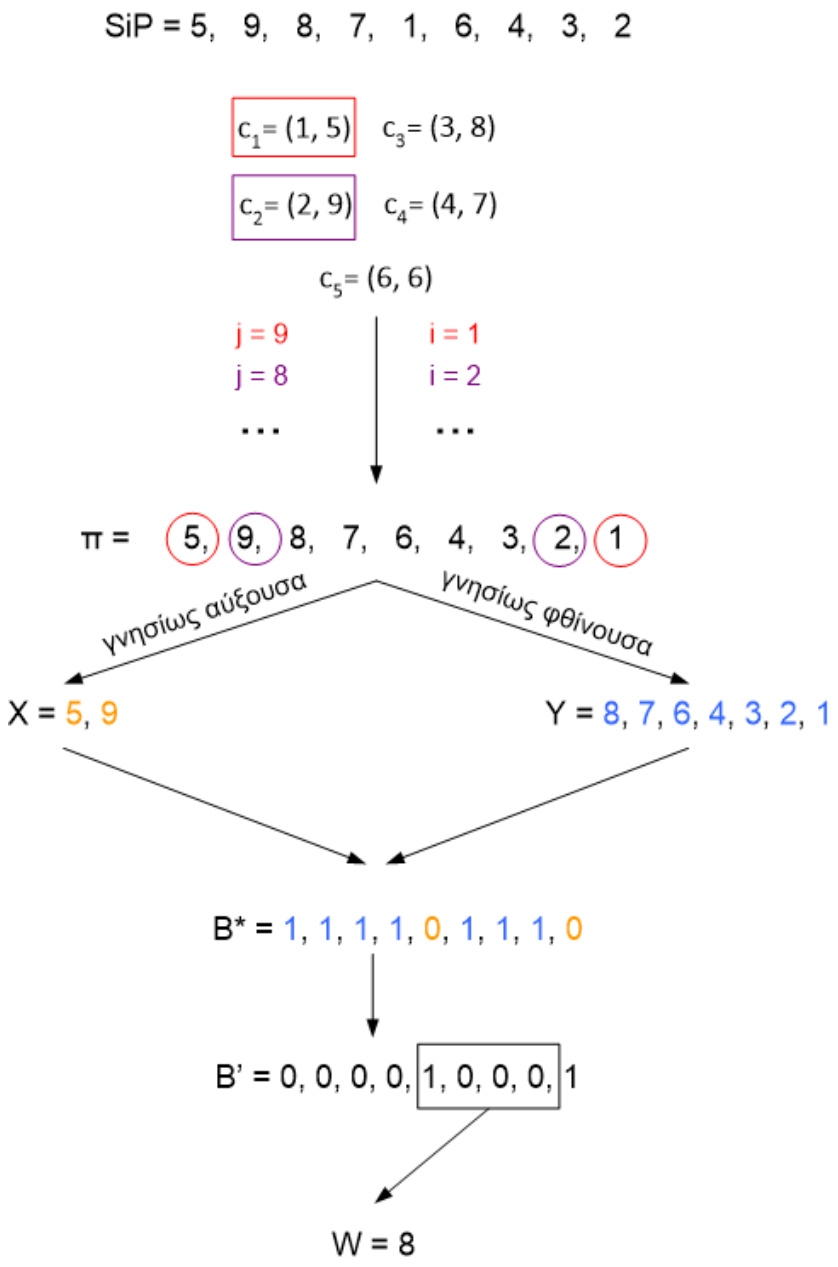
Εν τέλει, ολοκληρώνοντας την διαδικασία της αποκωδικοποίησης κατά την οποία ένα αναγώγιμο μεταθετικό γράφημα αποκωδικοποιείται σε έναν ακέραιο αριθμό, παρουσιάζεται και η τέταρτη και τελευταία φάση στην οποία γίνεται η μεταφορά από την αυτοαναστρέφουσα μετάθεση πίσω στον ακέραιο αριθμό από τον οποίο προήλθε. Υπενθυμίζεται στο σημείο αυτό, πως η φάση αυτή καθώς και όλες οι προηγούμενες, απαιτεί γραμμικό χρόνο εκτέλεσης και χώρο αποθήκευσης. Παρακάτω παρουσιάζονται τα βήματα του αλγορίθμου της φάσης αυτής.

Είσοδος: γράφημα SiP
Εξοδος: μετάθεση w

1. Υπολογισμός της αύξουσας κυκλικής αναπαράστασης $c_1 \prec_2 \prec \ldots \prec c_k$;
2. Εστω $i = 0$, $j = n'$;
3. Κατασκευή της διτονικής μετάθεσης π^b μήκους n' ως εξής:
Οσο το σύνολο C δεν είναι άδειο επανέλαβε
Επιλογή του ελάχιστου στοιχείου c της ακολουθίας C;
Αν ο επιλεγμένος κύκλος c είναι μήκους 2 και έστω $c = (a, b)$ **τότε**
3.1 $\pi_i = b$, $\pi_j = a$;
3.2 $i = i + 1$;
3.3 $j = j - 1$;
Αν ο επιλεγμένος κύκλος c είναι μήκους 1 και έστω $c = (a)$ **τότε**
3.1 $\pi_i = a$;
3.2 $i = i + 1$;
Αφαίρεση του κύκλου c από το σύνολο C;
Τέλος επανάληψης
4. Ευρεση των εξής υποακολουθιών από την π^b;
4.1 $X = \pi_1, \pi_2, \ldots, \pi_k$;
4.2 $Y = \pi_{\varkappa+1}, \pi_{\varkappa+2}, \ldots, \pi_{n'}$;
5. Κατασκευή της $B^* = b_1, b_2, \ldots, b_{n'}$ με 0 στις θέσεις της X και 1 στις θέσεις της Y;
6. Υπολογισμός της ακολουθίας $B' = \bar{B}^*$;
7. Κατασκευή της B αφαιρώντας τα πρώτα n 0 και το τελευταίο 1;
8. Υπολογισμός του ακεραίου αριθμού w από την δυαδική ακολουθία B;
9. **Επιστροφή** του w;

Αλγόριθμος 4: Τέταρτη φάση αποκωδικοποίησης $Decode\_SiP\_to\_W$

Στο σημείο αυτό ολοκληρώνοντας και την τέταρτη και τελευταία φάση αποκωδικοποίησης παρατίθεται και το αντίστοιχο παράδειγμα. Σε αυτό απεικονίζεται μία αυτοαναστρέφουσα μετάθεση προερχόμενη από το προηγούμενο παράδειγμα της τρίτης φάσης, η οποία αποκωδικοποιείται στον ακέραιο αριθμό στον οποίο αντιστοιχεί. Τονίζεται πως κάθε αυτοαναστρέφουσα μετάθεση η οποία έχει παραχθεί από αυτόν τον αλγόριθμο, αντιστοιχεί μονάχα σε έναν ακέραιο αριθμό. Σημειώνεται επίσης, πως στο παρόν παράδειγμα τα κόκκινα και το μοβ ορθογώνια παραλληλόγραμμα / κύκλοι αντιστοιχούν στην πρώτη και στη δεύτερη επανάληψη του αλγορίθμου.

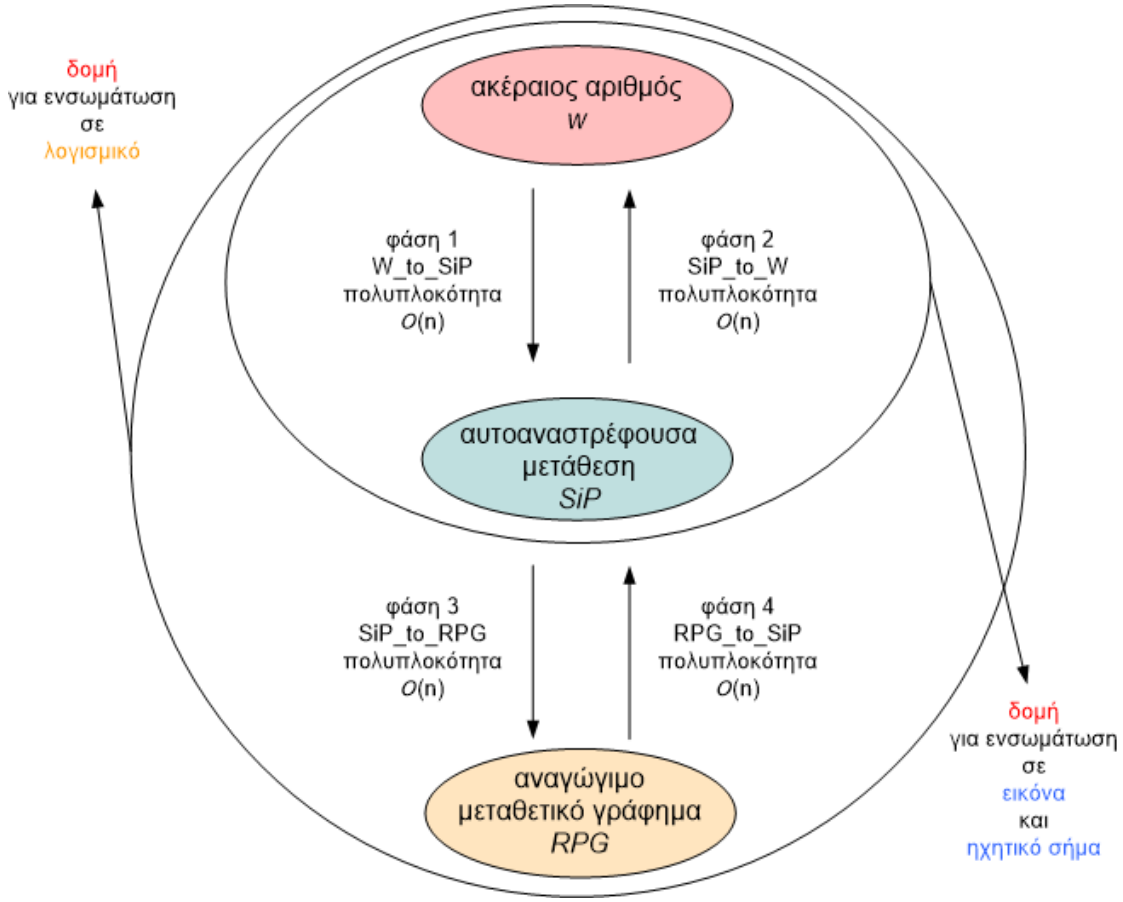


Σχήμα 3.4: Παράδειγμα τέταρτης φάσης αποκωδικοποίησης.

Ο αλγόριθμος Chroni & Nikolopoulos επί της ουσίας κρύβει υπό τον μανδύα μίας δομής (γράφημα) έναν ακέραιο αριθμό. Η δομή αυτή όπως θα γίνει φανερό στα επόμενα κεφάλαια θα αποτελέσει το υδατογράφημα το οποίο βάσει ενός μοντέλου ονόματι WaterRPG θα ενσωματωθεί μέσα σε λογισμικό. Ο αριθμός αυτός αποτελεί και την πολυπόθητη πληροφορία η οποία σε μεταγενέστερο στάδιο (αφού έχει ενσωματωθεί η δομή στον κώδικα λογισμικού) πρέπει να εξαχθεί προκειμένου να λειτουργήσει το μοντέλο αυτό υδατογράφησης λογισμικού.

Το γράφημα αυτό ενσωματώνεται στον κώδικα λογισμικού αφού προηγουμένως έχει μετατραπεί σε κώδικα. Κατά καιρούς σε επιστημονικές εργασίες συνεδρίων και περιοδικών

έχουν παρουσιαστεί τεχνικές οι οποίες από δομές (π.χ., γραφήματα, πίνακες, λίστες) παράγουν κώδικα [22] [43] [36]. Βέβαια ο παραγόμενος αυτός κώδικας από τη μία πλευρά ενώ είναι καθόλα λειτουργικός δίχως να δημιουργεί προβλήματα εκτέλεσης στον υπόλοιπο κώδικα, από την άλλη όμως δεν είναι διαφανές (stealthy) έναντι του κώδικα που ενσωματώνεται καθώς είναι κίβδηλος (dummy), με αποτέλεσμα να δίνει γρήγορα έναυσμα σε κακόβουλους χρήστες να ασχοληθούν μαζί του. Το γεγονός αυτό αποτελεί ένα μεγάλο μειονέκτημα στην ενσωμάτωση του υδατογραφήματος, καθώς το καθιστά ευάλωτο και ιδιαίτερα τρωτό σε επιθέσεις.



Σχήμα 3.5: Φάσεις και εφαρμογές του αλγορίθμου Chroni & Nikolopoulos.

3.1.2 Ανάλυση του Αλγορίθμου Chroni & Nikolopoulos

Ο αλγόριθμος Chroni & Nikolopoulos κατασκευάζει σε γραμμικό χρόνο και χώρο αναγώγιμα μεταθετικά γραφήματα από ακεραίους αριθμούς και το αντίστροφο. Η διαδικασία αυτή όπως έχει γίνει ήδη φανερό στα προηγούμενα κεφάλαια χωρίζεται σε τέσσερις διακριτές φάσεις (δύο για την κωδικοποίηση και δύο για την αποκωδικοποίηση). Αυτά τα γραφήματα

φέρουν ιδιότητες οι οποίες τους δίνουν μοναδικό χαρακτήρα και εφόδια αντίστασης έναντι τροποποιήσεων στη δομή τους. Οι ιδιότητες αυτές, με την εισαγωγή των γραφημάτων αυτών σε λογισμικά περνούν ή αλλιώς μεταβιβάζονται στους κώδικές τους. Αυτές είναι οι ακόλουθες:

- Τα γραφήματα αυτά έχουν μοναδικό hamiltonian μονοπάτι. Η ιδιότητα αυτή είναι ιδιαίτερα χρήσιμη στην ανακατασκευή του γραφήματος σε καταστάσεις όπου έχουν τροποποιηθεί οι ετικέτες των κόμβων.

- Τα γραφήματα αυτά, ανεξάρτητα από τον ακέραιο αριθμό τον οποίο κωδικοποιούν, έχουν πάντα περιττό πλήθος κόμβων.

- Κάθε κόμβος των γραφημάτων αυτών έχει δύο εξερχόμενες ακμές, μία ακμή στον κόμβο με την αμέσως μικρότερη ετικέτα (μπροστά ακμή) και μία ακμή σε κάποιον κόμβο με μεγαλύτερη ετικέτα (πίσω ακμή) εκτός από τον κόμβο με την ετικέτα s ο οποίος έχει μονάχα μία εξερχόμενη ακμή και συγκεκριμένα μία μπροστά ακμή στον κόμβο με την μεγαλύτερη ετικέτα όλου του γραφήματος και του κόμβου με την ετικέτα t ο οποίος δεν έχει καθόλου εξερχόμενες ακμές.

- Τα γραφήματα αυτά έχουν προέλθει από αυτοαναστρέφουσες μεταθέσεις οι οποίες είναι περιττού μήκους, έχουν όλους τους κύκλους τους με μήκους δύο εκτός από έναν με μήκος ένα.

- Τα γραφήματα αυτά έχουν προέλθει από διτονικές μεταθέσεις μήκους περιττού, οι οποίες είναι πάντα στην αρχή γνησίως αύξουσες και ύστερα γίνονται γνησίως φθίνουσες.

- Τα γραφήματα αυτά έχουν προέλθει από δυαδικές μεταθέσεις περιττού μήκους, έστω $n' = 2 * n + 1$ οι οποίες τα πρώτα n στοιχεία τους τα έχουν όλα άσσους και το τελευταίο στοιχείο τους είναι πάντα 0.

Στο σημείο αυτό σημειώνεται πως μονάχα τα αναγώγιμα μεταθετικά γραφήματα τα οποία παράγονται από τον αλγόριθμο Chroni & Nikolopoulos φέρουν όλες τις ιδιότητες αυτές και όχι όλη η κλάση των γραφημάτων αυτών. Επομένως, έχοντας τα γραφήματα αυτά όλες αυτές τις ιδιότητες γίνεται φανερό πως δύσκολα θα συμβεί σε αυτά κάποια τροποποίηση δίχως να μπορεί να αυτοδιορθωθεί ή τουλάχιστον να μπορεί να εντοπιστεί. Επιπλέον λόγω του γεγονότος πως μία από τις ιδιότητές τους είναι το μοναδικό hamiltonian μονοπάτι δεν απαιτείται επιπλέον πληροφορία για την ενσωμάτωση κάθε κόμβου στον κώδικα του λογισμικού. Με απλά λόγια δηλαδή, δεν χρειάζεται να κρατείται πληροφορία για τις ετικέτες των κόμβων κατά την διάρκεια ενσωμάτωσής τους στον κώδικα, πράγμα που θα γίνει καλύτερα κατανοητό στη συνέχεια της διατριβής.

Αξίζει να σημειωθεί επίσης πως οι παραπάνω ιδιότητες ενσωματώνοντας το αναγώγιμο μεταθετικό γράφημα στον κώδικα μεταφέρονται σε αυτό. Το γεγονός αυτό σημαίνει πως αν είναι δύσκολο οι επιθέσεις που θα συμβούν στο υδατογράφημα αυτό καθ αυτό πριν την

ενσωμάτωσή του να αλλοιώσουν τη δομή του, είναι ακόμη πιο δύσκολο οι επιθέσεις που θα συμβούν στο κώδικα να αλλοιώσουν τη δομή του. Αυτό ισχύει, καθώς στον κώδικα είναι πολύ πιθανόν να τροποποιηθούν τμήματά του τα οποία δεν σχετίζονται με τον κώδικα του υδατογραφήματος.

3.1.3 Επιθέσεις στο Αναγώγιμο Μεταθετικό Γράφημα

Ο αλγόριθμος Chroni & Nikolopoulos δέχεται ως είσοδο έναν ακέραιο αριθμό και τον μετατρέπει σε ένα αναγώγιμο μεταθετικό γράφημα στην έξοδό του χρησιμοποιώντας αυτοαναστρέφουσα μετάθεση σε γραμμικό χρόνο $O(n)$. Η δομή αυτή μπορεί να δεχτεί επιθέσεις ή αλλιώς τροποποιήσεις οι οποίες χωρίζονται σε τρεις διαφορετικές κατηγορίες είτε σε συνδυασμό, είτε αυτούσιες. Οι κατηγορίες αυτές είναι οι εξής:

- Επίθεση στους κόμβους του γραφήματος.
- Επίθεση στις ακμές του γραφήματος.
- Επίθεση στις ετικέτες που φέρουν οι κόμβοι του γραφήματος.

Συγκεκριμένα διαπράττοντας επίθεση στους κόμβους του γραφήματος, επί της ουσίας προστίθενται ή αφαιρούνται κόμβοι στο σύνολο των κόμβων. Προφανώς με την αφαίρεση κόμβων αφαιρούνται ταυτόχρονα και οι ακμές οι οποίες πρόσκεινται σε αυτούς. Διαπράττοντας επίθεση στις ακμές του γραφήματος προστίθενται ή αφαιρούνται ακμές στο σύνολο των ακμών. Σημειώνεται πως αναποδογυρίζοντας (flip) μία ακμή, από έναν κόμβο x σε ένα κόμβο y, αυτό που συμβαίνει στην πραγματικότητα είναι η αφαίρεση αυτής και η προσθήκη νέας ακμής από τον κόμβο y στον κόμβο x. Ενώ διαπράττοντας επίθεση στις ετικέτες των κόμβων αυτές είτε αλλάζουν, είτε διαγράφονται εντελώς.

Η δομή αυτή ύστερα από εμπεριστατωμένη πειραματική μελέτη είναι ιδιαίτερα ανθεκτική σε τροποποιήσεις στη δομή της [4] [15]. Σε επιθέσεις στους κόμβους κύριο ρόλο σε αυτό παίζει το πάντα περιττό πλήθος κόμβων που εμπεριέχονται στο σύνολο των κόμβων της δομής. Σε επιθέσεις στις ακμές σημαντικό ρόλο παίζει ο σταθερός αριθμός ακμών ανά κόμβο καθώς και ότι πάντα υπάρχει μία μπροστά ακμή και και μάλιστα σε κόμβο με την αμέσως μικρότερη ετικέτα και πάντα μία πίσω. Ενώ σε επιθέσεις στις ετικέτες των κόμβων καθοριστικό ρόλο παίζει το μοναδικό hamiltonian μονοπάτι. Αξίζει επίσης να σημειωθεί πως ακόμα και να πραγματοποιηθεί κάποια επίθεση στο γράφημα διατηρώντας παράλληλα τη δομή του ανέπαφη, η επίθεση αυτή με πάρα πολύ μεγάλη πιθανότητα θα εντοπιστεί στις δομές που δημιουργούνται κατά την αντίστροφη διαδικασία (φάσεις 3 και 4) κατά την οποία το αναγώγιμο μεταθετικό γράφημα αποκωδικοποιείται στον ακέραιο αριθμό από τον οποίον και προήλθε (π.χ., θα εντοπιστεί στην δομή της αυτοαναστρέφουσας μετάθεσης ή της διτονικής μετάθεσης).

3.1.4 Αρχική Προσέγγιση του Μοντέλου Υδατογράφησης

Εν αρχή προκειμένου να υδατογραφηθεί αποτελεσματικά ένα λογισμικό, πριν ακόμα εφαρμοστεί κάποιο μοντέλο υδατογράφησης σε αυτό, είναι απαραίτητο να ακολουθηθεί μία

στρατηγική βάσει της οποίας θα εξαχθούν συμπεράσματα χρήσιμα σχετικά με τον προγραμματισμό που θα ακολουθήσει (εισαγωγή υδατογραφήματος στον κώδικα). Η στρατηγική αυτή, σχετίζεται με τον τρόπο με τον οποίο πρέπει να σκέφτεται και να ενεργεί οποιοσδήποτε στοχεύει να προστατέψει τα πνευματικά δικαιώματα ενός λογισμικού από την στιγμή που το παίρνει για πρώτη φορά στα χέρια του, μέχρι και την στιγμή που το εξάγει στην αγορά με ενσωματωμένο το υδατογράφημα σε αυτό.

Όπως με τις δομές υδατογράφησης αλλά και τις τεχνικές υδατογράφησης λογισμικού, έτσι και με τις στρατηγικές δεν υπάρχει κάποια μαγική συνταγή η οποία να παρέχει τα βέλτιστα δυνατά αποτελέσματα. Στόχος κάθε φορά είναι να εξαχθούν πληροφορίες οι οποίες θα φανούν χρήσιμες στην επιλογή του καλύτερου και πιο συμβατού υδατογραφήματος πάντα συγκριτικά με τον κώδικα στον οποίον και θα ενσωματωθεί, αλλά και στον προγραμματισμό που θα ακολουθήσει προκειμένου το υδατογράφημα να κωδικοποιηθεί στον κώδικα. Τα στάδια από τα οποία αποτελείται η στρατηγική που προτείνεται από την παρούσα διατριβή είναι τα ακόλουθα:

- Μελέτη και πλήρης κατανόηση του κώδικα προς υδατογράφηση και από αλγοριθμική σκοπιά και από προγραμματιστική, με σκοπό να εντοπιστούν τα υποψήφια σημεία του κώδικα στα οποία θα μπορούσε να ενσωματωθεί το υδατογράφημα.

- Εκτέλεση του κώδικα με μία πληθώρα ακολουθιών εισόδου προκειμένου να καταγραφούν οι τιμές των χρόνων εκτέλεσης και να βγει ένας μέσος όρος αυτών καθώς και εκείνες οι ακολουθίες οι οποίες κάνουν τον μικρότερο και τον περισσότερο χρόνο εκτέλεσης.

- Κατασκευή υδατογραφήματος σύμφωνα με συγκεκριμένο αλγόριθμο. Το υδατογράφημα πρέπει να χαρακτηρίζεται από συγκεκριμένες ιδιότητες προκειμένου να είναι ανθεκτικό σε ενδεχόμενες επιθέσεις κακόβουλων χρηστών.

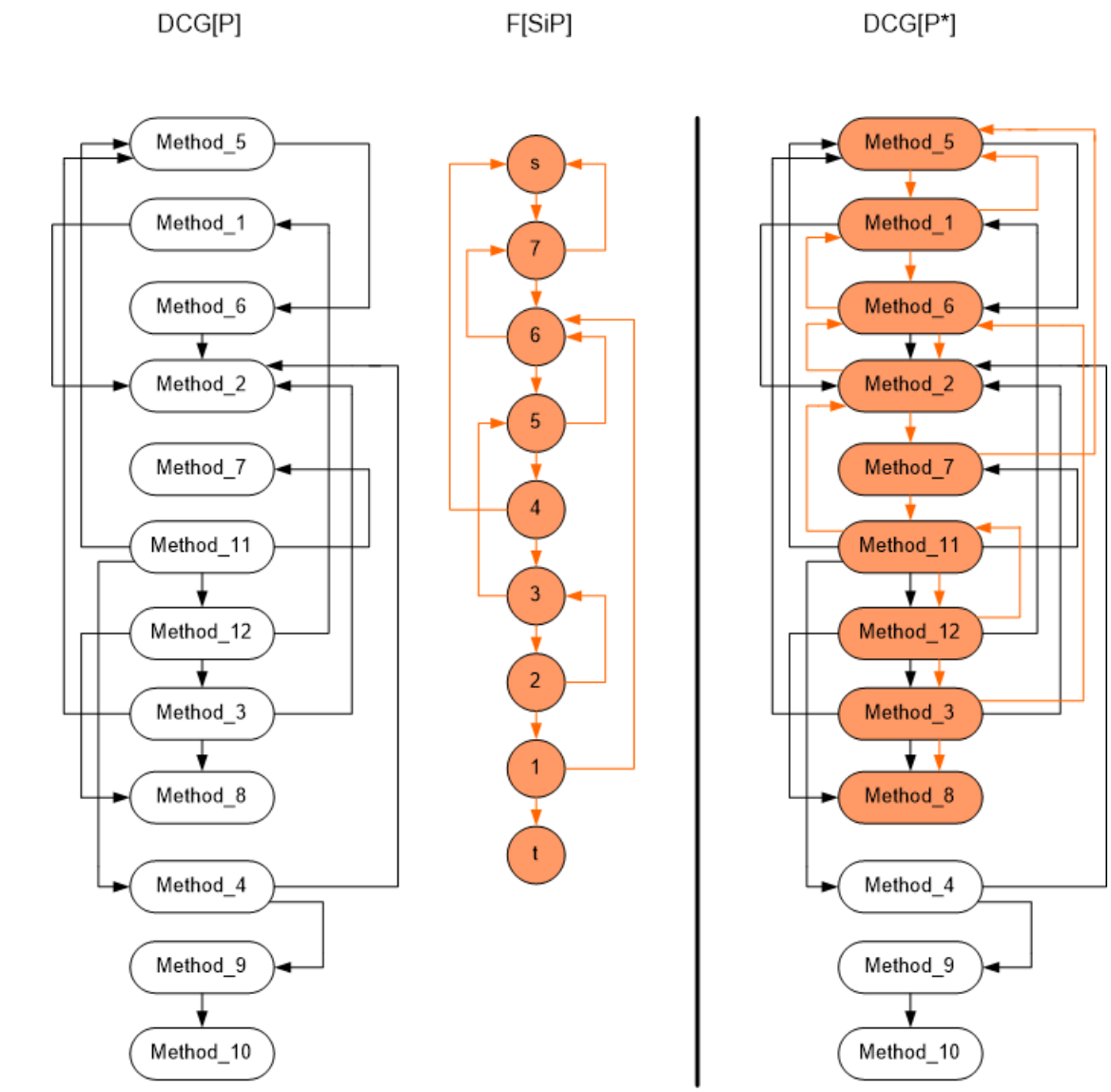
- Συλλογή εργαλείων που παρέχει ο κώδικας (π.χ., συναρτήσεις, μεταβλητές, πίνακες, νήματα, αδιαφανή κατηγορήματα, κλπ.) τα οποία θα φανούν χρήσιμα για την τεχνική υδατογράφησης.

- Συνδυασμός των συλλεχθέντων εργαλείων με σκοπό να εφαρμοστούν στον κώδικα (είτε στον πηγαίο, είτε στον εκτελέσιμο) του λογισμικού με τέτοιον τρόπο έτσι ώστε να τον προστατέψουν όσο το δυνατόν αποτελεσματικότερα (εφαρμογή διαδικασιών οι οποίες χειρίζονται και αξιοποιούν τα εργαλεία αυτά) κωδικοποιώντας πληροφορία μέσα στο λογισμικό.

Συγκεκριμένα, ακολουθώντας την εν λόγω στρατηγική, αυτός που εφαρμόζει την τεχνική της υδατογράφησης, έχοντας μελετήσει και κατανοήσει πλήρως τον κώδικα του λογισμικού, δημιουργώντας εν συνεχεία και το κατάλληλο υδατογράφημα, κύριο μέλημά του είναι να συλλέξει τα κατάλληλα εργαλεία που θα χρειαστεί. Στη συνέχεια τα εργαλεία αυτά θα συνδυαστούν μέσω κατάλληλων διαδικασιών υδατογράφησης όσο πιο αποτελεσματικά είναι εφικτό με στόχο την ενσωμάτωση πληροφορίας μέσα στο λογισμικό. Με απλά λόγια, ο

συνδυασμός των εργαλείων ουσιαστικά είναι η ανάπτυξη ενός μοντέλου μέρος του οποίου αποτελεί και ο αλγόριθμος υδατογράφησης.

Σημειώνεται πως στην εργασία αυτή προκειμένου να δημιουργηθεί ένα υδατογράφημα χρησιμοποιείται ο αλγόριθμος Chroni & Nikolopoulos (κωδικοποίηση ακεραίου αριθμού σε μεταθετικό αναγώγιμο γράφημα - RPG και το αντίστροφο μέσω αυτοαναστρέφουσας μετάθεσης - SiP) [13]. Υπάρχουν και άλλοι αλγόριθμοι οι οποίοι μετατρέπουν έναν ακέραιο αριθμό σε ένα μεταθετικό αναγώγιμο γράφημα [22] [23] αλλά το κύριο μειονέκτημά τους είναι πως δεν μπορούν να αντισταθούν σε επιθέσεις στο βαθμό που αντιστέκονται τα γραφήματα τα οποία παράγονται από τον αλγόριθμο Chroni & Nikolopoulos (π.χ., προσθήκη / διαγραφή ακμών, προσθήκη / διαγραφή κόμβων, τροποποίηση των ετικετών των κόμβων).



Σχήμα 3.6: Αρχική προσέγγιση μοντέλου υδατογράφησης λογισμικού WaterRPG.

Είναι κοινώς αποδεκτό στον τομέα της προστασίας λογισμικού πως οι υδατογραφήσεις σε λογισμικά παρουσιάζουν αρκετά προβλήματα κατά την εφαρμογή τους (implementation) και πολλές από τις ήδη υπάρχουσες τεχνικές και μοντέλα είναι ιδιαίτερα ευάλωτα σε σχετικά απλές επιθέσεις κακόβουλων χρηστών. Ενώ δηλαδή έχουν τεκμηριωθεί σωστά και πολλές από αυτές έχουν δώσει είτε θεωρητική, είτε πειραματική απόδειξη, επί του πρακτέος (στον προγραμματισμό) φαίνεται να παρουσιάζουν προβλήματα (π.χ., το ενσωματωμένο υδατογράφημα είναι ιδιαίτερα εμφανές, δεν χαρακτηρίζεται από διαφάνεια δηλαδή ή αλλοιώνεται ύστερα από απλές τροποποιήσεις των σημασιολογικών δομών του κώδικα). Εξού και το γεγονός πως ακόμη δεν έχει βρεθεί κάποια πλήρως αποτελεσματική μέθοδος υδατογράφησης. Το συμπέρασμα αυτό προκύπτει από διάφορα ενδεχόμενα, κάποια από τα οποία είναι τα ακόλουθα:

- Δεν χρησιμοποιήθηκε ανθεκτικό υδατογράφημα σε επιθέσεις. Το υδατογράφημα δεν είχε τις κατάλληλες ιδιότητες με αποτέλεσμα ύστερα από επιθέσεις εισβολέων αλλοιώθηκε ανεπανόρθωτα (δεν ήταν πλέον αναγνωρίσιμο από το λογισμικό αναγνώρισης).

- Δεν χρησιμοποιήθηκε αποτελεσματική τεχνική υδατογράφησης λογισμικού. Ασχέτως με την αποτελεσματικότητα του υδατογραφήματος ως δομή, η τεχνική υδατογράφησης ή ο προγραμματισμός που έλαβε χώρα για να ενσωματωθεί στον κώδικα του λογισμικού άφηνε κενά (τρωτά σημεία) τα οποία οι κακόβουλοι χρήστες μπορούσαν εύκολα να εκμεταλλευτούν.

- Δεν μελετήθηκε στον βαθμό που θα έπρεπε ο κώδικας (είτε ο πηγαίος, είτε ο εκτελέσιμος) του λογισμικού. Υπήρχαν καλύτερα σημεία μέσα στον κώδικα του λογισμικού στα οποία θα μπορούσε να είχε ενσωματωθεί το υδατογράφημα, προκειμένου να μην γίνεται εύκολα αντιληπτό από εισβολείς και ο χρόνος εκτέλεσης καθώς και ο χώρος αποθήκευσης να διατηρούνται σε χαμηλότερα επίπεδα.

- Δεν εφαρμόστηκε η κατάλληλη στρατηγική υδατογράφησης λογισμικού. Η στρατηγική υδατογράφησης δεν λάμβανε υπόψη ορισμένα βασικά στάδια ή περιείχε στάδια με λανθασμένη σειρά εφαρμογής.

Οι πρώτες σκέψεις οι οποίες έγιναν προκειμένου να δημιουργηθεί το μοντέλο υδατογράφησης λογισμικού WaterRPG στηριζόντουσαν σε σημεία σήμανσης (marks) και στην ιεραρχία κλήσεων (call hierarchy) του λογισμικού. Εμβαθύνοντας στην προσέγγιση αυτή γίνεται φανερό πως βάσει της ιεραρχίας κλήσεων και των σημαδιών τα οποία προστίθονταν μέσα σε ένα υποσύνολο συναρτήσεων του λογισμικού, καθίσταται εφικτό να ενσωματωθεί και να εξαχθεί η πληροφορία που αντιστοιχεί στο υδατογράφημα. Η πληροφορία αυτή παίρνει την μορφή γραφήματος το οποίο συσσωματώνεται με τον αρχικό κώδικα του λογισμικού με τέτοιον τρόπο έτσι ώστε να μην αλλοιώνεται η λειτουργικότητά του και να μην δίνει πολλά περιθώρια σε εισβολείς να το διακρίνουν.

Με τον όρο ιεραρχία κλήσεων, εννοείται το γράφημα που δημιουργείται από τις κλήσεις που εμπεριέχουν όλες οι συναρτήσεις του κώδικα σε άλλες συναρτήσεις. Οι κλήσεις αυτές,

μετά το πέρας της υδατογράφησης είτε μπορεί να είναι πραγματικές (real calls), είτε μπορεί να είναι εικονικές (water calls). Εικονική κλήση είναι όποια κλήση δεν υπήρχε εξ αρχής στον κώδικα, αλλά προστέθηκε στη συνέχεια. Σημειώνεται επίσης πως για να επιλεγούν τα κατάλληλα σημεία σήμανσης μέσα στις επιθυμητές μεθόδους, πρέπει να δημιουργηθούν τα διαγράμματα ελέγχου ροής (control flow graphs) για κάθε μία από αυτές και εν συνεχεία με βάση συγκεκριμένα κριτήρια σήμανσης να επιλεγούν τα σημεία. Συγκεκριμένα, στα σημεία αυτά θα φιλοξενείται κώδικας ο οποίος κατά την διάρκεια εκτέλεσης του λογισμικού μέσω μίας ακολουθίας εισόδου (κλειδί) θα σηματοδοτεί σε ποια συνιστώσα του υδατογραφήματος βρίσκεται κάθε στιγμή η ροή εκτέλεσης. Υπάρχουν διάφορες τεχνικές σήμανσης [22] σε κώδικα ορισμένες από τις οποίες είναι οι ακόλουθες:

- Προσθήκη κώδικα ο οποίος δεν επιδρά καθόλου στην λειτουργικότητα του κώδικα (π.χ., φορτώνοντας μία τιμή η οποία ποτέ δεν χρησιμοποιείται από τον αρχικό κώδικα κατά την διάρκεια εκτέλεσης του προγράμματος).

- Καταμέτρηση των εντολών του κώδικα μέσα σε ένα τμήμα (block) και χρησιμοποίηση της ισοτιμία (parity) ως σημάδι. Προσθήκη εντολών οι οποίες δεν κάνουν τίποτα άλλο από το να καταναλώνουν χρόνο στην κεντρική μονάδα επεξεργασίας (central processing unit - CPU) όπως η εντολή nop προκειμένου να φτιαχτεί κατάλληλα το σημάδι.

- Καταμέτρηση των προσβάσεων σε στατικές μεταβλητές (static variables) για να καθοριστεί το σημάδι. Προσθήκη επιπρόσθετων μεταβλητών και προσβάσεων για να φτιαχτεί κατάλληλα το σημάδι.

- Υπολογισμός αθροίσματος ελέγχου (ckecksum) των εντολών και χρησιμοποίηση ενός ή περισσοτέρων bits από αυτό ως σημάδι.

- Μετατροπή της ακολουθίας εντολών μέσα σε κάθε τμήμα κώδικα (block) σε κανονική μορφή (canonical form) και στη συνέχεια τροποποίηση αυτής της μορφής συστηματικά (systematically) για να κωδικοποιηθούν τα σημάδια.

- Προσθήκη σημαδιών στις μετα πληροφορίες (meta data) οι οποίες συνδέονται με κάθε ένα τμήμα κώδικα (block) (π.χ., τροποποίηση των εντολών για την αποσφαλμάτωση).

Αξίζει να σημειωθεί πως στην περίπτωση που υπάρχουν δύο συναρτήσεις στον κώδικα του λογισμικού έστω η f_X και η f_Y στις οποίες υπάρχει εξ αρχής κλήση από την f_X στην f_Y και προστεθεί κατά την διαδικασία υδατογράφησης μία ή περισσότερες επιπλέον κλήσεις (water calls) στη νέα ιεραρχία κλήσεων δεν θα δημιουργηθεί νέα ακμή από την συνάρτηση f_X στην f_Y. Το γεγονός αυτό, με μία πρώτη σκέψη φαίνεται να δημιουργεί πρόβλημα στην αναγνώριση του υδατογραφήματος. Στο σημείο αυτό όμως χρησιμοποιούνται και τα σημεία σήμανσης που περιέχουν οι επιλεγμένες συναρτήσεις, προκειμένου να προσδιοριστούν ποιες συναρτήσεις ανήκουν στο υποσύνολο των συναρτήσεων που χρησιμοποιήθηκαν για την διαδικασία υδατογράφησης, ποιες είναι οι επιπρόσθετες κλήσεις που περιέχουν στον κώδικά τους καθώς επίσης και με ποια σειρά σαρώνεται το ενσωματωμένο υδατογράφημα.

Στην ουσία η αρχική προσέγγιση ήταν ένα μοντέλο το οποίο ενσωμάτωνε υδατογραφήματα τα οποία έφεραν την στατική ιδιότητα και συγκεκριμένα τα υδατογραφήματα αυτά ήταν υδατογραφήματα κώδικα (code watermarks). Αντιστοιχούσε ένα υποσύνολο συναρτήσεων του κώδικα με τους κόμβους του αναγώγιμου μεταθετικού γραφήματος που θα ενσωματώνονταν στον κώδικα και τις ακμές αυτού τις αντιστοιχούσε με κλήσεις συναρτήσεων οι οποίες προστίθονταν μία προς μία προκειμένου να κωδικοποιήσουν το γράφημα στην ιεραρχία κλήσεων. Σε κάθε μία κλήση συνάρτησης τοποθετούνταν και ένα σημείο σήμανσης με στόχο την μελλοντική της αναγνώριση ως κλήση συνάρτησης η οποία αντιστοιχεί σε ακμή του γραφήματος. Όλες οι κλήσεις συναρτήσεων οι οποίες κωδικοποιούσαν τις ακμές του ενσωματωμένου γραφήματος ήταν κίβδηλες (dummy) και εμπεριέχονταν εντός συνθηκών ελέγχου προκειμένου να εκτελούνται μονάχα μία φορά η κάθε μία προς αποφυγή σπατάλης χρόνου [14].

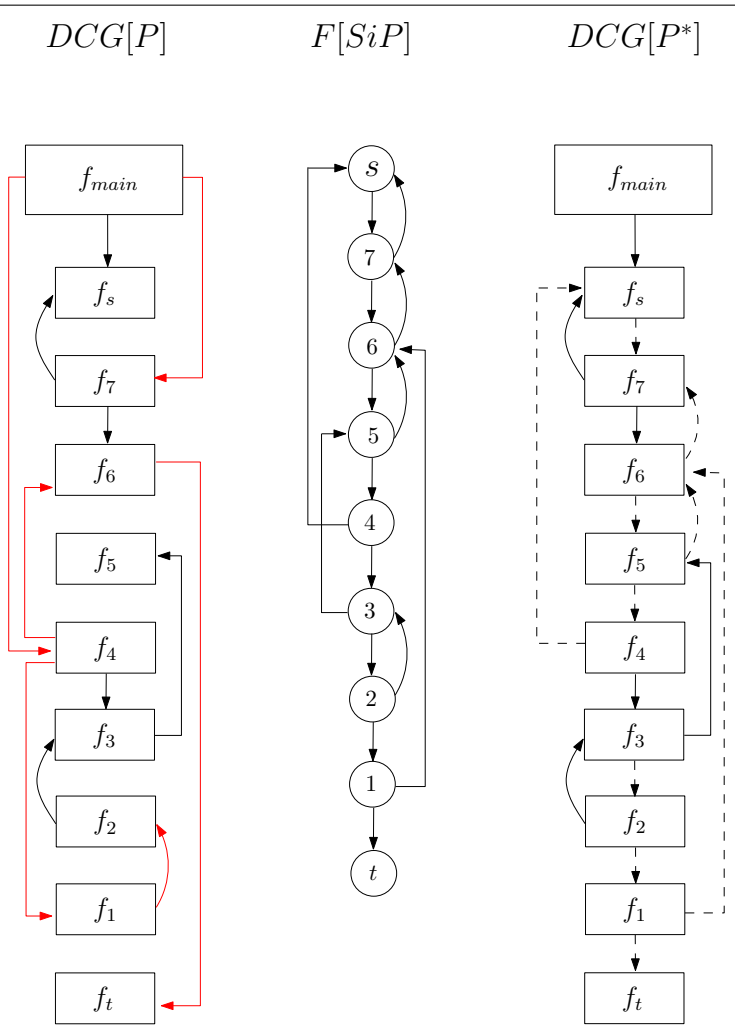
Προκειμένου όμως να αποφευχθεί ο εύκολος εντοπισμός των συνθηκών ελέγχου οι οποίες εμπεριείχαν τις κίβδηλες κλήσεις συναρτήσεων από εισβολείς, στο εσωτερικό των συνθηκών αυτών προστίθονταν και επιπλέον κίβδηλος κώδικας (dummy code) ο οποίος εκτελούνταν, αλλά δεν επιδρούσε αρνητικά στην λειτουργικότητα του προγράμματος. Σοβαρά μειονεκτήματα όμως αυτής της προσέγγισης ήταν πρώτον πως σε περίπτωση που αφαιρεθούν οι επιπρόσθετες κλήσεις συναρτήσεων το υδατογράφημα αλλοιώνεται, ενώ ο κώδικας διατηρείται λειτουργικός και δεύτερον ο επιπλέον κώδικας είναι ασυσχέτιστος με τον αρχικό κώδικα γεγονός που τον καθιστά εύκολα αναγνωρίσιμο.

3.2 Βασική Ιδέα του WaterRPG Μοντέλου

Λαμβάνοντας υπόψη όλα τα μειονεκτήματα της αρχικής προσέγγισης και ιδιαίτερα το μεγάλο πρόβλημα που δημιουργούσαν τα σημεία σήμανσης το μοντέλο υδατογράφησης WaterRPG κατέληξε να μετατραπεί σε ένα δυναμικό μοντέλο υδατογράφησης λογισμικού απαλλαγμένο πλέον από ένα μεγάλο σύνολο μειονεκτημάτων. Το δυναμικό μοντέλο υδατογράφησης λογισμικού WaterRPG στηρίζεται σε μία ρηξικέλευθη ιδέα, η οποία κωδικοποιεί το υδατογράφημα στο δυναμικό γράφημα κλήσεων (dynamic call graph) που δημιουργείται από την εκτέλεση του κώδικα μέσω μίας συγκεκριμένης ακολουθίας εισόδου. Λόγω του γεγονότος ότι το υδατογράφημα (reducible permutation graph - RPG) υπάρχει μέσα στον κώδικα του λογισμικού υπό τη μορφή κλήσεων συναρτήσεων είναι ιδιαίτερα δύσκολο να αλλοιωθεί καθώς για να συμβεί αυτό πρέπει να επέλθει τροποποίηση της ροής εκτέλεσης του κώδικα και συγκεκριμένα τροποποίηση της αλληλουχίας των κλήσεων συναρτήσεων που εκτελούνται, πράγμα το οποίο απαιτεί πλήρη κατανόηση κώδικα.

Σύμφωνα με το μοντέλο αυτό δεν προστίθεται ούτε κίβδηλος κώδικας (dummy code), αλλά ούτε και νεκρός κώδικας (dead code). Απεναντίας αυτό που γίνεται στην πραγματικότητα είναι να τροποποιείται ο ήδη υπάρχον κώδικας, προκειμένου δοθείσας μίας ακολουθίας εισόδου κλειδί, ως αποτέλεσμα να παράγεται ένα δυναμικό γράφημα κλήσεων ισομορφικό με το ενσωματωμένο υδατογράφημα το οποίο να παράγει πανομοιότυπο αποτέλεσμα με το αρχικό μη υδατογραφημένο πρόγραμμα. Επίσης δεν γίνεται χρήση και σημείων σήμανσης,

καθώς με την παραγωγή του δυναμικού γραφήματος κλήσεων είναι εφικτό να αναγνωριστεί το υδατογράφημα δίχως περαιτέρω στοιχεία όπως συγκεκριμένα σημάδια, παρά μόνο παρακολουθώντας και καταγράφοντας τις κλήσεις συναρτήσεων που συμβαίνουν.



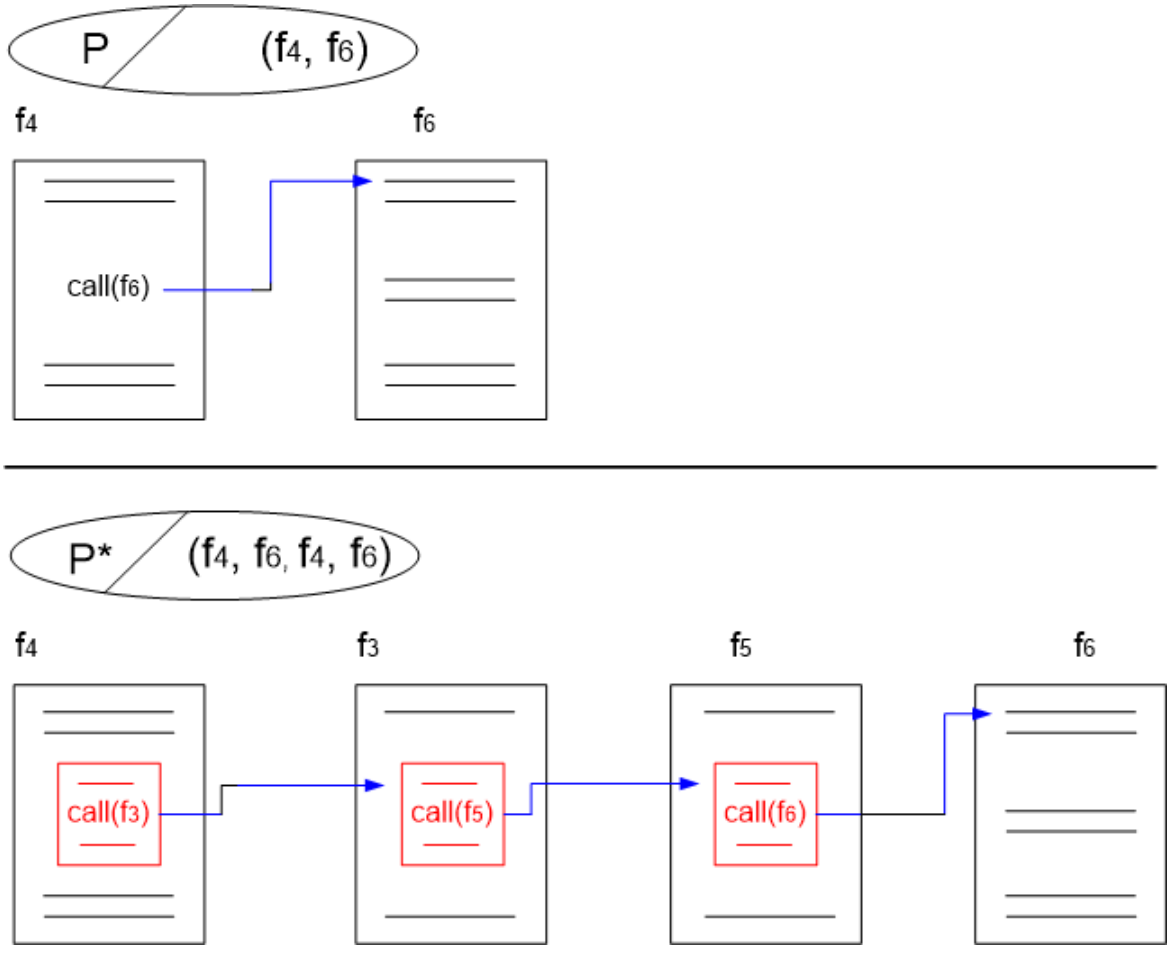
Σχήμα 3.7: Βασική ιδέα ενσωμάτωσης του υδατογραφήματος στον κώδικα σύμφωνα με το WaterRPG μοντέλο.

Συγκεκριμένα, αυτό που συμβαίνει είναι να αντιστοιχίζονται όλοι οι κόμβοι του αναγώγιμου μεταθετικού γραφήματος με συναρτήσεις του κώδικα οι οποίες ενεργοποιούνται δοθείσας μίας ακολουθίας εισόδου I_{key} και όλες οι ακμές του με εκείνες τις κλήσεις συναρτήσεων του κώδικα οι οποίες εκτελούνται με την ακολουθία αυτή. Επί της ουσίας δηλαδή, ο τρόπος με τον οποίο αντιστοιχίζονται οι κόμβοι με τις συναρτήσεις του κώδικα παραμένει ο ίδιος και αλλάζει ο τρόπος με τον οποίο αντιστοιχίζονται οι ακμές με τις κλήσεις των συναρτήσεων. Τώρα, όλες οι συναρτήσεις που ενεργοποιούνται, με την ακολουθία εισόδου κλειδί, αντιστοιχίζονται με κόμβους και όσες πραγματικές κλήσεις συναρτήσεων υπάρχουν εξ αρχής στον κώδικα και αντιστοιχούν σε ακμές στο γράφημα διατηρούνται και

γίνονται αναπόσπαστο τμήμα του υδατογραφήματος, ενώ οι υπόλοιπες που ενεργοποιούνται και δεν αντιστοιχούν σε κάποια από τις ακμές του γραφήματος αντικαθίστανται με άλλες τεχνητές κλήσεις (water calls) οι οποίες έχουν την ίδια λειτουργικότητα.

Σημειώνεται πως ενδέχεται να υπάρχουν είτε μία, είτε πολλαπλές εμφανίσεις της ίδιας κλήσης στο εσωτερικό μίας συνάρτησης οι οποίες εκτελούνται με την ακολουθία εισόδου I_{key} (π.χ., μία κλήση μέσα σε μία δομή επανάληψης και μία άλλη κλήση μετά τη δομή επανάληψης), η αντιστοίχηση ακμών με κλήσεις συναρτήσεων λαμβάνει υπόψη μονάχα την πρώτη εμφάνιση.

Η αντιστοίχηση κόμβων και συναρτήσεων είναι ένα προς ένα και γίνεται αρχίζοντας από τον κόμβο s του αναγώγιμου μεταθετικού γραφήματος και από την πρώτη συνάρτηση που καλείται στον κώδικα του προγράμματος (εκτός βέβαια της αρχικής) διασχίζοντας με τη σειρά και τους κόμβους του γραφήματος και τις συναρτήσεις που εκτελούνται. Επομένως το δυναμικό γράφημα κλήσεων που δημιουργείται με την ακολουθία εισόδου I_{key} είναι ισομορφικό με το ενσωματωμένο υδατογράφημα.



Σχήμα 3.8: Αντικατάσταση κλήσης συνάρτησης με αλληλουχία κλήσεων συναρτήσεων.

Τα υδατογραφήματα τα οποία ενσωματώνονται με το μοντέλο WaterRPG ανήκουν στα σήματα συντάκτη (authorship marks) και φέρουν την ιδιότητα του δυναμικού ίχνους εκτέλεσης (dynamic execution trace) καθώς για να αναγνωριστούν και να εξαχθούν από τον κώδικα αυτός πρέπει να εκτελεστεί μέσω μία συγκεκριμένης ακολουθίας εισόδου και ταυτόχρονα να γίνει καταγραφή του ίχνους εκτέλεσης για παράδειγμα μέσω ενός παρακολουθητή (profiler) ή μέσω ενός (debugger) προκειμένου να καταγραφούν όλες οι κλήσεις συναρτήσεων που εκτελέστηκαν. Σημειώνεται πως παρακάτω δίνεται αλγόριθμος ενσωμάτωσης και εξαγωγής υδατογραφήματος σε πηγαίο κώδικα λογισμικού, δεν παύει όμως να είναι καθ όλα εφικτή η ενσωμάτωση και απευθείας σε επίπεδο εκτελέσιμου κώδικα (π.χ., σε λογισμικά κωδικοποιημένα στη γλώσσα προγραμματισμού Java σε επίπεδο bytecode).

Το δυναμικό μοντέλο υδατογράφησης λογισμικού WaterRPG για να ενσωματώσει ένα αναγώγιμο μεταθετικό γράφημα, στον κώδικα διατηρώντας ταυτόχρονα την λειτουργικότητά του ανέπαφη, χρησιμοποιεί συστατικά του κώδικα (π.χ., συνθήκες ελέγχου) και κανόνες εκτέλεσης (π.χ., πότε οι συνθήκες ελέγχου θα αποτιμώνται ως αληθείς και πότε ως ψευδείς). Έτσι το υδατογράφημα κωδικοποιείται μέσα στον κώδικα του λογισμικού εκτελώντας τον κώδικά του αβίαστα (με την προϋπόθεση πως έχει δοθεί η ακολουθία εισόδου κλειδί) κατά την διάρκεια εκτέλεσης του υδατογραφημένου προγράμματος, πράττοντας ενέργειες οι οποίες είναι ζωτικής σημασίας για το λογισμικό (δίχως αυτές ο κώδικας δεν είναι ούτε ορθός, αλλά ούτε και λειτουργικός) και συνάμα χάρις αυτές ο εντοπισμός του υδατογραφήματος γίνεται εφικτός με την βοήθεια παρακολουθητών (profilers) [37] [44] [56].

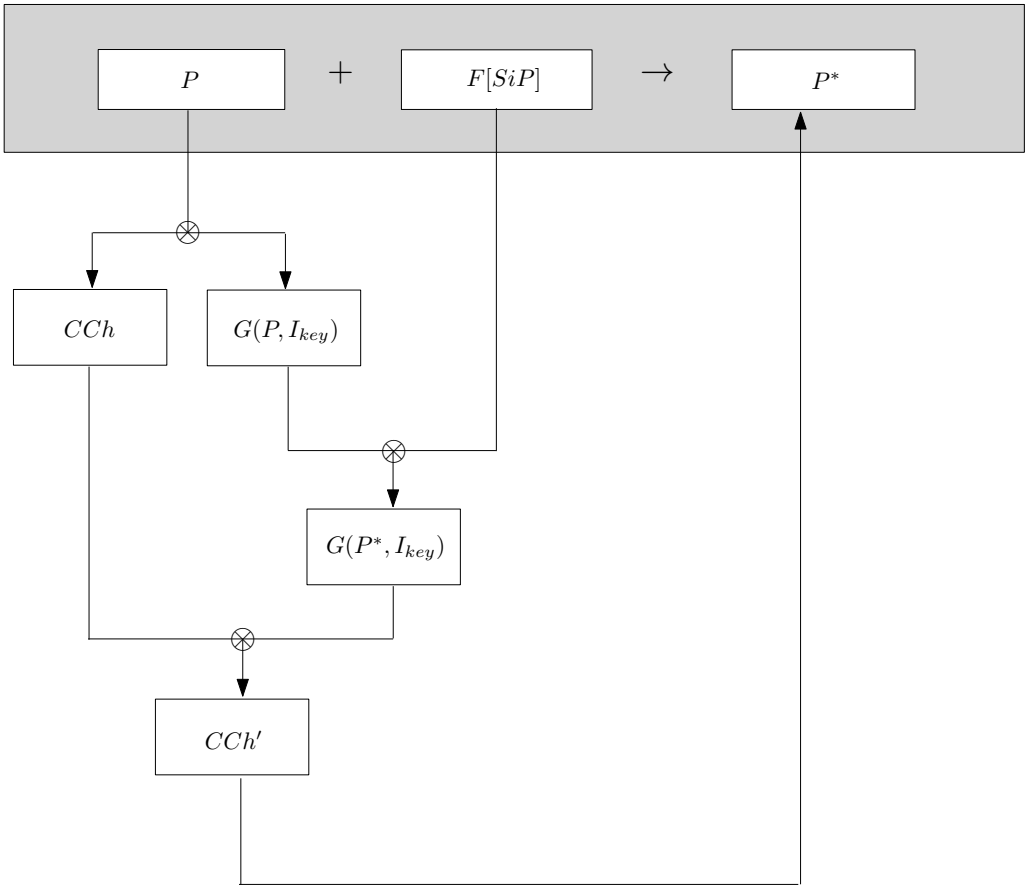
3.3 Διαδικασίες Ενσωμάτωσης και Εξαγωγής του Υδατογραφήματος

Εχοντας κατασκευάσει τη δομή υδατογράφησης λογισμικού, η οποία είναι το αναγώγιμο μεταθετικό γράφημα, μέσω του αλγορίθμου Chroni & Nikolopoulos, στόχος τώρα είναι αυτή η δομή να μπορέσει να κωδικοποιηθεί αποτελεσματικά μέσα σε κώδικα λογισμικού. Σε προηγούμενα κεφάλαια είχε παρουσιαστεί μία πρώτη προσέγγιση του μοντέλου υδατογράφησης, ονόματι WaterRPG, η οποία ενώ από τη μία πλευρά ήταν καινοτόμα, από την άλλη πλευρά είχε κάποια μειονεκτήματα τα οποία όπως και σε άλλα ήδη δημοσιευμένα σε επιστημονικές εργασίες μοντέλα δημιουργούσαν τρωτά και ιδιαίτερα εμφανή σημεία σήμανσης [45] [22] [64].

Με την πάροδο του χρόνου, ερευνώντας όλο και περισσότερο πάνω στον τομέα υδατογράφησης λογισμικού, λαμβάνοντας παράλληλα υπόψη διάφορες παραμέτρους όπως είναι η διαφάνεια, η πολυπλοκότητα ενσωμάτωσης και εξαγωγής, η αντίσταση σε πιθανές τροποποιήσεις, το επιπρόσθετο φορτίο σε χρόνο εκτέλεσης και χώρο αποθήκευσης, το μοντέλο WaterRPG τροποποιήθηκε πολλάκις μέχρι να καταλήξει στην τελευταία του έκδοση η οποία ξεπερνά τα εμπόδια τα οποία αντιμετωπίζουν πολλά μοντέλα ως τώρα. Με απλά λόγια δηλαδή, το υδατογράφημα κωδικοποιείται μέσα στον κώδικα χωρίς να χρησιμοποιούνται σημεία σήμανσης, χωρίς να είναι ευαίσθητο σε ένα διόλου μικρό εύρος από τροποποιήσεις σημασιολογικών δομών και βέβαια χωρίς την προσθήκη κώδικα ασυσχέτιστου με τον μη υδατογραφημένο.

Στην πραγματικότητα, αλλάζει ο τύπος (ιδιότητα) των υδατογραφημάτων που ενσωματώνονται στον κώδικα του λογισμικού, από στατικά και συγκεκριμένα υδατογραφήματα κώδικα σε δυναμικά υδατογραφήματα και συγκεκριμένα υδατογραφήματα δυναμικού ίχνους εκτέλεσης. Αυτά τα υδατογραφήματα κωδικοποιούνται στο δυναμικό γράφημα κλήσεων το οποίο δημιουργείται με την εκτέλεση του κώδικα και όχι στατικά στην ιεραρχία κλήσεων που κωδικοποιούνταν από το μοντέλο στα πρώιμα στάδιά του.

Επί της ουσίας, το μοντέλο αυτό τροποποιεί την ροή εκτέλεσης του κώδικα προσθέτοντας και αφαιρώντας κλήσεις συναρτήσεων πάντα με την βοήθεια συνθηκών ελέγχου οι οποίες αποτιμούν αδιαφανή κατηγορήματα της μορφής $Q^{P^?}$ προκειμένου να χειραγωγούνται κατά το δοκούν τα μονοπάτια εκτέλεσης διατηρώντας παράλληλα αναλλοίωτη την λειτουργικότητα του λογισμικού [10]. Στην πραγματικότητα αντικαθίστανται ένα υποσύνολο κλήσεων με άλλες κλήσεις οι οποίες χάρη στα αδιαφανή κατηγορήματα επιτελούν τις ίδιες λειτουργίες. Οι συναρτήσεις του κώδικα δηλαδή, είτε παίζουν τον ρόλο του ενδιάμεσου (μεταβιβάζουν την κλήση από τη μία συνάρτηση στην επόμενη) είτε παίζουν τον ρόλο του τελικού αποδέκτη (εκτέλεση αρχικού κώδικα συνάρτησης).



Σχήμα 3.9: Διαδικασία υδατογράφησης λογισμικού με το WaterRPG μοντέλο.

Τα αδιαφανή κατηγορήματα κατασκευάζονται με τη βοήθεια μεταβλητών υδατογράφησης (π.χ., ακεραίου τύπου), μία ή περισσότερες (απλές μεταβλητές οι οποίες χρησιμοποιούνται στην κωδικοποίηση του υδατογραφημάτος) οι οποίες, είτε μπορεί να υπάρχουν εξ αρχής μέσα στον κώδικα του λογισμικού, είτε μπορεί να προστεθούν στη συνέχεια. Οι εκφράσεις των κατηγορημάτων αυτών, αυτό που διαπράττουν στην ουσία είναι να ελέγχουν εύρη τιμών με σκοπό να επιτρέπουν ή όχι τη ροή εκτέλεσης ανά πάσα στιγμή να διέρχεται από συγκεκριμένα τμήματα κώδικα.

Κατά την διαδικασία ενσωμάτωσης στον κώδικα, προτού ακόμα ξεκινήσει η πιστή εφαρμογή του μοντέλου υδατογράφησης WaterRPG, ο κώδικας του λογισμικού εκτελείται με μία πληθώρα ακολουθιών εισόδου $I = \{I_1, I_2, \ldots, I_n\}$. Με δεδομένες αυτές τις εκτελέσεις, παράγεται εν συνεχεία για κάθε μία ακολουθία εισόδου ο δυναμικός πίνακας εκτέλεσής της $CCh = \{CCh_1, CCh_2, \ldots, CCh_n\}$ ο οποίος εμπεριέχει τα ζεύγη κλήσεων και τον χαρακτηρισμό τους, όπου στην περίπτωση του μη υδατογραφημένου προγράμματος όλες οι κλήσεις χαρακτηρίζονται, είτε ως πραγματικές μπροστά κλήσεις, είτε ως πραγματικές πίσω κλήσεις (ανάλογα με το αν η πρώτη εμφάνιση της καλούσας συνάρτησης είναι πριν ή μετά της πρώτης εμφάνισης της κληθείσας συνάρτησης) και ύστερα παράγεται το δυναμικό γράφημα κλήσεων $DCG = \{DCG_1, DCG_2, \ldots, DCG_n\}$ που αντιστοιχεί στον πίνακα αυτόν. Προφανώς, στη γενική περίπτωση των προγραμμάτων, το να βρεθεί ένα σύνολο ακολουθιών εισόδου το οποίο να καλύπτει όλα τα πιθανά ενδεχόμενα εκτέλεσης αποτελεί ένα άλυτο πρόβλημα δύσκολα διαχειρίσιμο.

Βασικός στόχος είναι να κατασκευαστεί ένα σύνολο ακολουθιών εισόδου το οποίο να καλύπτει ένα μεγάλο φάσμα τέτοιων εκτελέσεων. Επί της ουσίας να βρεθούν τόσες είσοδοι, οι οποίες να παράγουν όσο το δυνατόν διαφορετικά μεταξύ τους δυναμικά γραφήματα κλήσεων έτσι ώστε να υπάρχει ικανοποιητική ποικιλία επιλογής. Ύστερα, κάθε ένα από τα παραγόμενα γραφήματα αυτά, μελετάται προσεκτικά και συγκρίνεται με το υποψήφιο προς ενσωμάτωση αναγώγιμο μεταθετικό γράφημα προκειμένου να μπει ή όχι στο σύνολο των γραφημάτων / ακολουθιών εισόδου από το οποίο θα επιλεγεί το ένα και μοναδικό.

Συγκεκριμένα, αν υποτιθέσθω πως είναι $k, k \leq n$ τα γραφήματα τα οποία μοιάζουν σε μεγάλο βαθμό ή τουλάχιστον μοιάζουν περισσότερο από τα υπόλοιπα, επιλέγεται αυτό το οποίο συναρτήσει ορισμένων χαρακτηριστικών αξιολόγησης (π.χ., αντίκτυπο ενσωμάτωσης, τμηματική προστασία) θα επιφέρει τα βέλτιστα αποτελέσματα. Αφού λοιπόν έχει βρεθεί η ακολουθία εισόδου κλειδί I_{key} η οποία παράγει το δυναμικό γράφημα κλήσεων DCG_{key} το οποίο θα τροποποιηθεί με απώτερο σκοπό να γίνει ομοιομορφικό με το αναγώγιμο μεταθετικό γράφημα, ξεκινά και η εφαρμογή του αλγορίθμου υδατογράφησης λογισμικού.

Από την πλευρά της διαδικασίας εξαγωγής τα πράγματα είναι απλούστερα καθώς το μόνο που πρέπει να γίνει είναι να συνδεθεί ο υδατογραφημένος κώδικας με έναν παρακολουθητή (profiler) ή με έναν αποσφαλματωτή (debugger) και δοθείσας της ακολουθίας εισόδου I_{key} αυτός να εκτελεστεί καταγράφοντας τις κλήσεις συναρτήσεων που εκτελούνται καθ όλη της διάρκεια εκτέλεσης. Σημειωτέον, ότι δεν απαιτείται κάποια επιπλέον πληροφορία πέραν από τον υδατογραφημένο κώδικα και την ακολουθία εισόδου η οποία παίζει το ρόλο του κλειδιού.

3.4 Συνιστώσες του Μοντέλου WaterRPG

Το δυναμικό μοντέλο υδατογράφησης λογισμικού WaterRPG είναι αποτελούμενο από συνιστώσες οι οποίες τις συνδυάζει μέσω των αλγορίθμων ενσωμάτωσης και εξαγωγής με σκοπό να κωδικοποιεί και να αποκωδικοποιεί αναγώγιμα μεταθετικά γραφήματα μέσα και από κώδικες λογισμικών. Οι συνιστώσες αυτές επί της ουσίας χωρίζονται σε δύο κατηγορίες, οι συνιστώσες δεδομένων (data components) και οι συνιστώσες λειτουργίας (operational components). Στις συνιστώσες δεδομένων περιλαμβάνονται το δυναμικό γράφημα κλήσεων του μη υδατογραφημένου προγράμματος P, το αναγώγιμο μεταθετικό γράφημα $F[SiP]$ το οποίο είναι το υδατογράφημα που θα ενσωματωθεί στον κώδικα καθώς και το δυναμικό γράφημα κλήσεων του υδατογραφημένου προγράμματος P^*. Ενώ στις συνιστώσες λειτουργίας περιλαμβάνονται τα πρότυπα κλήσεων, οι συνθήκες ελέγχου και οι κανόνες εκτέλεσης.

3.4.1 Δυναμικό Γράφημα Κλήσεων

Το δυναμικό γράφημα κλήσεων του μη υδατογραφημένου προγράμματος P, το οποίο δημιουργείται δοθείσας της ακολουθίας εισόδου I_{key}, στη γενική περίπτωση είναι ένα τυχαίο κατευθυνόμενο γράφημα το οποίο περιέχει μία συνεκτική συνιστώσα. Το γεγονός ότι το γράφημα αυτό δεν μπορεί να έχει παραπάνω από μία συνεκτικές συνιστώσες προέρχεται από την γενική φιλοσοφία των προγραμμάτων όπου κάθε συνάρτηση ενεργοποιείται μόνο αν την καλέσει κάποια άλλη συνάρτηση.

Επομένως κάθε κόμβος του γραφήματος (εκτός βέβαια της αρχικής συνάρτησης) θα έχει τουλάχιστον μία εισερχόμενη ακμή. Από τη στιγμή όπου ο κόμβος αυτός δηλαδή υπάρχει στο γράφημα, σίγουρα θα υπάρχει τουλάχιστον μία εκ των συναρτήσεων που τον καλεί κατά τη διάρκεια εκτέλεσης. Ενώ το γράφημα κλήσεων του υδατογραφημένου προγράμματος P^* το οποίο δημιουργείται δοθείσας της ίδιας ακολουθίας I_{key} είναι ένα αναγώγιμο μεταθετικό γράφημα το οποίο προφανώς και περιέχει και αυτό μία και μόνο συνεκτική συνιστώσα. Τα δύο αυτά δυναμικά γραφήματα κλήσεων ενώ στη γενική των περιπτώσεων είναι διαφορετικά, επιτελούν ακριβώς την ίδια λειτουργία.

Στην περίπτωση του προγράμματος P το δυναμικό γράφημα κλήσεών του περιέχει μονάχα μία κατηγορία κλήσεων, τις πραγματικές κλήσεις (real calls), ενώ στην περίπτωση του προγράμματος P^* (με την προϋπόθεση βέβαια πως αυτά τα δύο γραφήματα δεν ταυτίζονταν εξ αρχής) το δυναμικό γράφημα κλήσεων περιέχει δύο κατηγορίες κλήσεων, τις πραγματικές κλήσεις (real calls) και τις τεχνητές κλήσεις (water calls). Οι εικονικές κλήσεις συναρτήσεων είναι εκείνες οι κλήσεις οι οποίες δεν υπήρχαν στο πρόγραμμα P, αλλά προστέθηκαν στο πρόγραμμα P^* προκειμένου να γίνει ισοφορμφικό με το αναγώγιμο μεταθετικό γράφημα που κωδικοποιείται μέσα στον κώδικά του.

Σημειώνεται επίσης, πως στο δυναμικό γράφημα κλήσεων του προγράμματος P^* το οποίο δημιουργείτο με την ακολουθία I_{key} εμφανίζονται και όλες εκείνες οι κλήσεις, έστω $call(f_i, f_j)$, που υπήρχαν στο P (καθώς και στο δυναμικό γράφημα του P που παράγονταν με την ίδια ακολουθία εισόδου) και δεν υπάρχουν πλέον στο P^* έχοντας αντικατασταθεί με

την αλληλουχία κλήσεων $call(f_i, f_{k_1}), call(f_i, f_{k_2}), \ldots, call(f_{k_n}, f_j)$. Οι κλήσεις οι οποίες παρεμβάλλονται μεταξύ των συναρτήσεων f_i και f_j στο μονοπάτι αυτό ονομάζονται ενδιάμεσες κλήσεις. Επί της ουσίας οι κλήσεις αυτές απλά μεταφέρουν την κλήση συνάρτησης που υπήρχε αρχικά στην f_i στον πραγματικό τελικό αποδέκτη της που ήταν η f_j.

3.4.2 Συνθήκες Ελέγχου

Οι συνθήκες ελέγχου παίζουν καθοριστικό ρόλο στην ενσωμάτωση του υδατογραφήματος στον κώδικα του λογισμικού, καθώς είναι εκείνες οι οποίες χειραγωγούν την ροή εκτέλεσης έτσι ώστε και το πρόγραμμα να διατηρείται λειτουργικό και το δυναμικό γράφημα κλήσεων που παράγεται από το υδατογραφημένο πρόγραμμα P^* να είναι ισομορφικό με το αναγώγιμο μεταθετικό γράφημα. Αυτές οι συνθήκες είτε μπορούν να προστεθούν σε ήδη υπάρχουσες δομές (π.χ., δομές ελέγχου, δομές επανάληψης) μέσα στον κώδικα, είτε μπορούν να προστεθούν σε νέες δομές οι οποίες προστίθενται για πρώτη φορά στο υδατογραφημένο πρόγραμμα και αποτιμούν εκφράσεις οι οποίες έχουν ως κύριο συστατικό τους τις μεταβλητές που χρησιμοποιούνται για την υδατογράφηση. Οι εκφράσεις αυτές των συνθηκών ελέγχων, αποτελούν αδιαφανή κατηγορήματα της μορφής $Q^{P?}$ καθώς πρώτον δεν είναι γνωστό εξ αρχής αν θα αποτιμηθούν ως αληθή ή ψευδή και δεύτερον καθώς η ροή εκτέλεσης εξελίσσεται και ενδεχομένως να διέλθει περισσότερες από μία φορές από μία τέτοια συνθήκη, τα αδιαφανή κατηγορήματά της μπορεί να αποτιμώνται κάποιες φορές ως αληθή και κάποιες άλλες ως ψευδή.

Στην πραγματικότητα οι συνθήκες ελέγχου παίζουν το ρόλο του διακόπτη, ο οποίος είτε επιτρέπει τη ροή εκτέλεσης να μπει στο κυρίως σώμα της συνάρτησης (στις εντολές που είχε πριν την υδατογράφηση) είτε μεταβιβάζει τη ροή εκτέλεσης σε άλλη συνάρτηση μέχρις ότου να φτάσει να εκτελεστεί η πραγματική συνάρτηση που θα έπρεπε να εκτελεστεί σύμφωνα με το αρχικό μη υδατογραφημένο πρόγραμμα. Στη γενική μορφή τέτοιες συνθήκες ελέγχου θα μπορούσαν να είναι οι ακόλουθες ή κάποιος συνδυασμός αυτών:

- **if** $(variable == value_1 \;||\; variable == value_2 \;||\; \ldots \;||\; variable == value_k)$ **then**
- **if** $(variable \geq value_i \;\&\&\; variable \leq value_j)$ **then**
- **if** $(variable \neq value_i \;\&\&\; variable \neq value_j)$ **then**
- **if** $(formula\ or\ type(variable))$ **then**

Στις συνθήκες ελέγχου οι οποίες αποτελούν τα αδιαφανή κατηγορήματα, εκτός από τις μεταβλητές υδατογράφησης μπορούν να λαμβάνουν χώρα και μεταβλητές του προγράμματος οι οποίες στο εσωτερικό των συνθηκών αυτών μπορούν να τροποποιούν τις τιμές τους (αν βέβαια δεν δημιουργούν προβλήματα λειτουργικότητας) και εν συνεχεία να επαναφέρουν τις τιμές τους, είτε όταν όταν η ροή εκτέλεσης εξέλθει από εκείνο το σημείο του κώδικα (block), είτε όταν όταν η ροή εκτέλεσης εξέλθει από το μονοπάτι εκτέλεσης όπου μεταβιβάζονται οι κλήσεις από τη μία συνάρτηση στην επόμενη.

3.4.3 Πρότυπα Κλήσεων

Θέλοντας να τροποποιηθεί η ροή εκτέλεσης του κώδικα, δοθείσας μίας ακολουθίας εισόδου κλειδί να παράγεται ένα δυναμικό γράφημα κλήσεων ισομορφικό με το αναγώγιμο μεταθετικό γράφημα το οποίο είναι κωδικοποιημένο μέσα στο λογισμικό, γίνεται εφαρμογή προτύπων κλήσεων. Τα πρότυπα αυτά, εστιάζουν σε κλήσεις συναρτήσεων και σε αναθέσεις τιμών σε μεταβλητές που χρησιμοποιούνται για την υδατογράφηση.

Συγκεκριμένα, τα πρότυπα χωρίζονται σε δύο κατηγορίες ανάλογα με το αν η κλήση συνάρτησης αντιστοιχεί σε μπροστά ή πίσω ακμή στο αναγώγιμο μεταθετικό γράφημα και τοποθετούνται αμφότερα στην καλούσα και στην κληθείσα συνάρτηση. Η διαφορά τους είναι οι τιμές που ανατίθενται στη μεταβλητή στην καλούσα συνάρτηση (π.χ., αν το πρότυπο περιέχει κλήση που αντιστοιχεί σε μπροστά ακμή στην μεταβλητή προστίθεται ένας σταθερός αριθμός $value_1$ πάντα πριν την εκτέλεση της κλήσης, ενώ αν το πρότυπο περιέχει κλήση που αντιστοιχεί σε πίσω ακμή στην μεταβλητή προστίθεται ένας σταθερός αριθμός $value_2$). Προφανώς σε ένα ελάχιστο μονοπάτι κλήσεων το οποίο έχει αντικαταστήσει μία πραγματική κλήση του μη υδατογραφημένου προγράμματος μπορεί (είναι πολύ πιθανόν) τα πρότυπα αυτά να εμφανίζονται και τα δύο σε αυτό.

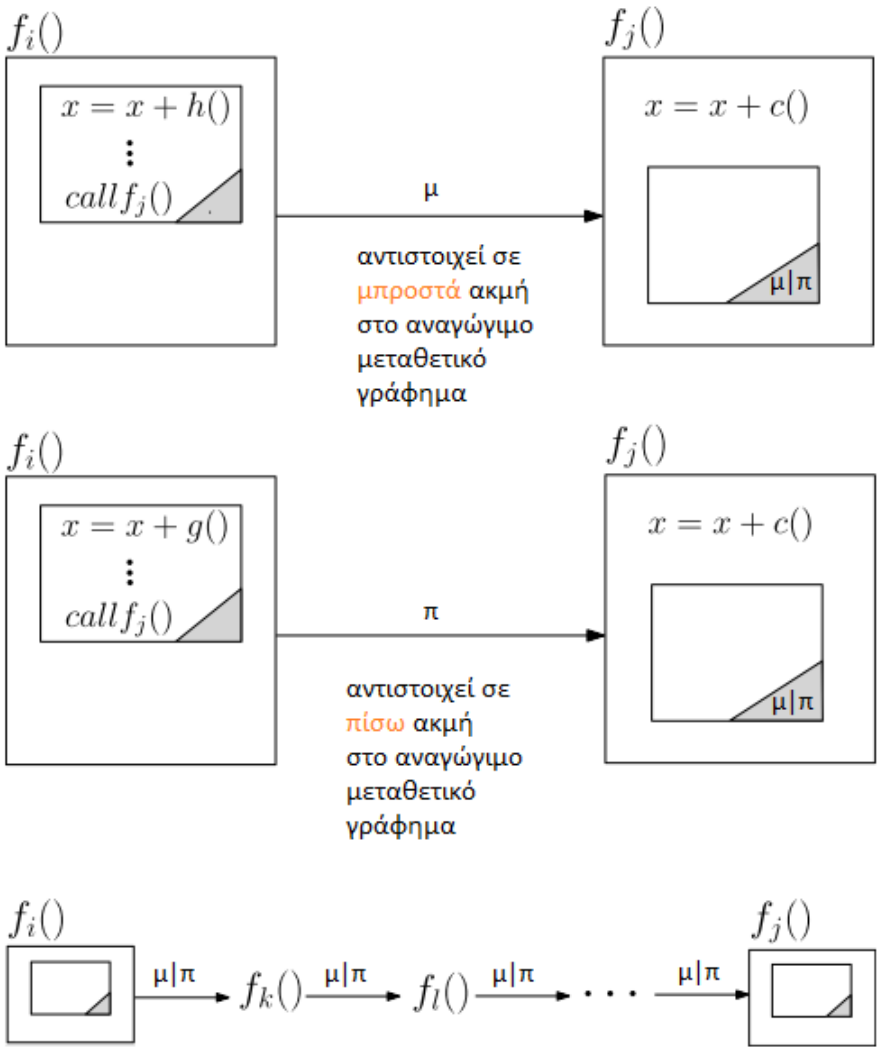
Στο σημείο αυτό αξίζει να σημειωθεί, πως στις περιπτώσεις κατά τις οποίες μία πραγματική κλήση του μη υδατογραφημένου προγράμματος αντιστοιχεί σε ακμή στο αναγώγιμο μεταθετικό γράφημα, διατηρείται στον κώδικα και δεν προστίθεται επιπλέον κλήση κατά την εφαρμογή του προτύπου. Το πρότυπο επί της ουσίας εφαρμόζεται χωρίς την προσθήκη νέας κλήσης απλά εκμεταλλεύεται την ήδη υπάρχουσα. Εν συντομία, τα πρότυπα κλήσεων ανάλογα με το αν η τρέχουσα κλήση κάθε φορά αντιστοιχεί σε μπροστά ή πίσω ακμή είναι τα ακόλουθα:

- Αν η κλήση συνάρτησης (f_i, f_j) αντιστοιχεί σε μπροστά ακμή στο αναγώγιμο μεταθετικό γράφημα, στην καλούσα συνάρτηση f_i αμέσως πριν την κλήση προστίθεται εντολή ανάθεσης στην μεταβλητή υδατογράφησης x η οποία είναι $x = x + h()$, όπου το $h()$ είναι μία σταθερά και στην κληθείσα συνάρτηση f_j ως πρώτη εντολή η ανάθεση $x = x + c()$, όπου το $c()$ είναι επίσης μία σταθερά.

- Αν η κλήση συνάρτησης (f_i, f_j) αντιστοιχεί σε πίσω ακμή στο αναγώγιμο μεταθετικό γράφημα, στην καλούσα συνάρτηση f_i αμέσως πριν την κλήση προστίθεται εντολή ανάθεσης στην μεταβλητή υδατογράφησης x η οποία είναι $x = x + g()$, όπου το $g()$ είναι μία σταθερά και στην κληθείσα συνάρτηση f_j ως πρώτη εντολή η ανάθεση $x = x + c()$, όπου το $c()$ είναι επίσης μία σταθερά.

Επαναδιατυπώνεται για λόγους συνοχής πως κατά την εφαρμογή ενός προτύπου κλήσεων, αν υπάρχει εξ αρχής είτε η κλήση συνάρτησης, είτε η ανάθεση τιμής στην μεταβλητή υδατογράφησης (προφανώς στην κληθείσα συνάρτηση καθώς στην καλούσα δεν δίνατε να υπάρχει διότι κάθε κλήση επεξεργάζεται μία και μόνο φορά) δεν προστίθεται νέα. Το μόνο που προστίθεται είναι και στην καλούσα και στην κληθείσα συνάρτηση στις συνθήκες ελέγχου που υπεισέρχονται στο πρότυπο αυτό, νέα αδιαφανή κατηγορήματα τα οποία αντιστοιχούν

στις τιμές της μεταβλητής οι οποίες λαμβάνονται όταν η ροή εκτέλεσης φτάσει να εκτελέσει τις εντολές του προτύπου αυτού.



Σχήμα 3.10: Πρότυπα κλήσεων συναρτήσεων.

Οι κλήσεις συναρτήσεων και κατ επέκταση οι συναρτήσεις του κώδικα οι οποίες βρίσκονται μεταξύ των συναρτήσεων f_i και f_j στο προηγούμενο σχήμα αποτελούν τις ενδιάμεσες συναρτήσεις οι οποίες παρεμβάλλονται για λόγους κατασκευής του δυναμικού γραφήματος κλήσεων. Ουσιαστικά, η κλήση η οποία υπήρχε στο μη υδατογραφημένο πρόγραμμα και θα έπρεπε να πραγματοποιηθεί ήταν η (f_i, f_j), αλλά πλέον στο υδατογραφημένο αντί αυτής, υπάρχει ένα μονοπάτι κλήσεων $(f_i, f_k), (f_k, f_l), \ldots, (f_m, f_j)$ το οποίο μεταβιβάζει την κλήση στην πραγματική κληθείσα συνάρτηση f_j.

3.4.4 Κανόνες Εκτέλεσης

Ολοκληρώνοντας με τις συνιστώσες του μοντέλου υδατογράφησης λογισμικού WaterRPG παρουσιάζονται και οι κανόνες εκτέλεσης των αδιαφανών κατηγορημάτων τα οποία αποτελούν τις εκφράσεις των συνθηκών ελέγχου. Οι κανόνες αυτοί ουσιαστικά ρυθμίζουν το πότε τα αδιαφανή κατηγορήματα $Q^{P^?}$ θα αποτιμώνται ως αληθή και πότε ως ψευδή προκειμένου και η λειτουργικότητα του κώδικα να διατηρείται και η ροή εκτέλεσης να παράγει ένα δυναμικό γράφημα κλήσεων ισομορφικό με το αναγωγιμό μεταθετικό γράφημα. Οι κανόνες αυτοί είναι οι εξής:

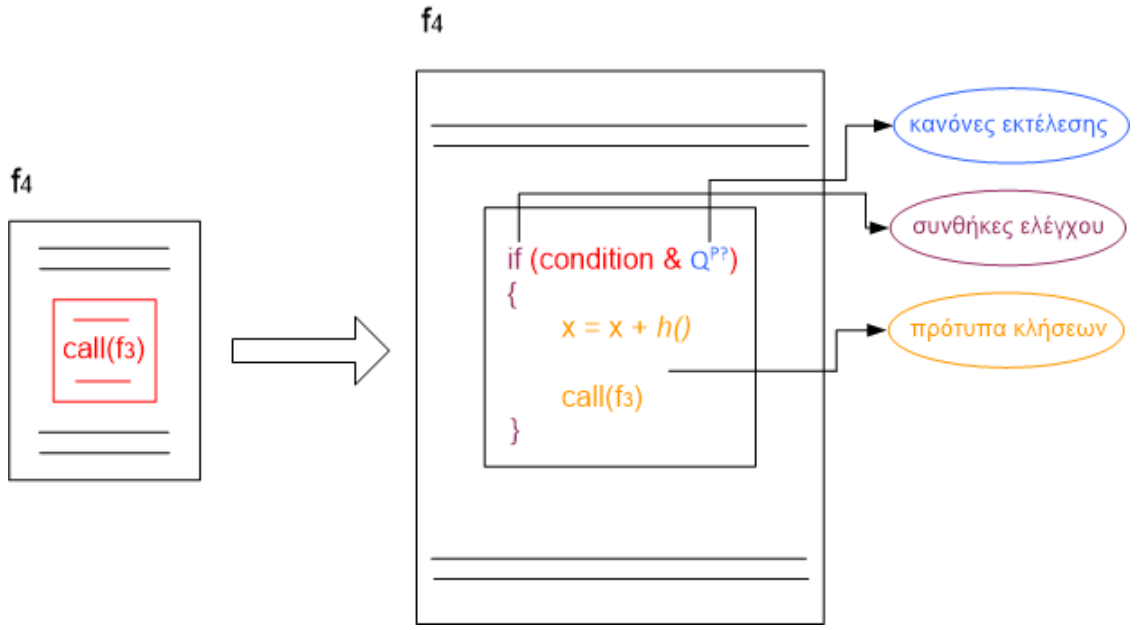
- Αν η κλήση (f_i, f_j) αποτελεί ενδιάμεση κλήση σε κάποιο από τα ελάχιστα μονοπάτια εκτέλεσης, τότε τα αδιαφανή κατηγορήματα στην καλούσα συνάρτηση f_i τα οποία αποτελούν την έκφραση της συνθήκης ελέγχου η οποία εμπεριέχει την ανάθεση στη μεταβλητή υδατογράφησης $x = x + h()$ ή $x = x + g()$ και την κλήση αυτή αποτιμώνται ως αληθή, τα αδιαφανή κατηγορήματα στην κλειθήσα συνάρτηση f_j τα οποία αποτελούν την έκφραση της συνθήκης ελέγχου η οποία εμπεριέχει την ανάθεση στη μεταβλητή υδατογράφησης $x = x + c()$ αποτιμώνται ως αληθή και τα αδιαφανή κατηγορήματα στην κλειθήσα συνάρτηση f_j τα οποία αποτελούν την έκφραση της συνθήκης ελέγχου η οποία εμπεριέχει τον αρχικό κώδικα της συνάρτηση αποτιμώνται ως ψευδή.

- Αν η κλήση (f_i, f_j) αποτελεί τελική κλήση σε κάποιο από τα τα ελάχιστα μονοπάτια εκτέλεσης, τότε τα αδιαφανή κατηγορήματα στην καλούσα συνάρτηση f_i τα οποία αποτελούν την έκφραση της συνθήκης ελέγχου η οποία εμπεριέχει την ανάθεση στη μεταβλητή υδατογράφησης $x = x + h()$ ή $x = x + g()$ και την κλήση αυτή αποτιμώνται ως αληθή, τα αδιαφανή κατηγορήματα στην κλειθήσα συνάρτηση f_j τα οποία αποτελούν την έκφραση της συνθήκης ελέγχου η οποία εμπεριέχει την ανάθεση στη μεταβλητή υδατογράφησης $x = x + c()$ αποτιμώνται ως αληθή και τα αδιαφανή κατηγορήματα στην κλειθήσα συνάρτηση f_j τα οποία αποτελούν την έκφραση της συνθήκης ελέγχου η οποία εμπεριέχει τον αρχικό κώδικα της συνάρτηση αποτιμώνται ως αληθή.

Στο σημείο αυτό διευκρινίζεται πως αν μία κλήση συνάρτησης βρίσκεται εντός κάποιας δομής επανάληψης οι αυξήσεις στις μεταβλητές υδατογράφησης $x = x + h()$, $x = x + g()$ και $x = x + c()$ πραγματοποιούνται πραγματοποιούνται κατά την πρώτη επανάληψη της δομής προκειμένου και να αποφευχθούν τυχόν εκρήξεις στις τιμές που θα λαμβάνουν οι μεταβλητές αυτές αλλά και να αποφευχθεί σπατάλη χρόνου προγραμματισμού για να καθοριστούν όλες οι ενδιάμεσες τιμές της μεταβλητής υδατογράφησης σε κάθε μία από τις επαναλήψεις της δομής. Το γεγονός αυτό έχει ιδιαίτερη σημασία σε καταστάσεις όπου δεν είναι γνωστό εξ αρχής το πλήθος των επαναλήψεων που θα συμβούν.

Επιπλέον, αν με κάποια άλλη ακολουθία εισόδου διαφορετική από την ακολουθία κλειδί μία συνάρτηση η οποία δεν αποτελεί τμήμα του υδατογραφήματος καλεί μία αντιστοιχισμένη συνάρτηση με κόμβο του αναγώγιμου μεταθετικού γραφήματος, τότε πριν πραγματοποιηθεί η κλήση αυτή πρέπει στο σώμα της καλούσας συνάρτησης να εκτελεστεί μία ανάθεση στη μεταβλητή υδατογράφησης η οποία να της δίνει τέτοια τιμή, η οποία να στις συνθήκες

ελέγχου της κληθείσας συνάρτησης να αποτιμάται ως αληθής προκειμένου ο κώδικας που υπήρχε στην μη υδατογραφημένη έκδοση της κληθείσας συνάρτησης να ξεκλειδώσει και να εκτελεστεί. Για παράδειγμα αν η συνθήκη ελέγχου της κλειθήσας συνάρτησης περιέχει ένα αδιαφανής κατηγόρημα το οποίο ελέγχει αν η μεταβλητή υδατογράφησης κυμαίνεται στο εύρος από m έως n, τότε η μεταβλητή στην καλούσα συνάρτηση πρέπει να λάβει τιμή η οποία να βρίσκεται εντός του εύρους αυτού.



Σχήμα 3.11: Συνδυασμός συνθήκης ελέγχου, πρότυπο κλήσης και κανόνων εκτέλεσης.

3.5 Αλγόριθμος Υδατογράφησης Λογισμικού

Ο αλγόριθμος υδατογράφησης λογισμικού σύμφωνα με το δυναμικό μοντέλο WaterRPG όπως και ο αλγόριθμος κατασκευής Chroni & Nikolopoulos της δομής υδατογράφησης χωρίζεται και αυτός σε επιμέρους φάσεις, με την διαφορά πως σε αυτήν την περίπτωση οι φάσεις είναι δύο, η φάση της ενσωμάτωσης και η φάση της εξαγωγής και όχι τέσσερις. Η φάση της ενσωμάτωσης του υδατογραφήματος ουσιαστικά προσθέτει ή αλλιώς κωδικοποιεί μη ορατή πληροφορία μέσα στον κώδικα, η οποία στην φάση της εξαγωγής αποσπάται ή αλλιώς αποκωδικοποιείται προκειμένου να αποδειχθεί ο πραγματικός ιδιοκτήτης των πνευματικών δικαιωμάτων.

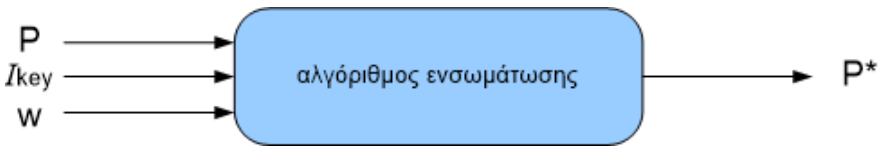
Προαπαιτούμενο και της φάσης της ενσωμάτωσης και της φάσης της εξαγωγής είναι το υδατογράφημα προκειμένου να κωδικοποιηθεί στο και να αποκωδικοποιηθεί από το λογισμικό μέσω των αλγορίθμων είναι να απαιτεί πολυωνιμικό χρόνο όπου στην καλύτερη περίπτωση ο χρόνος αυτός να είναι γραμμικός και στην περίπτωση κατά την οποία δεν έχει

τροποποιηθεί στο ελάχιστο ο υδατογραφημένος κώδικας, το υδατογράφημα να μπορεί πάντα να εξαχθεί από αυτόν. Ενώ θεμέλιος λίθος κάθε μοντέλου υδατογράφησης λογισμικού είναι η μη υδατογραφημένη έκδοση του κώδικα P καθώς και η υδατογραφημένη P^* να παράγουν ακριβώς το ίδιο αποτέλεσμα για κάθε ακολουθία εισόδου $O(P, I) \equiv O(P^*, I)$ απαιτώντας παραπλήσιο χρονικό διάστημα $time(P, I) \simeq time(P^*, I)$ και παραπλήσιο χωρικό $space(P, I) \simeq space(P^*, I)$.

3.5.1 Φάση Ενσωμάτωσης Υδατογραφήματος

Η φάση της ενσωμάτωσης του μοντέλου WaterRPG κωδικοποιεί πληροφορία μέσα στον κώδικα του λογισμικού η οποία διατηρεί αναλλοίωτη την λειτουργικότητά του. Η πληροφορία αυτή είναι ένας ακέραιος αριθμός ο οποίος έχει μετατραπεί σε αναγώγιμο μεταθετικό γράφημα μέσω του αλγορίθμου Chroni & Nikolopoulos και αποτελεί το υδατογράφημα το οποίο με την κωδικοποίησή του μέσα στο λογισμικό, προστατεύει τα πνευματικά δικαιώματα του νόμιμου κατόχου της πνευματικής ιδιοκτησίας. Στόχος είναι δοθείσας της ακολουθίας κλειδί να παράγεται δυναμικό γράφημα κλήσεων ισομορφικό με το αναγώγιμο μεταθετικό γράφημα.

Αξίζει να σημειωθεί πως με αυτό το μοντέλο υδατογράφησης λογισμικού μπορούν να ενσωματωθούν και οποιαδήποτε γραφήματα, πέραν από τα κλασσικά αναγώγιμα μεταθετικά γραφήματα. Ο λόγος για τον οποίον το μοντέλο WaterRPG ενσωματώνει αυτή την κλάση γραφημάτων είναι πως τα γραφήματα αυτά μπορούν και μιμούνται σε καλό βαθμό την ροή εκτέλεσης του κώδικα με αποτέλεσμα να μην δίνουν εύκολα στόχο σε κακόβουλους χρήστες.



Σχήμα 3.12: Ενσωμάτωση υδατογραφήματος.

Υπάρχει βέβαια και η εκδοχή (η οποία υποστηρίζεται από ορισμένους ερευνητές) που αναφέρει πως η ενσωμάτωση τέτοιων γραφημάτων σε λογισμικά δίνει περισσότερο στόχο από ότι θα έδινε η ενσωμάτωση τυχαίων γραφημάτων. Το γεγονός αυτό το στηρίζουν στο ότι τα αναγώγιμα μεταθετικά γραφήματα χαρακτηρίζονται από αυστηρές ιδιότητες οι οποίες σπανίως εμφανίζονται σε κώδικες λογισμικών (π.χ., κάθε κόμβος έχει ακριβώς 2 εξερχόμενες ακμές). Μία μέση λύση θα ήταν γραφήματα τα οποία χαρακτηρίζονταν από τις αυστηρές ιδιότητες των αναγώγιμων μεταθετικών γραφημάτων αλλά όχι σε όλη τους τη δομή.

Παρακάτω παρουσιάζεται βήμα προς βήμα ο αλγόριθμος της φάσης της ενσωμάτωσης ακολουθούμενος από παρατηρήσεις οι οποίες διευκρινίζουν ορισμένα από τα βήματά του.

Είσοδος: κώδικας P, ακολουθία εισόδου I_{key}, υδατογράφημα $F[SiP]$
Εξοδος: κώδικας P^*

---

1. Εκτέλεση του P με I_{key} και παραγωγή του δισδιάστατου πίνακα CCh;
2. Κατασκευή από τον CCh του δυναμικού γραφήματος κλήσεων $DCG[P]$;
3. Κατασκευή του CCh' από τον CCh:
Για κάθε εγγραφή $e = (f_i, f_j, *)$ στον CCh που αντιστοιχεί σε ακμή στο $DCG[P]$

3.1 Πραγματοποιείται έλεγχος ταύτισης στο $F[SiP]$;

Τέλος επανάληψης
4. Συμπερίληψη όσων κλήσεων δεν εμφανίστηκαν ως ακμές του $F[SiP]$:
Για κάθε ακμή στο $F[SiP]$ που δεν αντιστοιχίζεται σε κάποια εγγραφή στον CCh'

4.1 Προσθήκη κλήσης στον CCh';

Τέλος επανάληψης
5. Τροποποίηση κώδικα:
Για κάθε εγγραφή $e = (f_i, f_j, *)$ του CCh'

5.1 Εφαρμογή προτύπων κλήσεων, συνθηκών ελέγχου και κανόνων εκτέλεσης;

Τέλος επανάληψης
6. Ελεγχος λοιπών συναρτήσεων:
Για κάθε συνάρτηση που δεν αντιστοιχεί σε κόμβου του $F[SiP]$

6.1 Σάρωση κλήσεων συναρτήσεων των υπολοίπων συναρτήσεων του κώδικα;

Τέλος επανάληψης
7. **Επιστροφή** του P^*;

Αλγόριθμος 5: Πρώτη φάση ενσωμάτωσης *Encode_RPG_to_code*

Παρατήρηση Βήματος 3.1: Στο Βήμα 3.1 του αλγορίθμου ενσωμάτωσης του υδατογραφήματος στον κώδικα, γίνεται έλεγχος σε κάθε μία από τις εγγραφές $e = (f_i, f_j, *)$ του πίνακα CCh για το ενδεχόμενο της κλήσης (f_i, f_j) να αντιστοιχεί ως ακμή στο γράφημα $F[SiP]$. Αν η e δεν αντιστοιχεί σε ακμή τότε στον CCh στο σημείο της τρέχουσας εγγραφής η εγγραφή αυτή διαγράφεται και στη θέση της προστίθεται η ελάχιστη αλληλουχία κλήσεων που αντιστοιχεί στο ελάχιστο μονοπάτι μεταξύ των κόμβων που αντιστοιχούν στις συναρτήσεις f_i και f_j καθώς και ο χαρακτηρισμός ανάλογα με το αν η κάθε κλήση που προστέθηκε ως εγγραφή αντιστοιχεί σε μπροστά ή πίσω ακμή στο $F[SiP]$ (forward / backward) και με τον αν υπήρχε εξ αρχής ή προστέθηκε μετέπειτα (real / water) στον κώδικα.

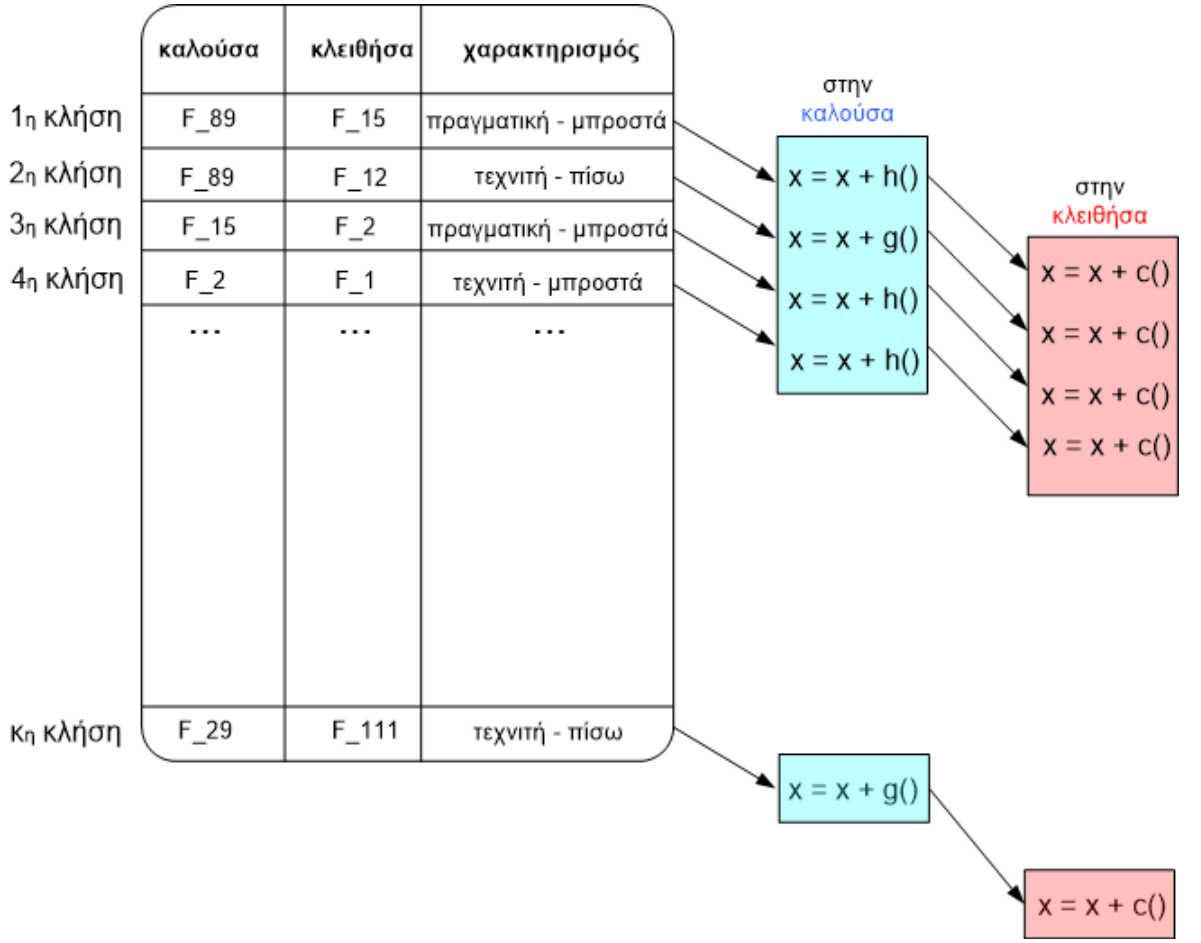
Παρατήρηση Βήματος 4.1: Στο Βήμα 4.1 του αλγορίθμου ενσωμάτωσης του υδατογραφήματος στον κώδικα, ελέγχονται ποιες από τις ακμές του γραφήματος $F[SiP]$ δεν έχουν προστεθεί ως κλήσεις συναρτήσεων. Για κάθε μία από αυτές πρέπει να βρεθεί η πρώτη εγγραφή στον CCh' που έχει στη θέση της κλειθήσας συνάρτησης f_j τη συνάρτηση που αντιστοιχεί στον κόμβο του $F[SiP]$ που εισέρχεται η ακμή αυτή. Αφού εντοπιστεί η εγγραφή αυτή, γίνεται ολίσθηση όλων των εγγραφών από το σημείο εκείνο και μετέπειτα μία θέση προς τα κάτω και στη συνέχεια στην κενή θέση που δημιουργήθηκε ακριβώς από κάτω πραγματοποιείται προσθήκη της εγγραφής που αντιστοιχεί στην προστιθέμενη κλήση με χαρακτηρισμό ως water backward.

Παρατήρηση Βήματος 5.1: Στο Βήμα 5.1 του αλγορίθμου ενσωμάτωσης του υδατογραφήματος στον κώδικα, περικλείονται εντός συνθηκών ελέγχου η νέα προσθήκη ανάθεσης τιμής στη μεταβλητή υδατογράφησης $x = x + h()$ ή $x = x + g()$ ανάλογα με το αν η κλήση αυτή αντιστοιχεί σε μπροστά ή πίσω ακμή στο $F[SiP]$ και η κλήση συνάρτησης. Η νέα συνθήκη ελέγχου έχει ως έκφραση αποτίμησης, τέτοια έτσι ώστε να αποτιμάται ως αληθή αν η μεταβλητή υδατογράφησης έχει την τρέχουσα τιμή. Επίσης, στην αρχή της κλειθήσας συνάρτησης περικλείεται εντός συνθήκης ελέγχου η ανάθεση $x = x + c()$ στη μεταβλητή υδατογράφησης η οποία αποτιμάται ως αληθή κάθε στιγμή που η συνάρτηση αυτή παίζει τον ρόλο της κλειθήσας συνάρτησης, εκτός των φορών που η κλήση βρίσκεται σε δομή επανάληψης όπου μονάχα την πρώτη φορά η συνθήκη αυτή γίνεται ψευδή.

Παρατήρηση Βήματος 6.1: Στο Βήμα 6.1 του αλγορίθμου ενσωμάτωσης του υδατογραφήματος στον κώδικα, προκειμένου ο υπόλοιπος κώδικας να παραμείνει λειτουργικός σε κάθε συνάρτηση που δεν σχετίζεται άμεσα με το υδατογράφημα, αλλά έμμεσα, ελέγχεται αν στο σώμα της υπάρχει κάποια κλήση σε συνάρτηση που σχετίζεται άμεσα με το υδατογράφημα. Αν υπάρχει τέτοια κλήση συνάρτησης, έστω (f_{other}, f_i), τότε πριν την εκτέλεσή της η μεταβλητή υδατογράφησης x λαμβάνει τέτοια τιμή η οποία να επιτρέπει στη ροή εκτέλεσης του κώδικα, δοθείσας μίας διαφορετικής ακολουθίας εισόδου από αυτήν που αποτελεί το κλειδί, να εισέλθει μέσα στον αρχικό κώδικα της συνάρτησης. Όμως η τιμή αυτή δεν είναι μία οποιαδήποτε, αλλά εκείνη η οποία η μεταβλητή θα λάμβανε αν με την ακολουθία εισόδου κλειδί η ροής εκτέλεσης έφτανε στην συνάρτηση f_i και έπρεπε να εκτελέσει τον αρχικό της κώδικα. Η τιμή αυτή υπολογίζεται από τον πίνακα CCh βρίσκοντας, είτε την πρώτη εμφάνιση ως πραγματική κλήση της f_i, είτε την κλήση της συνάρτησης f_i η οποία αποτελεί τελευταία κλήση σε κάποιο ελάχιστο μονοπάτι.

Ο πίνακας ο οποίος περιέχει με τη σειρά με την οποία εκτελέστηκαν τις κλήσεις των συναρτήσεων ονομάζεται CCh και είναι ιδιαίτερα χρήσιμος καθώς από αυτόν μπορούν να εξαχθούν οι τιμές που λαμβάνει η μεταβλητή ή οι μεταβλητές υδατογράφησης σε κάθε στιγμή κατά την διάρκεια εκτέλεσης του κώδικα. Το γεγονός αυτό, είναι άκρως σημαντικό στις περιπτώσεις όπου υπάρχει κλήση συνάρτησης από συνάρτηση μη αντιστοιχισμένη με κόμβο του υδατογραφήματος σε συνάρτηση αντιστοιχισμένη με κόμβο του υδατογραφήγματος,

διότι είναι απαραίτητο η μεταβλητή υδατογράφησης να λάβει τέτοια τιμή η οποία να επιτρέπει την ροή εκτέλεσης να περάσει και να εκτελέσει τον πραγματικό κώδικα της κληθείσας συνάρτησης. Επίσης, από τον πίνακα αυτόν παράγεται και το δυναμικό γράφημα κλήσεων, αντιστοιχίζοντας κάθε συνάρτηση με έναν κόμβο (μία φορά και για την πρώτη εμφάνιση) και κάθε κλήση συνάρτησης με μία ακμή (μία φορά και για την πρώτη εμφάνιση).

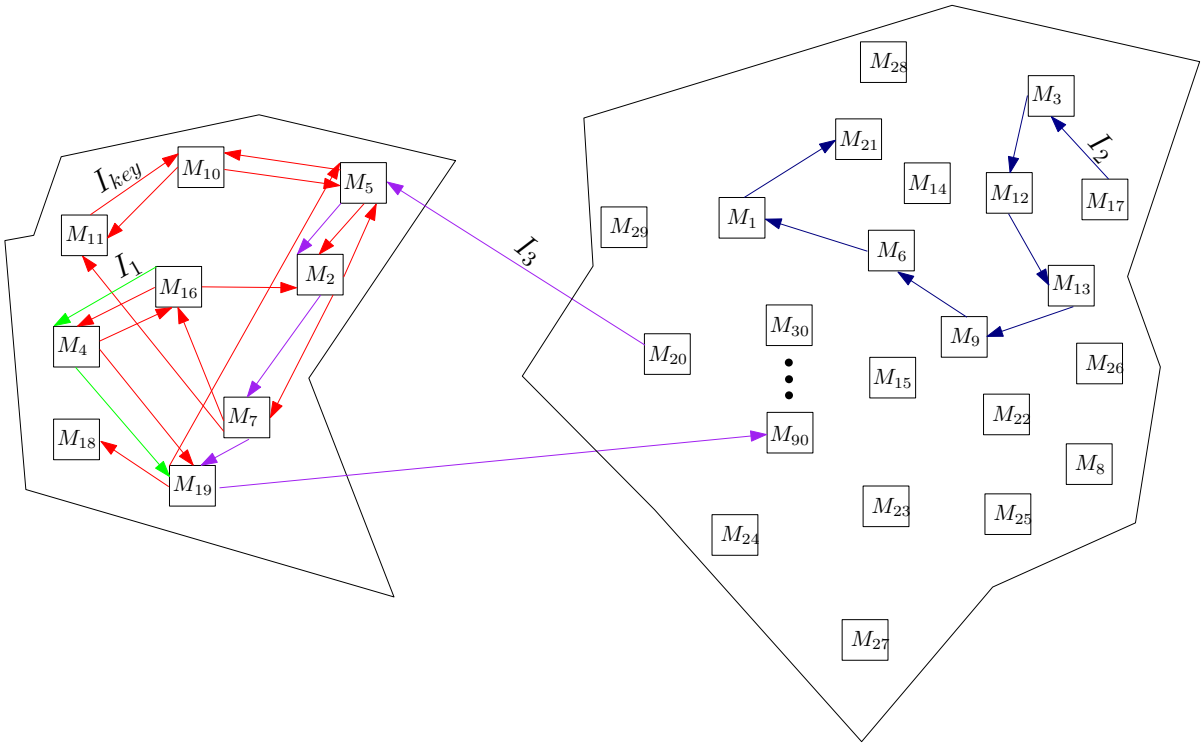


Σχήμα 3.13: Πίνακας ο οποίος εμπεριέχει όλες τις κλήσεις συναρτήσεων που εκτελούνται κατά τη διάρκεια εκτέλεσης.

Στο σημείο αυτό αξίζει να τονιστεί πως τα ενδεχόμενα που υπάρχουν σχετικά με το πως θα κληθούν οι συναρτήσεις του υδατογραφημένου κώδικα P^*, ύστερα από την υδατογράφησή του με το δυναμικό μοντέλο υδατογράφησης λογισμικού $WaterRpg$ είναι 4 στον αριθμό τους. Το κάθε ένα από τα ενδεχόμενα αυτά θέτει τους δικούς του τρόπους - κανόνες σύμφωνα με τους οποίους η μεταβλητή υδατογράφησης θα λαμβάνει τιμές κατά την διάρκεια εκτέλεσης του κώδικα.

Συγκεκριμένα δοθείσας της μυστικής ακολουθίας εισόδου I_{key} εκτελούνται όλες οι συναρτήσεις οι οποίες έχουν αντιστοιχηθεί με κόμβους και όλες οι κλήσεις αυτών οι οποίες αντιστοιχούν σε ακμές του αναγώγιμου μεταθετικού γραφήματος και με τέτοιον τρόπο έτσι

ώστε ως τελικό αποτέλεσμα να κατασκευαστεί ένα δυναμικό γράφημα κλήσεων ισομορφικό με αυτό. Στην περίπτωση αυτή η μεταβλητή υδατογράφησης λαμβάνει τιμές καθ όλη την διάρκεια εκτέλεσης του κώδικα, εντός των προτύπων κλήσεων, σύμφωνα με την ακολουθία αριθμών που έχει οριστεί ρητά στο μοντέλο WaterRPG.



Σχήμα 3.14: Πιθανές ροές εκτέλεσης.

Επίσης αν δοθείσας μίας άλλης ακολουθίας, έστω I_1, εκτελείται ένα υποσύνολο των συναρτήσεων που έχουν αντιστοιχιστεί με κόμβους του υδατογραφήματος και το υποσύνολο αυτό δεν εμπεριέχει ως αρχική εκτελεσθείσα συνάρτηση, την συνάρτηση που έχει αντιστοιχηθεί με τον κόμβο s, του υδατογραφήματος, τότε η μεταβλητή υδατογράφησης δεν έχει λάβει καμία τιμή (είναι αρχικοποιημένη σε μία τιμή έστω k εξ αρχής, οπότε και παραμένει σε αυτή). Οι υπόλοιπες συναρτήσεις που καλεί επομένως, κάνοντας έλεγχο στο σώμα τους μπορούν και γνωρίζουν πως η καλούσα συνάρτηση προέρχεται από συνάρτηση αντιστοιχισμένη με κόμβο του αναγώγιμου μεταθετικού γραφήματος αλλά, η ακολουθία εισόδου δεν είναι η I_{key}.

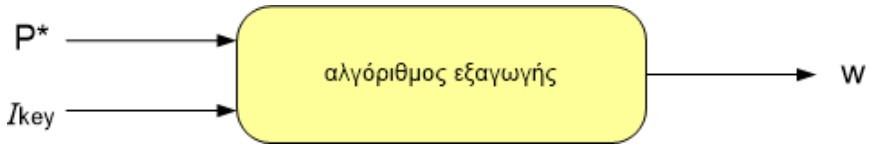
Αν αντιθέτως, η πρώτη εκτελεσθείσα συνάρτηση είναι αυτή που έχει αντιστοιχηθεί με τον κόμβο s, του γραφήματος $F[\pi^*]$, τότε η μεταβλητή της υδατογράφησης λαμβάνει κανονικά τις τιμές της σύμφωνα με την ακολουθία που έχει οριστεί στο μοντέλο WaterRPG χωρίς κανένα πρόβλημα δυσλειτουργίας, με τη διαφορά πως δεν θα εκτελεστούν όλες οι συναρτήσεις για να κατασκευαστεί δυναμικό γράφημα κλήσεων ισομορφικό με το ενσωματωμένο $F[SiP]$.

Επιπλέον άλλη μια ακολουθία εισόδου, έστω I_2, θα μπορούσε να εκτελεί από τον κώδικα του λογισμικού μονάχα συναρτήσεις οι οποίες δεν είναι αντιστοιχισμένες με κόμβους του $F[SiP]$. Το γεγονός αυτό, αποτελεί την πιο απλή και ανώδυνη περίπτωση καθώς δεν δημιουργεί πρόβλημα στην λειτουργικότητα του κώδικα, επομένως δεν είναι αναγκαία και καμία ανάθεση στην μεταβλητή υδατογράφησης, καθώς επίσης δεν είναι απαραίτητος και κανένας επιπλέον έλεγχος με συνθήκες ελέγχου.

Τέλος, υπάρχει και ένα ενδεχόμενο (το πιο περίπλοκο) σύμφωνα με το οποίο δοθείσας μίας ακολουθίας, έστω I_3, εκτελούνται κλήσεις συναρτήσεων με καλούσα συνάρτηση, συνάρτηση μη αντιστοιχισμένη με κόμβο του $F[SiP]$ και κληθείσα συνάρτηση αντιστοιχισμένη, ή το αντίθετο. Στην πρώτη από τις δύο περιπτώσεις η μεταβλητή υδατογράφησης πρέπει να λάβει τιμή, πριν η ροή εκτέλεσης φτάσει στην καλούσα συνάρτηση, η οποία να δηλώνει πως η κλήση έχει προέλθει από συνάρτηση μη αντιστοιχισμένη με κόμβους του $F[SiP]$ και πως πρέπει να εκτελεστεί ο αρχικός κώδικας της κληθείσας και μόνο. Ενώ στη δεύτερη περίπτωση η μεταβλητή αυτή δεν λαμβάνει καμία τιμή (διατηρεί αυτή που έχει μέχρι στιγμής) και η ροή εκτέλεσης συνεχίζει κανονικά, καθώς η κληθείσα συνάρτηση δεν αντιστοιχεί σε κόμβο του $F[SiP]$ (δεν εμπεριέχει πρότυπα κλήσεων), οπότε και δεν δημιουργεί περαιτέρω προβλήματα λειτουργικότητας.

3.5.2 Φάση Εξαγωγής Υδατογραφήματος

Η φάση της εξαγωγής του μοντέλου υδατογράφησης λογισμικού WaterRPG αποκωδικοποιεί πληροφορία μέσα από τον κώδικα η οποία σε προγενέστερο στάδιο είχε κωδικοποιηθεί σε αυτόν. Κατά την διεξαγωγή της φάσης αυτής ο κώδικας δεν πειράζεται στο ελάχιστο, ούτε περνά η δομή του από αναλύσεις. Αυτό που λαμβάνει χώρα είναι η εκτέλεση του λογισμικού με την ακολουθία εισόδου κλειδί με σκοπό να δημιουργήσει το δυναμικό γράφημα κλήσεών του.



Σχήμα 3.15: Εξαγωγή υδατογραφήματος.

Η δημιουργία αυτή μπορεί να γίνει πράξη, είτε με κάποιον αποσφαλματωτή, είτε με κάποιον παρακολουθητή. Όπως και να συμβεί όμως, σκοπός είναι να καταγραφεί το ίχνος εκτέλεσης του κώδικα, δοθείσας της ακολουθίας εισόδου κλειδί και συγκεκριμένα όλες οι κλήσεις των συναρτήσεών του που πραγματοποιούνται προκειμένου να κατασκευαστεί / εξαχθεί το υδατογράφημα από τον κώδικα. Παρακάτω παρουσιάζονται τα βήματα του αλγορίθμου της δεύτερης και τελευταίας φάσης.

Είσοδος: κώδικας P^*, ακολουθία εισόδου I_{key}
Εξοδος: υδατογράφημα $F[SiP]$

---

1. Εκτέλεση του P^* με την ακολουθία εισόδου I_{key};
2. Κατασκευή του πίνακα CCh' από το ίχνος εκτέλεσης του P^*;
3. Εστω $DCG(V, E) = \emptyset$;
4. Κατασκευή του δυναμικού γραφήματος κλήσεων του P^* ως εξής:
Για κάθε εγγραφή $e = (f_i, f_j, *)$ του CCh

4.1 Αν $\notin$ στο V οι κόμβοι που αντιστοιχούν στις f_i και f_j εισέρχονται;
4.1 Αν $\notin$ στο E η ακμή που αντιστοιχεί στην (f_i, f_j) εισέρχεται;

Τέλος επανάληψης
5. Κατασκευή ενός γραφήματος $F[SiP]$ ισομορφικού με το γράφημα $DCG = (V, E)$;
6. Στο γράφημα $F[SiP]$ υπολογισμός του μοναδικού hamiltonian path;
7. Σύμφωνα με την σειρά επίσκεψης των κόμβων του $F[SiP]$ τοποθέτηση ετικετών;
8. **Επιστροφή** του $F[SiP]$;

Αλγόριθμος 6: Δεύτερη φάση εξαγωγής *Decode_code_to_RPG*

Παρατήρηση Βήματος 6: Στο Βήμα 6 του αλγορίθμου εξαγωγής του υδατογραφήματος από τον κώδικα, υπολογίζεται το hamiltonian path πάνω στο αναγώγιμο μεταθετικό γράφημα προκειμένου να μπουν οι ετικέτες στους κόμβους αυτού. Το μονοπάτι αυτό υπάρχει πάντα στο αναγώγιμο μεταθετικό γράφημα και μπορεί να βρεθεί σε γραμμικό χρόνο $O(n)$. Συγκεκριμένα, το μονοπάτι κατασκευάζεται ξεκινώντας την αναζήτηση πάνω στο γράφημα $F[SiP]$ από τον κόμβο που έχει μία εξερχόμενη ακμή (τοποθετώντας του την ετικέτα s) και κάνει την μετάβαση πάντα σε κόμβο που δεν έχει επισκεφθεί βάζοντας με την σειρά τις ετικέτες $n-2, n-3, \ldots, 1, t$.

Επομένως, στο σημείο αυτό έχουν δοθεί τόσο οι αλγόριθμοι για την κατασκευή του υδατογραφήματος όσο και οι αλγόριθμοι για την εφαρμογή υδατογράφησης στον κώδικα, ολοκληρώνοντας με τον τρόπο αυτόν μία πλήρης υδατογράφησης λογισμικού. Το μόνο που υπολείπεται της όλης διαδικασίας είναι να αποδειχθεί γιατί το μοντέλο WaterRPG εφαρμόζοντάς το δημιουργεί λειτουργικά ορθό κώδικα.

3.5.3 Απόδειξη Ορθότητας Αλγορίθμου

Στη θεωρητική επιστήμη των υπολογιστών, η ορθότητα (correctness) ενός αλγορίθμου διαπιστώνεται όταν λέγεται πως ο αλγόριθμος αυτός είναι σωστός σε σχέση με μία ή περισσότερες προδιαγραφές. Η λειτουργική ορθότητα (functional correctness) αναφέρεται στην συμπεριφορά εισόδου - εξόδου (input - output) ενός αλγορίθμου, έτσι ώστε για κάθε

είσοδο που του δίνεται αυτός να παράγει σωστή έξοδο. Ουσιαστικά, γίνεται μία διάκριση μεταξύ της ολικής ορθότητας (total correctness) η οποία επιπλέον απαιτεί ο αλγόριθμος να τερματίζει και της μερικής ορθότητας (partial correctness) η οποία απλά απαιτεί πως αν μία ερώτηση (π.χ., κλήση συνάρτησης) δοθεί, να επιστραφεί η σωστή πληροφορία ή αλλιώς αποτέλεσμα.

Από τη στιγμή όμως, που δεν υπάρχει κάποια γενική απάντηση στο πρόβλημα της διακοπής (halting problem) καθώς το πρόβλημα αυτό ανήκει στα NP - πλήρη προβλήματα, η διαπίστωση της ολικής ορθότητας είναι πολύ δύσκολο να συμβεί. Η απόδειξη τερματισμού είναι ένας τύπος μαθηματικής απόδειξης η οποία παίζει έναν σημαντικό ρόλο στην τυπική επαλήθευση καθώς η ολική ορθότητα ενός αλγορίθμου εξαρτάται από τον τερματισμό. Παρακάτω παρατίθενται μία απόδειξη ορθότητας του αλγορίθμου ενσωμάτωσης υδατογραφήματος, όχι αυστηρά μαθηματική, αλλά από μία πιο χαλαρή με σκοπό να καταστεί σαφές πως το μοντέλο WaterRPG δεν δημιουργεί προβλήματα λειτουργικότητας.

Ερώτηση

Γιατί ο υδατογραφημένος κώδικας ενός λογισμικού P έχει ακριβώς την ίδια λειτουργικότητα με τον κώδικα του υδατογραφημένου λογισμικού P^* το οποίο δημιουργείται από το μοντέλο υδατογράφησης λογισμικού WaterRPG, για οποιαδήποτε ακολουθία εισόδου I;

Για το P

Εστω πως το αρχικό πρόγραμμα P περνάει επιτυχώς την φάση της μεταγλώττισης (compilation) δημιουργώντας ένα εκτελέσιμο αρχείο ονόματι $execution\ file$. Το $execution\ file$ για κάθε ακολουθία εισόδου I_x παράγει ένα συγκεκριμένο αποτέλεσμα $result_x$ δημιουργώντας παράλληλα έναν δισδιάστατο πίνακα $dynamic\ call\ table_x$ ο οποίος εμπεριέχει με τη σειρά εκτέλεσης όλες τις κλήσεις συναρτήσεων $call(f_i, f_j)$ που πραγματοποιήθηκαν, μία σε κάθε του γραμμή. Κάθε γραμμή αποτελεί μία εγγραφή στον πίνακα και κάθε τέτοιος πίνακας αντιστοιχεί σε μοναδικό $dynamic\ call\ graph$.

Κάθε μία από τις κλήσεις συναρτήσεων $call(f_i, f_j)$ που εκτελούνται στο P, όπως είναι ευρέως γνωστό από τον τρόπο λειτουργίας των γλωσσών προγραμματισμού στην πληροφορική, μεταβιβάζει την ροή εκτέλεσης του κώδικα από την καλούσα συνάρτηση f_i στην κλειθήσα συνάρτηση f_j. Με τον τρόπο αυτό, κατά την διάρκεια εκτέλεσης του κώδικα, αρχικά κατασκευάζεται μία στοίβα (main programming stack) η οποία στο εσωτερικό της δημιουργεί εμφωλευμένες στοίβες (nested stacks) κάθε φορά που εκτελείται μία κλήση συνάρτησης και διαγράφει στοίβες κάθε φορά που η ροή εκτέλεσης του κώδικα επιστρέφει (return) από μία συνάρτηση. Η κάθε συνάρτηση δηλαδή δεσμεύει μία στοίβα η οποία είναι εμφωλιασμένη μέσα σε άλλες στοίβες, καθώς κάθε συνάρτηση από κάπου αλλού (άλλη συνάρτηση) έχει καλεστεί και λειτουργεί μέσα σε αυτή.

Για το P^*

Εστω πως το υδατογραφημένο πρόγραμμα P^* τώρα, περνάει και αυτό με επιτυχία την φάση της μεταγλώττισης δημιουργώντας ένα εκτελέσιμο αρχείο ονόματι *execution file**. Το εν λόγω αρχείο για κάθε ακολουθία εισόδου I_x παράγει και αυτό ένα αποτέλεσμα $result_x$ δημιουργώντας έναν δισδιάστατο πίνακα *dynamic call table**. Ο πίνακας αυτός (όπως και ο προηγούμενος που δημιουργείται από το μη υδατογραφημένο πρόγραμμα P), δοθείσας μίας ακολουθίας εισόδου I, περιέχει σύμφωνα με τη σειρά με την οποία εκτελέστηκαν όλες τις κλήσεις των συναρτήσεων $call(f_i, f_j)$ στον κώδικα του P^*. Επομένως και αυτός ο πίνακας με την σειρά του αντιστοιχεί σε ένα μοναδικό *dynamic call graph**.

Όσες εγγραφές ή αλλιώς ζεύγη κλήσεων στον πίνακα *dynamic call table* του P δεν αντιστοιχούν σε ακμές στο *reducible permutation graph*, στον πίνακα *dynamic call table** έχουν αντικατασταθεί πλέον με εγγραφές ή αλλιώς ζεύγη κλήσεων $call(f_i, f_j)$ τα οποία τώρα αντιστοιχούν σε ακμές στο *reducible permutation graph* και συγκεκριμένα τα εν λόγω ζεύγη αποτελούν το ελάχιστο μονοπάτι (shortest path) μεταξύ της καλούσας f_i και της κλειθήσας συνάρτησης f_j στο γράφημα $(f_i, f_{k_1}, f_{k_2}, \ldots, f_j)$. Υπάρχει πάντα μονοπάτι στο *reducible permutation graph* από έναν κόμβο σε έναν άλλον, λόγω του unique hamiltonian path που περιέχει [10].

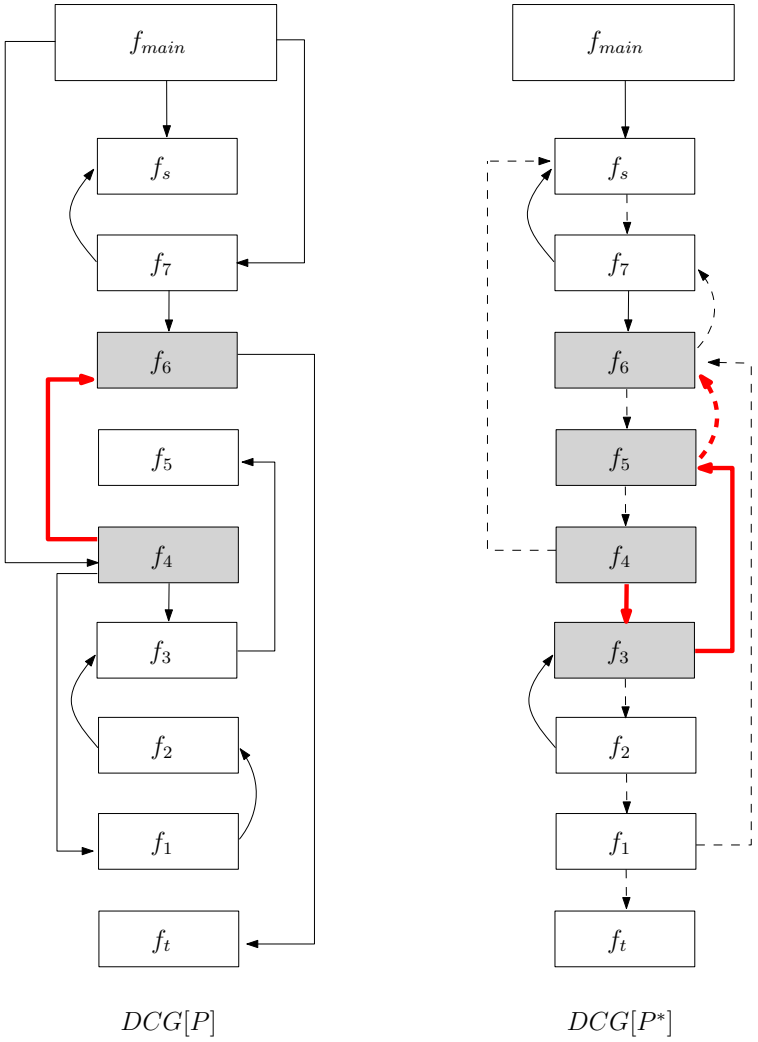
Οι κλήσεις συναρτήσεων οι οποίες μεσολαβούν μέχρι να φτάσει να εκτελεστεί η πραγματική κλήση συνάρτησης η οποία υπήρχε στο μη υδατογραφημένο πρόγραμμα αλλά όχι στο υδατογραφημένο (κάθε φορά που έχει αντικατασταθεί κάποια κλήση συνάρτησης του P) δεν δημιουργούν πρόβλημα στην λειτουργικότητα του κώδικα. Αυτό οφείλεται στο γεγονός ότι η μόνη λειτουργία που πραγματοποιούν οι κλήσεις αυτές είναι να μεταφέρουν την ροή εκτέλεσης στην σωστή συνάρτηση κάθε φορά. Συγκεκριμένα στις ενδιάμεσες κληθείσες συναρτήσεις του ελάχιστου μονοπατιού $(f_i, f_{k_1}, f_{k_2}, \ldots, f_j)$, δεν εκτελείται ο αρχικός κώδικας τους, αλλά ο κώδικας του υδατογραφήματος. Ο κώδικας αυτός περιέχει τρία διακριτά στοιχεία: μία συνθήκη ελέγχου η οποία αποτιμά ένα ή πολλά αδιαφανές κατηγορήματα $Q^{p^?}$, μία ανάθεση τιμής στην μεταβλητή υδατογράφησης και μία κλήση συνάρτησης $call(f_x, f_y)$.

Τέλος όσες κλήσεις οι οποίες αντιστοιχούν σε ακμές στο *reducible permutation graph* δεν έχουν προστεθεί ακόμη στον κώδικα του P^*, μετά την αντικατάσταση αυτών που δεν αντιστοιχούσαν σε ακμές, προστίθενται και αυτές προκειμένου το *dynamic call table** που θα δημιουργηθεί δοθείσας μίας συγκεκριμένης ακολουθίας εισόδου να δημιουργεί ένα *dynamic call graph** ισομορφικό με το ενσωματωμένο *reducible permutation graph* του προγράμματος P.

Για το P και το P^*

Αρα το πρόγραμμα P και το πρόγραμμα P^* για τις ίδιες ακολουθίες εισόδους I δημιουργούν διαφορετικά *dynamic call tables* και εν συνεπεία διαφορετικά *dynamic call graphs*, αλλά έχουν την ίδια ακριβώς λειτουργικότητα.

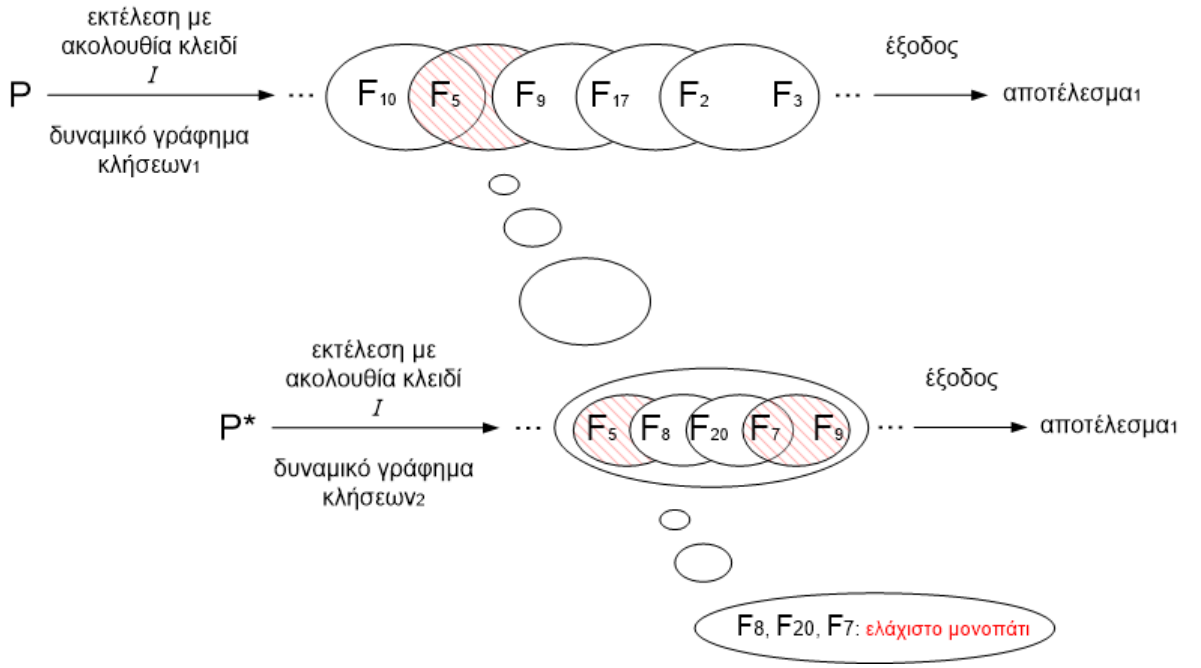
Αυτό είναι αλήθεια διότι, δοθείσας μία ακολουθίας εισόδου I, αν εκτελεστεί μία κλήση του P η οποία αντιστοιχεί σε ακμή του *reducible permutation graph* τότε η κλήση αυτή διατηρείται ανέπαφη στο P^*, ενώ αν εκτελεστεί μία κλήση του P η οποία δεν αντιστοιχεί σε ακμή του *reducible permutation graph* αυτή αντικαθίσταται με ζεύγη κλήσεων τα οποία εκτελούνται με την σειρά δημιουργώντας ένα μονοπάτι κλήσεων από την πραγματική καλούσα f_i προς την πραγματική κληθείσα f_j συνάρτηση.



Σχήμα 3.16: Ελάχιστο μονοπάτι πάνω στο δυναμικό γράφημα κλήσεων.

Το αρχικό πρόγραμμα P και το υδατογραφημένο πρόγραμμα P^* δοθέντος της ίδιας ακολουθίας εισόδου I παράγουν διαφορετικές αλληλουχίες κλήσεων συναρτήσεων οι οποίες δημιουργούν και διαφορετικά δυναμικά γραφήματα κλήσεων, πάρα ταύτα έχουν την ίδια ακριβώς λειτουργικότητα. Αυτό που στην πραγματικότητα συμβαίνει είναι στην υδατογραφημένη έκδοση του προγράμματος P μία κλήση συνάρτησης που υπήρχε στην μη υδατογραφημένη έκδοση έχει πλέον αντικατασταθεί με ένα μονοπάτι κλήσεων συναρτήσεων οι οποίες

αυτό που πραγματοποιούν είναι μονάχα η διοχέτευση της αρχικής κλήσης μέσω ενός υποσυνόλου συναρτήσεων, ο οποίες έχουν αντιστοιχηθεί με κόμβους του υδατογραφήματος, δίχως να εκτελεστεί ο αρχικός κώδικάς τους.



Σχήμα 3.17: Αλληλουχία κλήσεων συναρτήσεων στο P και στο P^*.

Κεφαλαιο 4

Εφαρμογη και Αξιολογηση του WaterRPG Μοντελου

4.1 Πτυχές Εφαρμογής του Μοντέλου WaterRPG

4.2 Ανάλυση του Μοντέλου Υδατογράφησης WaterRPG

4.3 Αξιολόγηση του Μοντέλου Υδατογράφησης WaterRPG

4.1 Πτυχές Εφαρμογής του Μοντέλου WaterRPG

Το δυναμικό μοντέλο υδατογράφησης λογισμικού WaterRPG έχει δύο πτυχές εφαρμογής σε κώδικα. Η πρώτη είναι η καθολική εφαρμογή. Στην εφαρμογή αυτή πραγματοποιείται η πιστή εφαρμογή του αλγορίθμου ενσωμάτωσης βήμα προς βήμα, δίχως να λαμβάνονται υπόψη τα χαρακτηριστικά του εκάστοτε κώδικα. Η δεύτερη είναι η διαφανής εφαρμογή. Στην εφαρμογή αυτή, σε αντίθεση με την προηγούμενη, γίνεται αξιοποίηση των χαρακτηριστικών και των ιδιοτροπιών του κώδικα που επρόκειτο να ενσωματωθεί το υποψήφιο υδατογράφημα κάνοντας με τον τρόπο αυτό όχι πιστή, αλλά υπό προϋποθέσεις και προσαρμοσμένες ενέργειες πάνω στον κώδικα έχοντας βέβαια ως απώτερο σκοπό αμφότερα στις περιπτώσεις αυτές την κωδικοποίηση γραφήματος μέσα σε κώδικα λογισμικού [11].

Οι πτυχές αυτές, ενσωματώνουν ακριβώς το ίδιο υδατογράφημα μέσα στον κώδικα. Με απλά λόγια δηλαδή, δεν επηρεάζεται το υδατογράφημα από την εφαρμογή με την οποία αυτό θα ενσωματωθεί μέσα στον κώδικα. Το πλεονέκτημα της καθολικής εφαρμογής έναντι της διαφανής είναι πως μπορεί να εφαρμοστεί εύκολα και γρήγορα ενσωματώνοντας ένα υδατογράφημα, έχοντας το τίμημα πως το υδατογράφημα αυτό ενδεχομένως να γίνεται πιο γρήγορα αντιληπτό στα μάτια κακόβουλων χρηστών από ότι θα γίνονταν αν το ίδιο υδατογράφημα είχε ενσωματωθεί με την διάφανη εφαρμογή. Με τη διαφορά, ότι στη

διάφανη εφαρμογή θα έπρεπε προηγουμένως ο κώδικα να μελετηθεί ενδελεχώς προκειμένου να εντοπιστούν τα υποψήφια σημεία προς βελτιστοποίηση του μοντέλου.

Σχήμα 4.1: Πτυχές εφαρμογής του δυναμικού μοντέλου WaterRPG.

4.1.1 Καθολική Εφαρμογή

Στην καθολική εφαρμογή του μοντέλου WaterRPG ο αλγόριθμος ενσωμάτωσης εφαρμόζεται με τον ίδιο ακριβώς τρόπο σε όλους τους κώδικες λογισμικών, δίχως να λαμβάνει υπόψη του τον τρόπο κατασκευής και τα περιεχόμενα των συναρτήσεων. Με τον τρόπο αυτόν, το βασικό πλεονέκτημα είναι πως το υδατογράφημα ενσωματώνεται στο λογισμικό γρήγορα, χωρίς να απαιτηθεί η ανάλυση του κώδικα, ενώ το μειονέκτημα είναι πως ακολουθώντας τα βήματα τυφλά ύστερα από την εφαρμογή υδατογράφησης στον κώδικα, μέσα σε αυτόν έχουν δημιουργηθεί πλέον πρότυπα στις συναρτήσεις οι οποίες παίζουν τον ρόλο των κόμβων του αναγώγιμου μεταθετικού γραφήματος τα οποία επαναλλαμβάνονται δίνοντας έτσι ένα κίνητρο περισσότερο σε κακόβουλους χρήστες να ελέγξουν τον κώδικα για τυχόν υδατογραφήσεις.

Με την καθολική εφαρμογή του μοντέλου υδατογράφησης στον κώδικα, στοχεύοντας στην κωδικοποίηση του υδατογραφήματος σε ένα από τα δυναμικά γραφήματα κλήσεων συναρτήσεων, εν περιλήψει σε κάθε ένα από τα ζεύγη κλήσεων (f_i, f_j) συμβαίνουν τα εξής στάδια:

1. Συμπεριλαμβάνεται ολόκληρο το σώμα της καλούσας συνάρτησης f_i εντός συνθήκης ελέγχου η οποία αποτιμά αδιαφανές κατηγόρημα το οποίο έχει κατασκευαστεί με τη βοήθεια της μεταβλητής υδατογράφησης, έστω x. Ύστερα, μέσα στον κώδικα της f_i για κάθε κλήση (f_i, f_j^1) και (f_i, f_j^2), αν αυτή υπάρχει, η οποία αντιστοιχεί σε μπροστά και πίσω ακμή του αναγώγιμου μεταθετικού γραφήματος αντίστοιχα προστίθεται πριν την εκτέλεση της κάθε κλήσης, ανάθεση στη μεταβλητή υδατογράφησης $x = x + h()$ αν η κλήση αντιστοιχεί σε μπροστά ακμή και $x = x + g()$ αν αντιστοιχεί

σε πίσω ακμή. Μετέπειτα κάθε ένας από τους συνδυασμούς ανάθεση και κλήσης εισέρχεται εντός συνθηκών ελέγχου οι οποίες και αυτές αποτιμούν με τη σειρά τους αποτιμούν αδιαφανή κατηγορήματα με τα οποία έχουν κατασκευαστεί με τη βοήθεια της μεταβλητής υδατογράφησης.

2. Στην περίπτωση κατά την οποία στο σώμα της καλούσας συνάρτησης f_i δεν υπάρχει κάποια κλήση της μορφής (f_i, f_j^1), τότε πριν από την εκτέλεση της συνθήκης ελέγχου η οποία συμπεριλαμβάνει το αρχικό σώμα της f_i προστίθεται ο συνδυασμός εντολών ανάθεσης $x = x + h()$ και κλήσης συνάρτησης (f_i, f_j^1) εντός συνθήκης ελέγχου η οποία αποτιμά συνθήκη όμοια με την συνθήκη του σταδίου 1 που αντιστοιχούσε σε μπροστά ακμή στο αναγώγιμο μεταθετικό γράφημα.

3. Στην περίπτωση όπου στο σώμα της καλούσας συνάρτησης f_i δεν υπάρχει κλήση της μορφής (f_i, f_j^2), τότε πριν από την εκτέλεση της συνθήκης ελέγχου η οποία συμπεριλαμβάνει το αρχικό σώμα της f_i και μετά από την τυχόν προσθήκη συνθήκης ελέγχου του σταδίου 1 προστίθεται ο συνδυασμός εντολών ανάθεσης $x = x + g()$ και κλήσης συνάρτησης (f_i, f_j^2) εντός συνθήκης ελέγχου η οποία αποτιμά συνθήκη όμοια με την συνθήκη του σταδίου 1 που αντιστοιχούσε σε πίσω ακμή στο αναγώγιμο μεταθετικό γράφημα.

4. Στις κλειθήσες συναρτήσεις f_j^1 και f_j^2 προστίθενται ως πρώτες εκτελέσιμες εντολές στον κώδικά τους μία ανάθεση στη μεταβλητή υδατογράφησης $x = x + c()$ η οποία βρίσκεται εντός μίας συνθήκης ελέγχου η οποία αποτιμά αδιαφανή κατηγορήματα τα οποία γίνονται αληθή κάθε φορά που η ροή εκτέλεσης εισέρχεται στις συναρτήσεις αυτές.

5. Σε όλες τις κληθείσες συναρτήσεις f_j^* του κώδικα της f_i οι οποίες δεν αντιστοιχούν σε κάποια από τις ακμές, είτε μπροστά, είτε πίσω του αναγώγιμου μεταθετικού γραφήματος υπολογίζεται η ακολουθία $(f_i, f_{k_1}, \ldots, f_j^*)$ η οποία αντιστοιχεί στο ελάχιστο μονοπάτι μεταξύ των κόμβων που αντιστοιχούν στις συναρτήσεις f_i και f_j^* στο γράφημα αυτό. Μετά η κλήση (f_i, f_j^*) διαγράφεται και στη θέση της προστίθεται ο συνδυασμός των εντολών $x = x + h()$ και (f_i, f_{k_1}) αν η κλήση αυτή αντιστοιχεί σε μπροστά ακμή ή $x = x + g()$ και (f_i, f_{k_1}) αν η κλήση αυτή αντιστοιχεί σε πίσω ακμή. Στη συνέχεια σε κάθε ένα ζεύγος συναρτήσεων, καλούσας και κλειθήσας, που εμφανίζονται στην ακολουθία $(f_i, f_{k_1}, \ldots, f_j^*)$, εκτός βέβαια του πρώτου, γίνονται οι αντίστοιχες ενέργειες προσθήκης συνδυασμών ανάθεση και κλήσης μέσα σε συνθήκες ελέγχου.

Ολες οι άλλες κλήσεις συναρτήσεων f_j^{**} οι οποίες βρίσκονται εντός της συνάρτησης f_i δεν τροποποιούνται, διότι πολύ απλά δεν ενεργοποιούνται με την ακολουθία εισόδου κλειδί, επομένως δεν συμβάλουν στη δημιουργία του δυναμικού γραφήματος κλήσεων το οποίο μετά το πέρας της υδατογράφησης θα είναι ισομορφικό με το αναγώγιμο μεταθετικό γράφημα.

4.1.2 Διαφανής Εφαρμογή

Στόχος της υδατογράφησης λογισμικού δεν αποτελεί μονάχα η ενσωμάτωση κάποιας πληροφορίας μέσα στον κώδικα, αλλά πρέπει ταυτόχρονα η πληροφορία αυτή να μην είναι ευδιάκριτη από κακόβουλους χρήστες. Έχοντας παρουσιάσει την καθολική εφαρμογή του μοντέλου WaterRPG, φάνηκε πως δημιουργούνται ορισμένα εμφανή πρότυπα μέσα στον κώδικα λογισμικού. Τα πρότυπα αυτά όμως με την διαφανή εφαρμογή του μοντέλου εξαλείφονται, ικανοποιώντας με τον τρόπο αυτόν και τον δεύτερο στόχο της υδατογράφησης λογισμικού.

Προτού παρουσιαστούν οι διαφανείς εφαρμογές του μοντέλου υδατογράφησης σημειώνεται πως αυτές δεν εφαρμόζονται τυφλά και ανεξάρτητα από τον κώδικα προς υδατογράφηση. Οι εφαρμογές αυτές είναι πλήρως εξαρτημένες από τον εκάστοτε κώδικα και ανάλογα με τα στοιχεία του που περιέχει μπορεί και βρίσκει εφαρμογή ένα υποσύνολο των διάφανων εφαρμογών. Παρακάτω παρατίθενται οι διάφανες εφαρμογές του WaterRPG δυναμικού μοντέλου υδατογράφησης λογισμικού:

1. **Δημιουργία εμφωλευμένων προτύπων κλήσεων συναρτήσεων.** Με την εμφώλευση αυτή μπορεί μέσα σε ένα πρότυπο κλήσης που αντιστοιχεί σε μπροστά ακμή στο αναγώγιμο μεταθετικό γράφημα να μπορεί να μπει και ένα πρότυπο κλήσης η οποία να αντιστοιχεί σε πίσω ακμή στο αναγώγιμο μεταθετικό γράφημα ή το αντίστροφο.

2. **Προσθήκη πολλαπλών εικονικών κλήσεων.** Από τη στιγμή που οι εικονικές κλήσεις εκτελούνται χωρίς δεν δημιουργούν προβλήματα λειτουργικότητας αφού ελέγχονται με συνθήκες ελέγχου και στην καλούσα και στην κλειθήσα συνάρτηση μπορούν να προστεθούν και πλεονάζουσες μέσα στον κώδικα κάθε αντιστοιχισμένης συνάρτησης με κόμβο του αναγώγιμου μεταθετικού γραφήματος.

3. **Απομάκρυνση συνθηκών ελέγχου.** Οι συνθήκες ελέγχου οι οποίες εμπεριέχουν τις αναθέσεις τύπου $x = x+c()$ στη μεταβλητή υδατογράφησης μπορούν να διαγραφούν με την προϋπόθεση πως η συνάρτηση στην οποία και βρίσκονται δεν καλείται από κάποια συνάρτηση μη αντιστοιχισμένη με κόμβο του αναγώγιμου μεταθετικού γραφήματος. Ο λόγος για τον οποίον συμβαίνει αυτό είναι πως αν διαγραφεί η συνθήκη ελέγχου και η συνάρτηση καλεστεί από κάποια άλλη συνάρτηση ασυσχέτιστη με το υδατογράφημα η μεταβλητή υδατογράφησης θα πάρει μη δεκτή τιμή και πιθανόν (ενδεχομένως και όχι, αλλά υπάρχει πιθανότητα) να δημιουργήσει πρόβλημα λειτουργικότητας στην εκτέλεση του κώδικα.

4. **Δημιουργία πολύπλοκων αδιαφανών κατηγορημάτων.** Τα αδιαφανή κατηγορήματα τα οποία εισάγονται σε συνθήκες ελέγχου με την καθολική εφαρμογή του μοντέλου εμπεριέχουν τελεστές τύπου: $=, \neq$. Στη διάφανη εφαρμογή οι τελεστές αυτοί μπορούν να αντικατασταθούν με τελεστές τύπου: $\leq, <, \geq, >$ οι οποίοι δεν ελέγχουν ακριβείς τιμές αλλά εύρη τιμών τα οποία δυσκολεύουν στην ανάλυση του κώδικα από τρίτους.

5. **Συνένωση συνθηκών ελέγχου.** Οι συνθήκες ελέγχου οι οποίες έχουν προστεθεί χάρη στο υδατογράφημα και εμπεριέχουν αδιαφανή κατηγορήματα, μπορούν να συνενωθούν με ήδη υπάρχουσες συνθήκες του κώδικα με σκοπό να διαγραφεί το επιπλέον τμήμα κώδικα το οποίο περικλείει τις εντολές του προτύπου κλήσεων.

6. **Ανάθεση τιμών με πολύπλοκες εκφράσεις.** Οι εκφράσεις που χρησιμοποιούνται για την ανάθεση τιμών στη μεταβλητή υδατογράηφησης μπορεί να γίνουν με κάποιον μαθηματικό τύπου και όχι απευθείας με ανάθεση τιμής, προκειμένου να μην είναι απευθείας ορατές οι τιμές που προστίθενται σε αυτήν.

7. **Χρήση πολλαπλών μεταβλητών υδατογράφησης.** Κατά τη διάρκεια εκτέλεσης με την ακολουθία εισόδου μπορεί να χρησιμοποιούνται περισσότερες από μία μεταβλητές για την υδατογράφησης λογισμικού οι οποίες θα αποτιμώνται στις επιπρόσθετες συνθήκες ελέγχου. Οι μεταβλητές αυτές είτε μπορεί να χρησιμοποιούνται όλες καθ όλη τη διάρκεια εκτέλεσης, είτε μπορεί να τεθεί ένα ξεχωριστό όριο σε κάθε μία από αυτές το οποίο θα σηματοδοτεί πότε σταματάει ή χρήση της και θα σηματοδοτεί τη χρήση της επόμενης.

8. **Τυχαία τοποθέτηση εικονικών κλήσεων.** Οι εικονικές κλήσεις σε κάθε αντιστοιχισμένη συνάρτηση μπορούν να προστεθούν, είτε πριν, είτε μετά από τη συνθήκη ελέγχου που περικλείει τον αρχικό κώδικα της συνάρτησης.

9. **Απομάκρυνση ανάθεσης στη μεταβλητή υδατογράφησης.** Στις περιπτώσεις όπου οι κλήση που εκτελείται είναι πραγματική τότε η ανάθεση τιμής πριν από την κλήση αυτή μπορεί και να διαγραφεί.

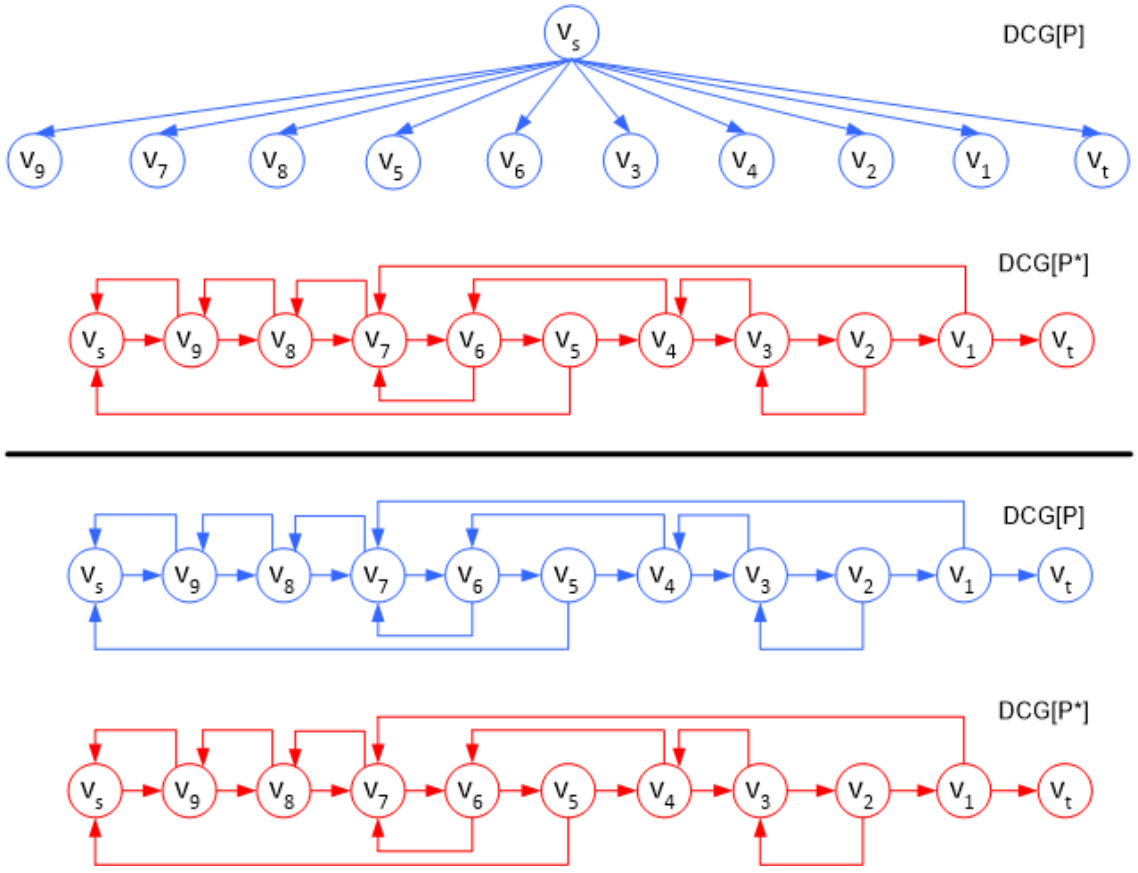
Κάνοντας χρήση ενός υποσυνόλου των προηγούμενων διάφανων εφαρμογών, πέραν από το γεγονός ότι εξαλείφονται τα πρότυπα μέσα στον κώδικα τα οποία ενδεχομένως να κινούσαν την περιέργεια κακόβουλων χρηστών, εξοικονομείται χρόνος εκτέλεσης καθώς λιγότερες εντολές άρα και λιγότερος χρόνος που αφιερώνεται σε αυτές και χώρος αποθήκευσης στην μόνιμη του υπολογιστή καθώς και στην εφήμερη (μνήμη που χρησιμοποιεί το λογισμικό για την εκτέλεση του κώδικα) καθώς.

4.2 Ανάλυση του Μοντέλου Υδατογράφησης WaterRPG

Το μοντέλο υδατογράφησης λογισμικού κάνει συστηματική τροποποίηση των κλήσεων συναρτήσεων του κώδικα με σκοπό να καταφέρει να κωδικοποιήσει μέσα σε ένα δυναμικό γράφημα κλήσεων ένα αναγώγιμο μεταθετικό γράφημα το οποίο παίζει το ρόλο του υδατογραφήματος. Η κωδικοποίηση αυτή απαιτεί πολυωνυμικό χρόνο εκτέλεσης και συγκεκριμένα χρόνο τάξης $O(n^2)$.

Από τη μία πλευρά, η καλύτερη περίπτωση εφαρμογής του μοντέλου αυτού είναι το μη υδατογραφημένο πρόγραμμα δοθείσας της ακολουθίας εισόδου κλειδί να παράγει εξ αρχής ένα αναγώγιμο μεταθετικό γράφημα. Σε αυτήν την περίπτωση με τη καθολική

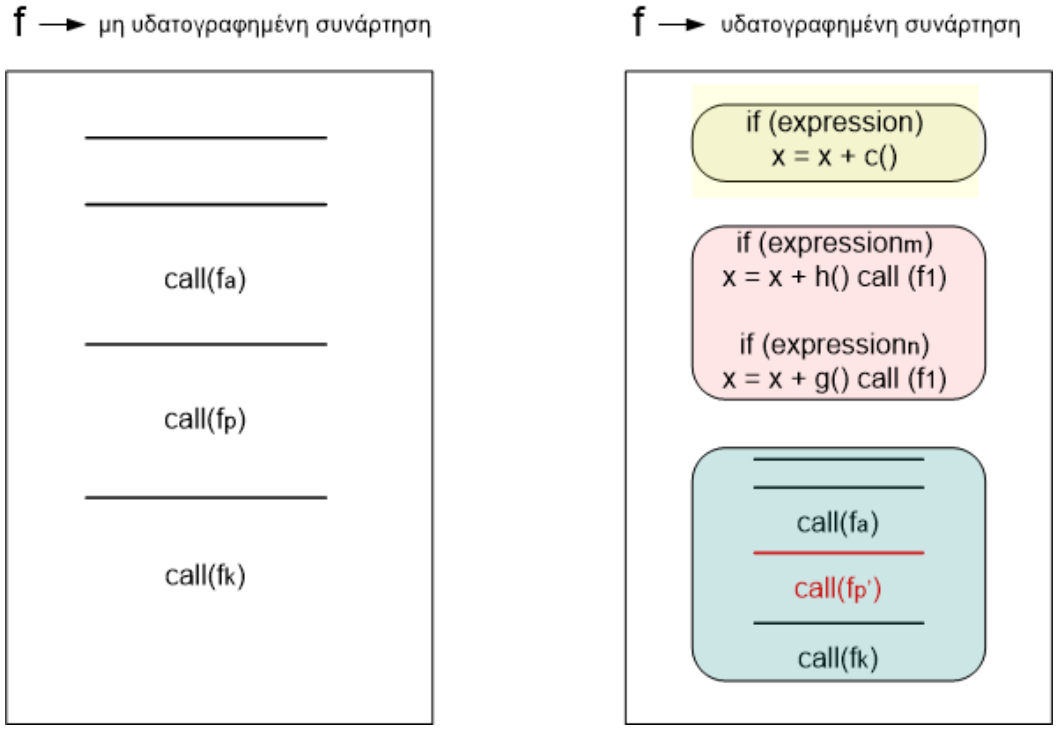
εφαρμογή υδατογράφησης δεν αλλάζεται καμία κλήση συνάρτησης στον κώδικα, αλλά εφαρμόζονται κανονικά τα πρότυπα κλήσεων (αυξάνοντας κατά ελάχιστο τον χώρο και τον χρόνο εκτέλεσης) ενώ με την περίπτωση της διάφανους εφαρμογής υδατογράφησης δεν αλλάζει ούτε καμία κλήση συνάρτησης αλλά και ούτε πειράζεται κάτι στον κώδικα. Ο κώδικας ουσιαστικά (προφανώς μη εσκεμμένα) εμπεριέχει υδατογράφημα εξ αρχής. Από την άλλη πλευρά, η χειρότερη περίπτωση εφαρμογής του μοντέλου είναι καμία από τις κλήσεις του αρχικού κώδικα να μην αντιστοιχεί σε κλήση στο αναγώγιμο μεταθετικό γράφημα και επίσης όλες οι κλήσεις να βρίσκονται εντός της αρχικής συνάρτησης με αποτέλεσμα να προστίθενται τα μέγιστα δυνατά ελάχιστα μονοπάτια εκτέλεσης κάθε φορά που χρειάζεται αντικατάσταση κλήσης η οποία δεν αντιστοιχεί σε ακμή στο αναγώγιμο μεταθετικό γράφημα.



Σχήμα 4.2: Η χειρότερη και η καλύτερη κατάσταση υδατογράφησης με βάση τα δυναμικά γραφήματα κλήσεων συναρτήσεων.

Ολοκληρώνοντας την υδατογράφηση λογισμικού με τη καθολική εφαρμογή στις συναρτήσεις οι οποίες έχουν αντιστοιχιστεί με κόμβους του αναγώγιμου μεταθετικού γραφήματος έχουν δημιουργηθεί τρία τμήματα κώδικα (blocks of code). Στο πρώτο από τα τμήματα αυτά περιλαμβάνεται ο κώδικας ο οποίος σχετίζεται με την ανάθεση τιμής $x = x + c()$ στη

μεταβλητή υδατογράφησης. Στο δεύτερο από τα τμήματα αυτά περιλαμβάνεται ο κώδικας ο οποίος σχετίζεται με τις τεχνητές κλήσεις συναρτήσεων οι οποίες δεν αντικαθιστούν κάποια πραγματική καθώς και με τις αναθέσεις $x = x + h()$ και $x = x + g()$ και τέλος στο τρίτο από αυτά τα τμήματα περιλαμβάνεται ο κώδικας ο οποίος υπήρχε στον μη υδατογραφημένο κώδικα, συμπεριλαμβανομένου των πραγματικών κλήσεων που αντικαταστάθηκαν με κίβδηλες μέσω των ελαχίστων μονοπατιών.



Σχήμα 4.3: Τμήματα που κώδικα που δημιουργούνται στις υδατογραφημένες συναρτήσεις.

4.2.1 Πραγματικό Κόστος Προτύπου Κλήσεων

Το μοντέλο δυναμικής υδατογράφησης λογισμικού ονόματι WaterRpg χρησιμοποιεί δύο ειδών πρότυπα κλήσεων προκειμένου να ενσωματώσει ένα υδατογράφημα μέσα στον κώδικα ενός προγράμματος. Τα πρότυπα αυτά ονομάζονται μπροστά πρότυπο (forward pattern) και πίσω πρότυπο (backward pattern) και σχετίζονται άμεσα με τις ακμές του αναγώγιμου μεταθετικού γραφήματος με τις οποίες κάθε φορά έχουν αντιστοιχηθεί. Καθώς ως γνωστόν κάθε μία κλήση συνάρτησης η οποία εμπεριέχεται στα πρότυπα, αντιστοιχεί σε μπροστά ακμή αν το πρότυπο που την περιέχει είναι το μπροστά πρότυπο, ενώ αντιστοιχεί σε πίσω ακμή αν το πρότυπο που την περιέχει είναι το πίσω πρότυπο.

Πάρα ταύτα ανεξάρτητα από το είδος του προτύπου που χρησιμοποιείται κάθε φορά από το WaterRpg μοντέλο, λαμβάνουν χώρα κάποιες ενέργειες προκειμένου αυτό να κωδικοποι-

ήσει κάποιο υδατογράφημα μέσα στον κώδικα. Οι ενέργειες αυτές είναι 4 στον αριθμό τους και επιβάλλουν, η κάθε μία από αυτές, την αντίστοιχη επιβάρυνση στον χώρο αποθήκευσης του λογισμικού, στον χρόνο εκτέλεσης αυτού καθώς επίσης και στον χώρο που χρειάζεται το λογισμικό προκειμένου να φορτωθεί στο σωρό εκτέλεσης (heap space) και να εκτελεστεί, το γνωστό στον προγραμματισμό σε γλώσσα Java ως heap space usage.

Σχήμα 4.4: Ενέργειες που επιτελούνται στην στοίβα κλήσεων για μία κλήση συνάρτησης.

Η πρώτη ενέργεια που περιλαμβάνεται στα πρότυπα λοιπόν, είναι η αποτίμηση του α-διαφανούς κατηγορήματος $Q_p^?$ σε αληθής (true) ή ψευδής (false), το οποίο υπάρχει στις συνθήκες ελέγχου και απαρτίζεται από την μεταβλητή υδατογράφησης και ενδεχομένως και από άλλες εκφράσεις με μεταβλητές ήδη υπάρχουσες του κώδικα. Ο χώρος και ο χρόνος που χρειάζεται, για αυτή την ενέργεια εξαρτάται από το πόσο απλή ή σύνθετη είναι η έκφραση που αποτελεί το αδιαφανή κατηγόρημα. Σημειώνεται, πως όσο πιο απλή είναι η εν λόγω έκφραση τόσο πιο γρήγορα γίνεται η αποτίμηση της κατά τον χρόνο εκτέλεσης (μικρή επιβάρυνση), αλλά τόσο πιο εύκολα μπορεί να σπάσει από κάποιον κακόβουλο χρήστη (γρήγορη αλλοίωση). Αποτελεί ουσιαστικά ένα trade - off το οποίο καλείται να επιλέξει ο προγραμματιστής τη στιγμή κατά την οποία υδατογραφεί τον κώδικα ενός λογισμικού.

Στη συνέχεια η δεύτερη και η τέταρτη κατά σειρά ενέργειες των προτύπων είναι παρόμοιες, καθώς αμφότερες οι εντολές που τις αποτελούν, επιτελούν αναθέσεις στην μεταβλητή υδατογράφησης, με σκοπό ο κώδικας του λογισμικού να γνωρίζει ανά πάσα στιγμή πως η ροή εκτέλεσης έχει διέλθει από το εκάστοτε σημείο. Τα κόστη αυτών των ενεργειών, εξαρτώνται άμεσα από την πολυπλοκότητα που χαρακτηρίζει την εκάστοτε ανάθεση τιμής, καθώς όσο πιο μικρή πολυπλοκότητα έχει, τόσο πιο λίγες ενέργειες απαιτούνται να λάβουν χώρα στην στοίβα κλήσεων (call stack) προκειμένου να λάβει τιμή η μεταβλητή υδατογράφησης, αλλά τόσο πιο γρήγορα θα μπορέσει να εξάγει την τιμή της μεταβλητής ένας κακόβουλος χρήστης. Επίσης ένα trade - off δηλαδή που καλείται να επιλέξει ο προγραμματιστής.

Τέλος η τρίτη κατά σειρά ενέργεια των προτύπων κλήσεων που πραγματοποιείται η κλήση συνάρτησης. Η εντολή αυτή ως γνωστόν, στον προγραμματισμό αυτό που κάνει είναι να μεταβιβάζει την ροή εκτέλεσης από μία συνάρτηση η οποία παίζει τον ρόλο της καλούσας και η οποία εμπεριέχει την κλήση σε μία άλλη η οποία παίζει τον ρόλο της κλειθήσας διαπράττοντας συγκεκριμένες εντολές [7].

Ύστερα από πειραματική μελέτη στην γλώσσα προγραμματισμού Java, βρέθηκε πως η εντολή κλήσης συνάρτησης, δίχως να περιέχει ορίσματα, απαιτεί περίπου στα 2 nano second προκειμένου να εκτελεστεί, σε έναν επεξεργαστή Intel core 2 duo, με συχνότητα στα 1,77 GHz. Η μέτρηση πραγματοποιήθηκε κάνοντας χρήση των εντολών μέτρησης χρόνου: $longstartTime = System.nanoTime()$; $longendTime = System.nanoTime()$; και $longduration = endTime - startTime$; εκτελώντας τα προγράμματα 20 φορές το κάθε ένα και παίρνοντας ύστερα τον μέσο όρο όλων το εκτελέσεων για το κάθε ένα από αυτά. Το γεγονός αυτό αποδεικνύει πως ο καθαρός χρόνος που απαιτεί προκειμένου να πραγματοποιηθεί μία κλήση συνάρτησης στην Java είναι απειροελάχιστος συγκριτικά με τον χρόνο που απαιτούν τα προγράμματα για να εκτελεστούν στην γενική των περιπτώσεων.

Είναι ευρέως γνωστό, πως στις γλώσσες προγραμματισμού οι οποίες κατά την διάρκεια εκτέλεσής τους κάνουν χρήση στοίβας, κάθε φορά που εκτελείται μία κλήση συνάρτησης δημιουργείται μία νέα στοίβα (υποστοίβα / εμφωλευμένη στοίβα) για την εκτέλεση της κληθείσας συνάρτησης. Η δημιουργία αυτή πραγματοποιείται εντός της ήδη υπάρχουσας στοίβας η οποία αντιστοιχεί στην καλούσα συνάρτηση [7]. Στη συνέχεια όποιες εντολές περιέχει η κληθείσα συνάρτηση εκτελούνται με την βοήθεια της νέας αυτής στοίβας.

Συγκεκριμένα, προκειμένου η ροή εκτέλεσης να περάσει από μία συνάρτηση σε μία άλλη μέσω κλήση συνάρτησης και εν συνεχεία να γυρίσει πάλι πίσω στο σημείο από όπου είχε κάνει αυτήν την μετάβαση, πρέπει να πραγματοποιηθούν 12 ενέργειες πάντα με την βοήθεια της στοίβας εκτέλεσης [34]. Αξίζει να τονιστεί πως από αυτές τις 12 ενέργειες, αυτές που κοστίζουν περισσότερο είναι εκείνες οι οποίες περιλαμβάνουν την επίλυση ενός ονόματος σε μία πραγματική μεταβλητή. Η επίλυση σημαίνει εντοπισμό της διεύθυνσης μνήμης και του τύπου της μεταβλητής. Για παράδειγμα στις γλώσσες προγραμματισμού οι οποίες μεταγλωττίζονται πρώτα προτού εκτελεστούν, ανήκουν η Java, η C και η Fortran. Αυτού του είδους η αναζήτηση όμως, σε κάποιες άλλες γλώσσες προγραμματισμού όπως είναι η JavaScript είναι ιδιαίτερα δαπανηρή διαδικασία, διότι κάθε φορά που εκτελείται μία

κλήση συνάρτησης πρέπει να γίνει επίλυση μίας διαφορετικής μεταβλητής ή συνάρτησης ενώ σε κάποιες άλλες όπως είναι η C δεν είναι. Επομένως, η κλήση συνάρτησης ενδεχομένως να διαφέρει αισθητά από τη μία γλώσσα προγραμματισμού στην άλλη [35].

4.2.2 Μεταβλητές Υδατογράφησης

Οι μεταβλητές που χρησιμοποιούνται για την υδατογράφηση λογισμικού μέσω του μοντέλου WaterRPG μπορεί να είναι διαφόρων τύπων (π.χ., ακέραιοι, χαρακτήρες) και να παίρνουν τιμές σύμφωνα με έναν προκαθορισμένο μαθηματικό τύπο. Δεν υπάρχει κάποιο άνω όριο στο πλήθος των μεταβλητών που πρέπει να χρησιμοποιηθούν προκειμένου να εφαρμοστεί το μοντέλο, ούτε και στις τιμές οι οποίες θα ανατίθενται στις μεταβλητές αυτές.

Ο ρόλος των μεταβλητών αυτών, είναι να δέχονται τιμές οι οποίες αφού αποτιμηθούν στις συνθήκες ελέγχου να καθορίζουν τα σωστά μονοπάτια εκτέλεσης προκειμένου να διατηρείται ορθή η λειτουργικότητα του κώδικα και δοθείσας της ακολουθίας εισόδου κλειδί να παράγεται δυναμικό γράφημα κλήσεων ισομορφικό με το ενσωματωμένο αναγώγιμο μεταθετικό γράφημα. Είναι σημαντικό να σημειωθεί, πως οι τιμές που θα λαμβάνουν οι μεταβλητές αυτές κατά την διάρκεια εκτέλεσης του λογισμικού με την ακολουθία κλειδί δεν πρέπει να επαναλαμβάνονται, δημιουργώντας αναθέσεις ίδιας τιμής σε διαφορετικά σημεία της ροής εκτέλεσης. Ο λόγος για τον δεν πρέπει να συμβεί αυτό είναι πως κάθε συνάρτηση πρέπει να γνωρίζει αν πρέπει να εκτελέσει τον αρχικό κώδικά της ή απλώς να μεταβιβάσει την κλήση συνάρτησης στην επόμενη συνάρτηση στο ελάχιστο μονοπάτι εκτέλεσης.

Επομένως, αν κατά την εκτέλεση η μεταβλητή υδατογράφησης τη μία φορά λάβει μία τιμή έστω $value$ πριν την κλήση μίας συνάρτησης έστω f η οποία πρέπει να εκτελέσει τον αρχικό της κώδικα και μία άλλη φορά η μεταβλητή αυτή λάβει την ίδια τιμή έστω $value$ και πριν την κλήση της f ενώ η f το μόνο που πρέπει να κάνει είναι να μεταβιβάσει την ροή εκτέλεσης χωρίς να εκτελέσει τον αρχικό της κώδικα τότε θα δημιουργηθεί πρόβλημα στην λειτουργικότητα του κώδικα. Παραδείγματα τέτοιων ακολουθιών μπορεί να είναι τα εξής:

- fibonacci
- lucas
- odd even
- ulam
- composite
- simple sequence: $a_n = a_1 + (n-1) * d;\ a_1 \& d \in Z$

Προκειμένου η επιλεχθείσα ακολουθία να μην δημιουργεί πρόβλημα κατά την διάρκεια εκτέλεσης στον κώδικα και παράλληλα να μην δίνει στόχο σε κακόβουλους χρήστες πρέπει η αύξηση ή μείωση των αριθμών της να μην γίνεται γρήγορα. Επίσης ο αλγόριθμος που χρησιμοποιεί η ακολουθία αυτή πρέπει να είναι έξυπνος και η εύρεση του ν-οστού όρου της πρέπει να γίνεται τάχιστα για επιφέρει το ελάχιστο δυνατό κόστος εκτέλεσης. Επιπλέον, ο

αριθμός μεταβλητών προκειμένου να παραχθούν οι τιμές που θα ανατεθούν πρέπει να είναι ο ελάχιστος δυνατός.

Μία καλή επιλογή τύπου για τις μεταβλητές υδατογράφησης στοχεύοντας στην ελαχιστοποίηση του κόστους εκτέλεσης είναι οι ακέραιοι αριθμοί. Οι ακέραιοι στην γλώσσα προγραμματισμού Java χωρίζονται σε 3 κατηγορίες. Η πρώτη κατηγορία ονομάζεται short integer και οι αριθμοί αυτής της κατηγορίας καταλαμβάνουν χώρο στη μνήμη ίσο με 16 bits, η δεύτερη κατηγορία ονομάζεται integer και οι αριθμοί που ανήκουν σε αυτήν καταλαμβάνουν χώρο ίσο με 32 bits και τέλος η τρίτη κατηγορία ονομάζεται long integer και οι αριθμοί που ανήκουν σε αυτή χώρο ίσο με 64 bits. Από την άλλη πλευρά, οι πραγματικοί αριθμοί στην γλώσσα Java χωρίζονται σε 2 κατηγορίες. Στην πρώτη κατηγορία η οποία ονομάζεται float, ανήκουν οι αριθμοί που καταλαμβάνουν 32 bits στην μνήμη και στην δεύτερη κατηγορία η οποία ονομάζεται double precision, ανήκουν οι αριθμοί που καταλαμβάνουν 64 bits.

Γνωρίζοντας κάποιος τις πληροφορίες αυτές, θα μπορούσε να εξάγει το συμπέρασμα πως στο δυναμικό μοντέλο υδατογράφησης λογισμικού ονόματι, WaterRpg, είτε χρησιμοποιηθούν ακέραιες μεταβλητές είτε χρησιμοποιηθούν πραγματικές μεταβλητές στην υδατογράφηση λογισμικών σε Java θα έχει το ίδιο αποτέλεσμα και σε χώρο και σε χρόνο εκτέλεσης. Πάρα ταύτα το συμπέρασμα αυτό είναι εσφαλμένο καθώς ο ηλεκτρονικός υπολογιστής προκειμένου να εκτελέσει πράξεις με πραγματικούς αριθμούς χρειάζεται περισσότερο χρόνο. Το γεγονός αυτό είναι αληθές διότι, αν και η εσωτερική αναπαράστασή τους αμφότερα γίνεται με 0 και 1 η πληροφορία που φέρουν οι τύποι αυτοί είναι διαφορετική.

4.3 Αξιολόγηση του Μοντέλου Υδατογράφησης WaterRPG

Προκειμένου να εξαχθούν αντικειμενικά συμπεράσματα γύρο από την πρακτική συμπεριφορά του δυναμικού μοντέλου υδατογράφησης λογισμικού WaterRPG, αυτό εφαρμόστηκε πάνω σε διάφορα προγράμματα γραμμένα στην γλώσσα προγραμματισμού Java και αξιολογήθηκε σύμφωνα με ένα υποσύνολο των κριτηρίων αξιολόγησης. Τα προγράμματα αυτά είτε κωδικοποιούσαν απλά παιχνίδια, είτε κωδικοποιούσαν ευρέα διαδεδομένους αλγορίθμους. Η υδατογράφηση στους κώδικες έγινε και με τη καθολική και με τη διάφανη εφαρμογή με σκοπό να φανεί η διαφορά και των δύο αυτών προσεγγίσεων, συγκριτικά με τον αρχικό μη υδατογραφημένο κώδικα.

4.3.1 Αξιολόγηση με Βάση Ιδιότητες

Στην αξιολόγηση αυτού του τύπου, δεν χρειάζεται εφαρμογή του μοντέλου υδατογράφησης προκειμένου να εξαχθούν συμπεράσματα για τη συμπεριφορά του (practical behaviour). Από τη στιγμή που το WaterRPG φέρει συγκεκριμένες ιδιότητες καθίσταται απευθείας γνωστό αν αντιστέκεται ή όχι σε τροποποιήσεις ή αλλιώς σε επιθέσεις οι οποίες μπορούν να λάβουν χώρα στον υδατογραφημένο κώδικα (π.χ., αν ένα υδατογράφημα στηρίζεται στην σειρά με την οποία εμφανίζονται τα τμήματα του κώδικα και κάποιος τροποποιήσει την

σειρά αυτή, δεν είναι απαραίτητο να πραγματοποιηθεί η εξαγωγή του υδατογραφήματος για να δειχθεί αν αντέχει ή όχι το υδατογράφημα στην ενέργεια αυτή).

Οι επιθέσεις αυτές μπορεί να εντάσσονται είτε στις τεχνικές της συσκότισης, είτε στις τεχνικές της βελτιστοποίησης κώδικα λογισμικού. Το κάθε ένα από τα δύο αυτά είδη επιθέσεων, χωρίζεται σε κατηγορίες, ορισμένες από τις οποίες εμπεριέχουν ως επιμέρους υποκατηγορίες ορισμένες κοινές. Συγκεκριμένα οι τεχνικές της συσκότισης κώδικα διακρίνονται σε βασικές κατηγορίες ανάλογα με τα σημεία εστίασής τους. Οι κατηγορίες αυτές είναι οι εξής:

- Συσκοτίσεις διάταξης (layout obfuscations)
- Συσκοτίσεις δεδομένων (data obfuscations)
- Συσκοτίσεις ελέγχου (control obfuscations)
- Συσκοτίσεις πρόληψης (preventive obfuscations)

Το μοντέλο υδατογράφησης λογισμικού WaterRPG είναι ευάλωτο μονάχα στις συσκοτίσεις ελέγχου και συγκεκριμένα σε ένα υποσύνολο τεχνικών της κατηγορίας αυτής. Οι συσκοτίσεις ελέγχου περιλαμβάνουν επιμέρους κατηγορίες οι οποίες είναι η συσσωμάτωση (aggregation), η διάταξη (ordering), και ο υπολογισμός (computation) από τις οποίες το μοντέλο WaterRPG αντιμετωπίζει προβλήματα μόνο με τη συσσωμάτωση. Επί της ουσίας η συσσωμάτωση αυτό που κάνει είναι ή να συγχωνεύει (inline methods) ή να διαχωρίζει (outline methods) ή να κλωνοποιεί συναρτήσεις (clone methods) του κώδικα. Επομένως, αν οι συναρτήσεις αυτές έχουν αντιστοιχιστεί με κάποιους από τους κόμβους του αναγώγιμου μεταθετικού γραφήματος να υπεισέρχονται προβλήματα αναγνώρισης του υδατογραφήματος. Οι ενέργειες αυτές αντιστοιχούν σε επιθέσεις στους κόμβους του αναγώγιμου μεταθετικού γραφήματος.

Όσο αναφορά τις υπόλοιπες κατηγορίες συσκότισης, από τη στιγμή που καμιά τους δεν επεμβαίνει σε θέματα όπως τροποποίηση της αλληλουχίας εκτέλεσης των κλήσεων συναρτήσεων (π.χ., αντικατάσταση μίας κλήσης με μία άλλη), το υδατογράφημα δεν κινδυνεύει από τέτοιες εφαρμογές.

Οι τεχνικές της βελτιστοποίησης κώδικα χωρίζονται και αυτές σε κατηγορίες ανάλογα με τις τροποποιήσεις που λαμβάνουν χώρα στον κώδικα. Συγκεκριμένα οι κατηγορίες αυτές είναι οι ακόλουθες:

- Βελτιστοποιήσεις δομών κώδικα
- Βελτιστοποιήσεις δεδομένων
- Βελτιστοποιήσεις σε επίπεδο εκτέλεσης

Οι κατηγορίες της βελτιστοποίησης κώδικα χωρίζονται και αυτές με τη σειρά τους σε επιμέρους υποκατηγορίες. Από τις κατηγορίες αυτές, μόνο στις βελτιστοποιήσεις δομών

του κώδικα υπάρχουν ορισμένες υποκατηγορίες, όπως η διαδιεργασιακή ανάλυση (inter-procedural analysis) η οποίες αν εφαρμοστούν μπορούν και αλλοιώνουν το ενσωματωμένο υδατογράφημα. Οι υπόλοιπες με την εφαρμογή τους, καθιστούν τον κώδικα γρηγορότερα εκτελέσιμο δίχως να επηρεάζουν τη δομή του υδατογραφήματος.

4.3.2 Πειραματική Αξιολόγηση

Οι κώδικες που χρησιμοποιήθηκαν, είτε κατασκευάστηκαν εξ αρχής για αυτόν τον σκοπό, είτε κατεβάστηκαν από ιστοσελίδες οι οποίες τα διέθεταν δωρεάν και ήταν όλοι τους περίπου ίδιας έκτασης και δομής. Σε αυτούς ενσωματώθηκαν ως υδατογραφήματα, αναγώγιμα μεταθετικά γραφήματα τα οποία περιείχαν 11, 13, και 15 κόμβους και με την καθολική εφαρμογή και με τη διάφανη. Τα πειράματα υδατογράφησης έλαβαν χώρα σε ηλεκτρονικό υπολογιστή με επεξεργαστή dual-core 2.0 GHZ ο οποίος είχε μνήμη στα 3.0 GB, έτρεχε το λειτουργικό σύστημα των Windows Vista και είχε εγκατεστημένη την Java έκδοσης 1.6.0.26.

Το δυναμικό μοντέλο υδατογράφησης λογισμικού WaterRPG εφαρμόστηκε πάνω στον πηγαίο κώδικα λογισμικών τα οποία στη συνέχεια ελέγχθηκαν για την σωστή τους λειτουργία και κατόπιν αξιολογήθηκαν με βάσει τα ακόλουθα κριτήρια:

- Χώρος αποθήκευσης μόνιμης και όχι
- Χρόνος εκτέλεσης
- Αντίκτυπο εντολών σε επίπεδο bytecode
- Ανθεκτικότητα υδατογράφησης
- Διαφάνεια

Στο σημείο αυτό σημειώνεται πως ο χώρος αποθήκευσης διακρίθηκε σε 2 είδη. Τον χώρο που απαιτεί το εκτελέσιμο αρχείο για να αποθηκευτεί στη μόνιμη μνήμη του ηλεκτρονικού υπολογιστή και τον χώρο που απαιτεί το εκτελέσιμο αρχείο στη μνήμη (heap space usage) προκειμένου να εκτελεστεί. Ο χώρος που απαιτεί το εκτελέσιμο αρχείο κατά την εκτέλεσή του επί της ουσίας είναι ο χώρος της στοίβας εκτέλεσης και ότι αυτή περιέχει.

Παρακάτω παρατίθενται πίνακες σχετικοί με την πειραματική αξιολόγηση, οι οποίοι περιέχουν πληροφορία για το πλήθος των κλήσεων συναρτήσεων στις υδατογραφημένες εκδόσεις και στη μη υδατογραφημένη έκδοση, για τον χρόνο εκτέλεσης, για τον χώρο που απαιτείται για την μόνιμη αποθήκευση καθώς και για τον χώρο που απαιτείται για την προσωρινή αποθήκευση κατά την διάρκεια εκτέλεσης [12].

Οι χρόνοι οι οποίοι μετρήθηκαν προκειμένου να υπολογιστεί το ποσοστό αύξησης στο σωρό εκτέλεσης έγινε με τη βοήθεια του παρακολουθητή (profiler) του προγραμματιστικού (software development kit - SDK) εργαλείου NetBeans, ενώ οι χρόνοι οι οποίοι μετρήθηκαν προκειμένου να υπολογιστεί το ποσοστό αύξησης στον χρόνο εκτέλεσης έγινε με τη βοήθεια της εντολής $System.currentTimeMillis()$ η οποία επιστρέφει χρόνο σε νάνο δευτερόλεπτα.

Το κάθε ένα από τα προγράμματα εκτελέστηκε 20 φορές, δοθέντος 10 διαφορετικών ακολουθιών εισόδου, υπολογίζοντας μετέπειτα τον μέσο όρο από αυτές και στη συνέχεια τον μέσο όρο από όλα τα προγράμματα.

Πίνακας 4.1: Πλήθος Κλήσεων σε P και σε P^*

Κόμβοι $F[\pi^*]$	Κλήσεις P	Κλήσεις P^*	Πραγματικές	Εικονικές
11	32	48	20	28
13	32	51	21	30
15	32	56	19	37

Πίνακας 4.2: Χρόνος Εκτέλεσης (msec)

Κόμβοι $F[\pi^*]$	$P \to P_N^*$	$P \to P_S^*$	$P_N^* \to P_S^*$
11	+5.25%	+3.82%	-1.37%
13	+7.65%	+5.99%	-1.56%
15	+11.07%	+9.19%	-1.72%

Πίνακας 4.3: Χώρος Μόνιμης Αποθήκευσης (Kb)

Κόμβοι $F[\pi^*]$	$P \to P_N^*$	$P \to P_S^*$	$P_N^* \to P_S^*$
11	+20.98%	+16.71%	-3.65%
13	+26.35%	+18.81%	-6.34%
15	+30.10%	+21.76%	-6.85%

Πίνακας 4.4: Χώρος Αποθήκευσης Σωρού (Mb)

Κόμβοι $F[\pi^*]$	$P \to P_N^*$	$P \to P_S^*$	$P_N^* \to P_S^*$
11	+7.69%	+4.61%	-2.94%
13	+10.76%	+6.15%	-4.34%
15	+15.38%	+9.23%	-5.63%

Τα αποτελέσματα τα οποία απεικονίζονται στους προηγούμενους πίνακες δείχνουν περίπου μία σταθερή αύξηση καθώς αυξάνεται το πλήθος των κόμβων του αναγώγιμου μεταθετικού γραφήματος τόσο σε απαιτούμενο χρόνο περαίωσης όσο και σε απαιτούμενο χώρο

φόρτωσης και αποθήκευσης. Σύμφωνα με την πληροφορία αυτή, εξάγεται το συμπέρασμα πως ο επιπρόσθετος κώδικας του υδατογραφήματος δεν σχετίζεται άμεσα με τον κώδικα προς υδατογράφηση, αλλά έμμεσα.

Ουσιαστικά, ο επιπρόσθετος κώδικας που συνδέεται με τον υδατογράφημα είναι ανάλογος με το πλήθος των κόμβων οι οποίοι θα αντιστοιχιστούν με συναρτήσεις και όχι με αυτόν καθ αυτόν τον κώδικα των συναρτήσεων του κώδικα. Επιπλέον, στα προγράμματα αυτά χρησιμοποιήθηκε πάνω από το 70% των συναρτήσεων που περιείχαν. Γεγονός το οποίο δείχνει πως αν στα ίδια προγράμματα υπήρχαν περισσότερες συναρτήσεις στον κώδικα οι οποίες δεν χρησιμοποιούνταν για την υδατογράφηση τα ποσοστά των πινάκων θα κυμαίνονταν σε ακόμη χαμηλότερα επίπεδα.

Επιπλέον, στα πειράματα τα οποία έλαβαν χώρα έγινε και καταμέτρηση των εντολών σε επίπεδο bytecode (εκτελέσιμο αρχείο σε Java) προκειμένου να εντοπιστεί το σύνολο των εντολών που χρησιμοποιείται για την ενσωμάτωση του υδατογραφήματος. Τα αποτελέσματα έδειξαν πως μόνο οι εντολές οι οποίες είναι υπεύθυνες για τις αναθέσεις τιμών σε μεταβλητές και οι εντολές οι οποίες είναι υπεύθυνες για τις συνθήκες ελέγχου είχαν υποστεί αύξηση. Αυτό είναι απολύτως λογικό καθώς αυτές οι εντολές και μόνο χρησιμοποιήθηκαν προκειμένου να χειραγωγηθεί η ροή εκτέλεσης και να αλλάξει το παραγόμενο δυναμικό γράφημα κλήσεων και να διατηρηθεί ορθή η λειτουργικότητα εκτέλεσης.

Αυξηση επίσης είχαν υποστεί και οι εντολές οι οποίες καλούν συναρτήσεις, αλλά σε πολύ χαμηλότερο επίπεδο, καθώς στην υδατογράφηση πολλές από τις πραγματικές κλήσεις συναρτήσεων χρησιμοποιούνται για τον σκοπό αυτόν. Παρατηρείται επίσης πως σε περιπτώσεις όπου εφαρμόζεται σε μεγάλο εύρος η διαφανής εφαρμογή του μοντέλου WaterRPG παρατηρείται αισθητή μείωση του επιπρόσθετου φορτίου σε εντολές από ότι είχε δημιουργηθεί με την καθολική εφαρμογή.

Παρακάτω παρατίθενται πίνακες σχετικοί με την πειραματική αξιολόγηση, οι οποίοι περιέχουν πληροφορία για το πλήθος των εντολών στις υδατογραφημένες εκδόσεις και στη μη υδατογραφημένη έκδοση χωρισμένες σε επιμέρους σύνολα. Τα σύνολα αυτά, αποτελούν και τα σύνολα των εντολών οι οποίες χρησιμοποιούνται προκειμένου να γίνει πράξη η υδατογράφηση λογισμικού.

Πίνακας 4.5: Bytecode Σύνολα Εντολών

Bytecode	P	Καθολική P_N^*	Διάφανη P_S^*
Control Statements	519.4	42.0%	25.0%
Invocations	188.3	10.6%	10.6%
Assignments	1346.7	45.5%	32.4%
Rest Instructions	941.6	0%	0%

Πίνακας 4.6: Ενδεικτικές Εντολές Bytecode

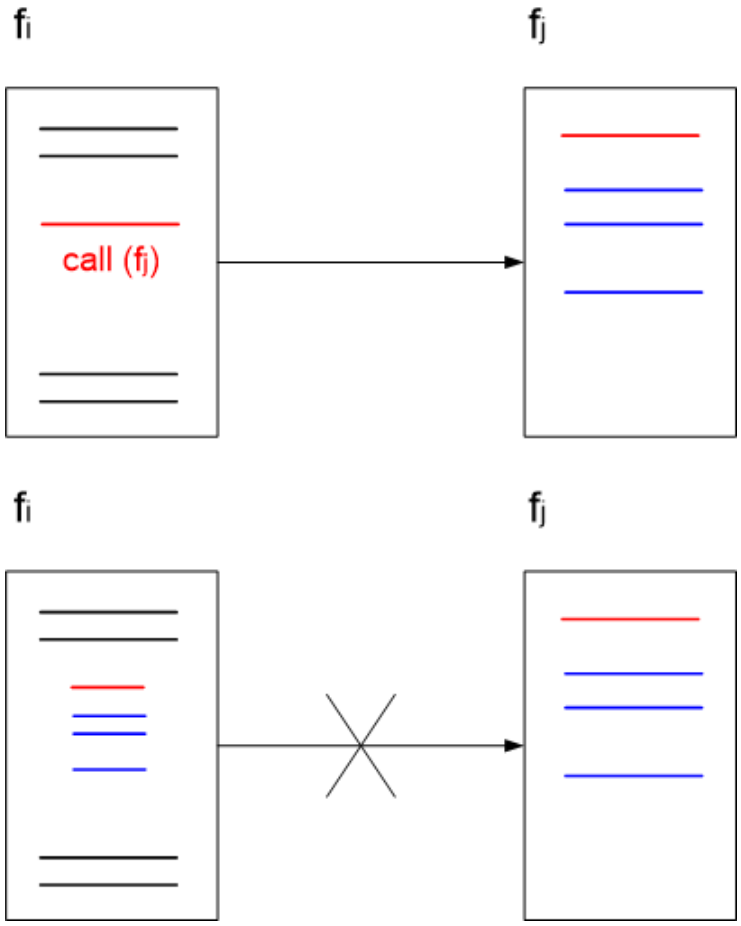
Bytecode	P	Καθολική P_N^*	Διάφανη P_S^*
Control Statements			
`if_icmpne`	19	78	57
`ifne`	3	4	4
`goto`	43	45	45
Invocations			
`invokevirtual`	188	208	208
Assignments			
`iconst_1`	186	202	202
`getstatic`	368	614	529
`iadd`	84	132	132
`alaod_0`	136	156	156
Rest Instructions			
`dup`	33	33	33
`ldc`	19	19	19
`pop`	3	3	3

Ολοκληρώνοντας την πειραματική αξιολόγηση του μοντέλου υδατογράφησης λογισμικού WaterRPG έγιναν εφαρμογές συσκότισης και βελτιστοποίησης κώδικα με τη βοήθεια του εργαλείου ProGuard με απώτερο σκοπό να καταστεί σαφές αν όντως το υδατογράφημα παραμένει αναλλοίωτο στους κώδικες των προγραμμάτων που χρησιμοποιήθηκαν και για τα προηγούμενα αποτελέσματα. Οι ιδιότητες του μοντέλου δείχνουν υπό ποιες προϋποθέσεις το ενσωματωμένο υδατογράφημα αλλοιώνεται, το μόνο που απομένει λοιπόν είναι να γίνει εμφανές αυτό και επί του πρακτέος και να επιβεβαιωθεί.

Το εργαλείο ProGuard είναι ένα λογισμικό το οποίο δέχεται ως είσοδο ένα εκτελέσιμο αρχείο σε Java και εφαρμόζει σε αυτό τεχνικές συσκότισης και βελτιστοποίησης κώδικα, παράγοντας ως αποτέλεσμα ένα τροποποιημένο αρχείο με την ίδια ακριβώς λειτουργικότητα με το αρχικό. Στη συνέχεια, για να ελεγχθεί ο πηγαίος τροποποιημένος πλέον κώδικας χρησιμοποιήθηκε ένα άλλο εργαλείο, το Java Decompiler, το οποίο δεχόταν ως είσοδο ένα εκτελέσιμο αρχείο σε Java και παρήγαγε μέσω αντίστροφης μηχανικής (reverse engineering) τον πηγαίο κώδικα αυτού.

Τα αποτελέσματα των συσκοτίσεων και των βελτιστοποιήσεων, ελέγχοντας το αρχείο το οποίο παρήγαγε ως έξοδο το εργαλείο Java Decompiler, έδειξαν πως είχαν γίνει αλλαγές σε όλο το φάσμα του κώδικα (π.χ., για τη συσκότιση έλαβαν χώρα η μετονομασία μεταβλητών και συναρτήσεων, η διαγραφή πληροφοριών για την αποσφαλμάτωση, και για την βελτιστοποίηση το ξεδίπλωμα ορισμένων δομών επανάληψης, η αντικατάσταση ορισμένων σύνθετων εκφράσεων στις δομές ελέγχου με απλούστερες, η επαναχρησιμοποίηση μεταβλητών, η αντικατάσταση ορισμένων κλήσεων συναρτήσεων με τον κώδικα της κλειθήσας συνάρτησης και η διαγραφή κλήσεων συστήματος).

Το συμπέρασμα είναι πως δημιουργείτο πρόβλημα μονάχα σε ορισμένα σημεία του κώδικα που υπήρχαν κλήσεις συναρτήσεων οι οποίες κατά τη βελτιστοποίηση κώδικα, για λόγους ταχύτερης εκτέλεσης, αυτές είχαν αντικατασταθεί με το σώμα της κλειθήσας συνάρτησης που έπρεπε να εκτελεστεί. Ουσιαστικά δεν εκτελούνταν η κλήση συνάρτησης η οποία κατά την καταγραφή του ίχνους εκτέλεσης, στη διαδικασία εξαγωγής του υδατογραφήματος, θα έπρεπε να συμβεί για να αναγνωριστεί κάποια ακμή από το αναγώγιμο μεταθετικό γράφημα, αλλά μέσα στην ίδια συνάρτηση έμπαινε στο σημείο της κλήση ο κώδικας της κληθείσας συνάρτησης.



Σχήμα 4.5: Αντικατάσταση κλήσης με κύριο σώμα κληθείσας συνάρτησης.

Οι κλήσεις αυτών των συναρτήσεων, οι οποίες αντικαταστάθηκαν στη βελτιστοποιημένη έκδοση, παρατηρήθηκε πως καλούσαν συναρτήσεις μικρής έκτασης που περιείχαν απλούς υπολογισμούς (π.χ., σειριακό κώδικα που επιτελούσε πράξεις όπως πρόσθεση και αφαίρεση) και βρισκόντουσαν στην ίδια κλάση με την καλούσα συνάρτηση (κοντινές κατειλημμένες διευθύνσεις μνήμης) και η δημιουργία υποστοίβας εντός της τρέχουσας στοίβας εκτέλεσης ήταν ασύμφορη ως προς τον χρόνο εκτέλεσης. Το γεγονός αυτό δείχνει πως για την

υδατογράφηση λογισμικού με το μοντέλο WaterRPG πρέπει να επιλέγονται συναρτήσεις για να αντιστοιχιστούν με κόμβους του αναγώγιμου μεταθετικού γραφήματος οι οποίες εμπεριέχουν ή κώδικα μεγάλης έκτασης με όχι μόνο σειριακές εντολές, ή να βρίσκονται σε διαφορετικές κλάσεις.

Κεφαλαιο 5

Συγκρισεις Μοντελων Υδατογραφησης Λογισμικου

5.1 Συγκρίσεις Ιδιοτήτων

5.2 Συγκρίσεις Ανθεκτικότητας

5.1 Συγκρίσεις Ιδιοτήτων

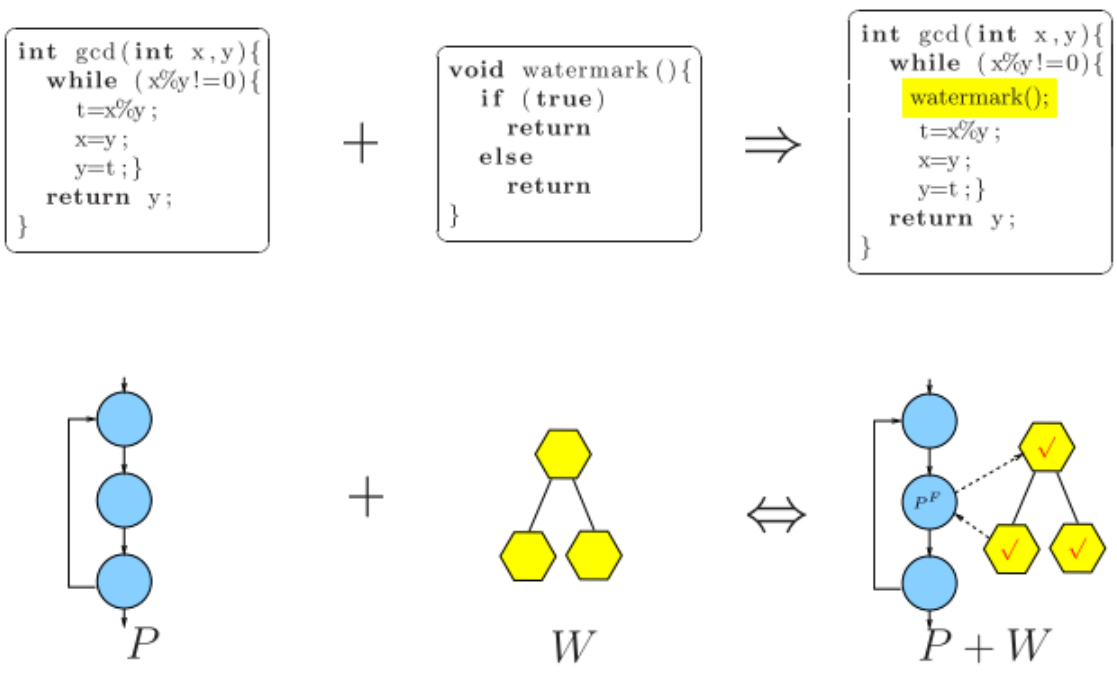
Τα μοντέλα υδατογράφησης λογισμικού ανάλογα με τις συνιστώσες που αποτελούν τη δομή του υδατογραφήματος και τους αλγορίθμους ενσωμάτωσης και εξαγωγής που χρησιμοποιούν, χαρακτηρίζονται από ιδιότητες οι οποίες τους δίνουν μοναδική ταυτότητα. Οι ιδιότητες αυτές αυτομάτως δίνουν στα μοντέλα υδατογράφησης τόσο πλεονεκτήματα, όσο και μειονεκτήματα. Παρακάτω, παρουσιάζονται αλγόριθμοι από μία αφαιρετική οπτική γωνία, παρατηρήσεις, ιδιότητες καθώς και πλεονεκτήματα και μειονεκτήματα από τρία γνωστά μοντέλα υδατογράφησης λογισμικού σύμφωνα με τη χρονολογία με την οποία και αυτά δημοσιεύτηκαν.

5.1.1 Μοντέλο 1

Το πρώτο κατά σειρά μοντέλο παρουσιάστηκε για πρώτη φορά το έτος 2004 στην εργασία με τίτλο: Graph theoretic software watermarks: implementation, analysis and attacks από τους συγγραφείς C. Collberg, A. Huntwork, E. Carter και G. Townsend. Η εργασία τους αυτή είχε βέβαια τις βάσεις της σε μία προγενέστερή της, του έτους 2001, με τίτλο: A graph theoretic approach to software watermarking από τους συγγραφείς R. Venkatesan, V. Vazirani και S. Sinha από την οποία και υιοθέτησαν τη βασική ιδέα επεκτήνοντάς την με έναν διαφορετικό τρόπο αυτούς προκειμένου να ξεπεράσουν ορισμένα σοβαρά μειονεκτήματα κακής διαφάνειας. Παρακάτω παρατίθενται τα στάδια του αλγορίθμου ενσωμάτωσης υδατογραφήματος σε κώδικα λογισμικού.

1. Η τιμή του υδατογραφήματος v διαμερίζεται σε k τιμές $v_0, v_1, \ldots, v_{k-1}$.
2. Οι k τιμές κωδικοποιούνται σε κατευθυνόμενα γραφήματα $G_0, G_1, \ldots, G_{k-1}$ (RPGs).
3. Τα RPGs τροποποιούνται σε διαγράμματα ελέγχου ροής $W_0, W_1, \ldots, W_{k-1}$ (CFGs).
4. Οι συναρτήσεις του προγράμματος P χωρίζεται σε συστάδες.
5. Τα CFGs ενσωματώνονται στο P προσθέτοντας κλήσεις συναρτήσεων.
6. Χρησιμοποιούνται σημεία σήμανσης για κάθε ένα από τα CFGs που ενσωματώνεται.

Σύμφωνα με τα στάδια αυτά του αλγορίθμου ενσωμάτωσης κάθε συστάδα περιέχει συναρτήσεις (κόμβους) είτε πραγματικές, είτε κίβδηλες. Οι συναρτήσεις αυτές θεωρούνται συνδεδεμένες μεταξύ τους αν υπάρχει κλήση (ακμή) από μία συνάρτηση σε μία άλλη. Επίσης, η κατασκευή των συστάδων που δημιουργούνται στο στάδιο 4 έχει ως κύριο στόχο να ελαχιστοποιηθούν οι συνδέσεις (κλήσεις συναρτήσεων) μεταξύ των συστάδων για να μην δίνεται στόχος σε κακόβουλους χρήστες.



Σχήμα 5.1: Γραφοθεωριτικό υδατογράφημα.

Επιπλέον, η τεχνική για τα σημεία σήμανσης που χρησιμοποιεί ο αλγόριθμος στο στάδιο 6 είναι ονομάζεται checksum. Η τεχνική αυτή υπολογίζει το MD5 digest για κάθε ένα από τα τμήμα (block) των κίβδηλων συναρτήσεων. Συγκεκριμένα σύμφωνα με τον αλγόριθμο αυτόν, ένα τμήμα θεωρείται μαρκαρισμένο μονάχα αν τα 2 bits τελευταίας προτεραιότητας του είναι μηδενικά. Παρακάτω παρουσιάζονται και οι ιδιότητες του μοντέλου αυτού.

- Η υδατογράφηση γίνεται σε επίπεδο πηγαίου κώδικα.
- Κάνει στατική υδατογράφηση κώδικα λογισμικού.
- Το υδατογράφημα που ενσωματώνει ανήκει στα υδατογραφήματα κώδικα (code watermarks).
- Ενσωματώνει πολλά RPGs για ένα υδατογράφημα.
- Προσθέτει κλήσεις μεθόδων μεταξύ των συστάδων χρησιμοποιώντας την τεχνική του τυχαίου περιπάτου (random walk).
- Για κάθε CFGs που προσθέτει, δημιουργεί μία συνάρτηση η οποία παίρνει ως όρισμα έναν ακέραιο αριθμό και επιστρέφει έναν ακέραιο αριθμό.
- Κάθε συστάδα εμπεριέχει κόμβους οι οποίοι είναι είτε πραγματικές, είτε κίβδηλες συναρτήσεις.
- Κάνει ομαδοποιήσεις προτού αρχίσει να προσθέτει κλήσεις συναρτήσεων προκειμένου να αυξήσει την πολυπλοκότητα των γραφημάτων στα οποία προστίθενται οι κλήσεις αυτές. Αν δεν έκανε τη λειτουργία αυτή θα δημιουργείτο κόμβοι με περισσότερες από 2 εξερχόμενες ακμές ή κόμβοι με μικρό αριθμό εισερχόμενων ακμών με αποτέλεσμα να δίνει στόχο το υδατογράφημα.
- Μέσα στην κίβδηλη συνάρτηση, ενσωματώνονται μικρά τμήματα κώδικα τα οποία αντιστοιχούν στους κόμβους του ενός από τα RPG και ενσωματώνονται επίσης εντολές όπως conditional jumps ή fall through paths οι οποίες αντιστοιχούν στις ακμές αυτού του RPG.

Τα πλεονεκτήματα του εν λόγω μοντέλου είναι πως δεν ενσωματώνει ανενεργό κώδικα (dead code) ο οποίος θα μπορούσε να αφαιρεθεί πολύ εύκολα. Επίσης, αν γίνουν τροποποιήσεις στο υδατογράφημα, κάνοντας χρήση error - correcting ιδιοτήτων που φέρει η δομή του υδατογραφήματος οι τροποποιήσεις αυτές μπορούν να εντοπιστούν με μεγάλη πιθανότητα ή ακόμα και να διορθωθούν. Τέλος μπορεί και αντιστέκεται σε επιθέσεις όπως inlining, register re-allocation, local variable merging, array splitting, class inheritance modi?cation, local variable splitting.

Από την άλλη πλευρά, τα μειονεκτήματα είναι πως ενσωματώνει κίβδηλο κώδικα (dummy code) και πως χρησιμοποιεί τεχνικές για σημεία σήμανσης. Επιλέον, το μοντέλο αυτό ενσωματώνει RPGs μέσα στον κώδικα τα οποία εκτελούνται με περισσότερες από μία ακολουθίες εισόδου. Τα γραφήματα αυτά όμως εκ κατασκευής τους, μιμούνται τα διαγράμματα ελέγχου ροής. Από την στιγμή όμως, που τις αυστηρές ιδιότητες αυτών σπανίως χαρακτηρίζουν τα συνήθη διαγράμματα ελέγχου ροής των εκάστοτε λογισμικών, καθώς μοιάζουν σε μεγάλο βαθμό, αλλά δεν φέρουν τις ιδιότητες αυτών, να δίνουν εύκολα στόχο σε κακόβουλους χρήστες.

Επίσης, ενσωματώνει συναρτήσεις οι οποίες δεν είναι διαφανείς (stealthy). Το γεγονός αυτό είναι αληθές καθώς οι εντολές σε επίπεδο bytecode των επιπρόσθετων μεθόδων περιέχουν 20% αριθμητικές εντολές, ενώ στη γενική περίπτωση των προγραμμάτων σε γλώσσα προγραμματισμού Java οι μέθοδοι σε επίπεδο bytecode περιέχουν μονάχα 1% αριθμητικές εντολές, πράγμα το οποίο δίνει ερεθίσματα σε κακόβουλους χρήστες. Οι κλήσεις μεταξύ των κίβδηλων μεθόδων και των πραγματικών ποτέ δεν εκτελούνται, ενώ δεν αντιστέκεται σε επιθέσεις όπως primitive boxing, basic block splitting, method merging, class encryption, code duplication.

5.1.2 Μοντέλο 2

Το δεύτερο κατά σειρά μοντέλο παρουσιάστηκε για πρώτη φορά το έτος 2004 στην εργασία με τίτλο: Dynamic path based software watermarking από τους συγγραφείς C. Collberg, E. Carter, S. Debray, A. Huntwork, J. Kececioglu και C. Linn. Η εργασία αυτή, επί της ουσίας δεν στηρίχτηκε σε κάποια προγενέστερή της, αλλά εισήγαγε για πρώτη φορά την υδατογράφηση λογισμικού μέσω δυναμικού μονοπατιού εκτέλεσης το οποίο κατασκευάζεται με την βοήθεια μίας επιπρόσθετης συνάρτησης η οποία χειραγωγεί τα μονοπάτια εκτέλεσης. Παρακάτω παρατίθενται τα στάδια του αλγορίθμου ενσωμάτωσης υδατογραφήματος σε κώδικα λογισμικού.

1. Η τιμή του υδατογραφήματος v διαμερίζεται σε k δυαδικές συμβολοσειρές (bit strings) $bt_0, bt_1, \ldots, bt_{k-1}$.

2. Για κάθε ένα από τα k bit strings κατασκευάζεται μία λίστα από $k+1$ branch function call $(a_0, a_1, \ldots, a_k)$ έτσι ώστε:

   - Για κάθε a_i η αμέσως προηγούμενη εντολή είναι ένα unconditional jump, έτσι ώστε η ροή εκτέλεσης να μην μπορεί να διέλθει από το a_i.
   - Οι διευθύνσεις από 2 γειτονικές εντολές $(ai, ai + 1)$ κωδικοποιούν το bit 0 αν $address(a_i) < address(a_{i+1})$ και το bit 1 αν $address(a_i) > address(a_{i+1})$.

3. Η ενσωμάτωση ξεκινά από ένα unconditional control flow edge $l_{begin} \rightarrow l_{end}$.

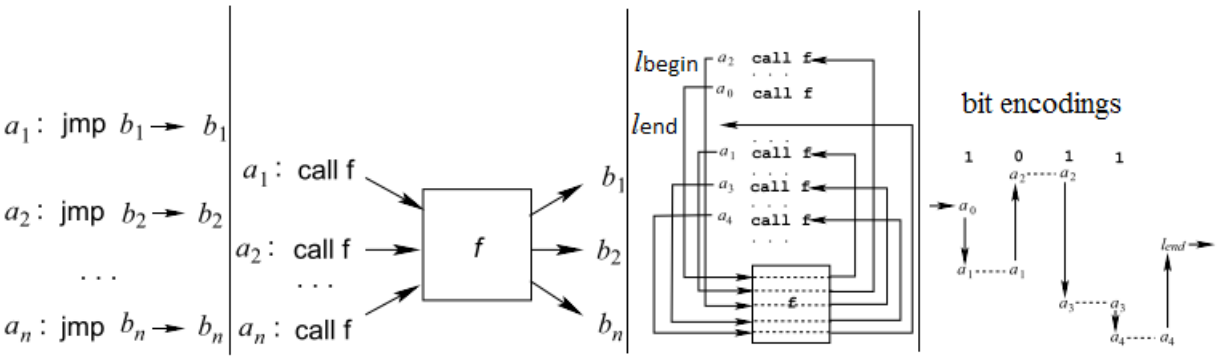
4. Για να κατασκευαστεί η λίστα, το a_0 εισέρχεται στη διεύθυνση l_{begin}, ύστερα επαναληπτικά κατασκευάζεται το a_{i+1} από την τελευταία εντολή ai που προστέθηκε στη λίστα:

   - Χρησιμοποιείται η τιμή bt_i για να σαρωθεί είτε forward (if $bt_i = 1$ need jump forward) είτε backward (if $bt_i = 0$ need jump backward) μέχρις ότου να βρεθεί σημείο το οποίο να ικανοποιεί την συνθήκη αυτή και να προστεθεί μία κλήση.
   - Επαναλαμβάνεται η ίδια ακριβώς διαδικασία μέχρις ότου να κατασκευαστούν όλες οι $a_0, a_1, \ldots, a_k$.

Σύμφωνα με τα στάδια αυτά, η ενσωμάτωση των δυαδικών συμβολοσειρών ξεκινά από ένα τυχαίο σημείο το οποίο δεν εμπεριέχεται σε συνθήκη ελέγχου. Επίσης, κατά την διάρκεια ενσωμάτωσης μίας δυαδικής συμβολοσειράς αντικαθίστανται όλες οι εντολές jump του εκτελέσιμου κώδικα, με εντολές invoke σε μία κίβδηλη συνάρτηση η οποία είναι και αυτή που θα επιτελέσει το ρόλο του jump στη συνέχεια, ανάλογα με το όρισμα που είχε το invoke, προκειμένου να χειραγωγηθεί ο τρόπος με τον οποίον στο Βήμα 4 πραγματοποιούνται τα forward και τα backward jumps. Παρακάτω, παρουσιάζονται και οι ιδιότητες του μοντέλου αυτού.

- Η υδατογράφηση γίνεται σε επίπεδο εκτελέσιμου κώδικα.
- Κάνει δυναμική υδατογράφηση κώδικα λογισμικού.
- Το υδατογράφημα που ενσωματώνει ανήκει στα δυναμικά υδατογραφήματα δομών δεδομένων (dynamic data structure watermarks).
- Ενσωματώνει πολλαπλά bit strings με την μορφή διακλαδώσεων.
- Δημιουργεί μία συνάρτηση f η οποία μετατρέπει τις εντολές jump σε κλήσεις στην f και στη συνέχεια η f πραγματοποιεί τα jump.
- Ανάλογα με τον αν το jump είναι forward ή backward (σε μεγαλύτερη ή μικρότερη διεύθυνση μνήμης) κωδικοποιείται το 1 ή το 0.

Το βασικό πλεονέκτημα του μοντέλου αυτού είναι πως δεν απαιτεί να εντοπίσει όλα τα bit strings που ενσωματώθηκαν για την εξαγωγή του υδατογραφήματος. Επίσης, ενσωματώνοντας διακλαδώσεις (branches) μπορούν εύκολα να κωδικοποιηθούν bits, καθώς το branch είναι εκ κατασκευής του δυαδικό. Αξίζει να σημειωθεί πως στις διακλαδώσεις που προστίθενται το μοντέλο αυτό μπορεί και χρησιμοποιεί μεταβλητές του αρχικού κώδικα και πως δεν ενσωματώνει ανενεργό κώδικα στον ήδη υπάρχον.



Σχήμα 5.2: Υδατογράφημα δυναμικού μονοπατιού.

Από την άλλη πλευρά τα μειονεκτήματα είναι πως, ενσωματώνει υδατογραφήματα προσθέτοντας κώδικα, με loop code generation και με condition code generation, εντός των branches ο οποίος είναι κίβδηλος. Το μοντέλο αυτό από τη στιγμή που έχει ως βάση του την εισαγωγή branches ύστερα από την εφαρμογή υδατογράφησης, στα σημεία του κώδικα που έχουν γίνει αλλαγές, υπάρχουν πολύ περισσότερα branches (συνθήκες ελέγχου ή / και δομές επανάληψης) σε αριθμό από ότι στον υπόλοιπο κώδικα. Το μοντέλο αυτό, μονάχα με τις τεχνικές error - correction και tamperproofing μπορεί και ενσωματώνει υδατογράφημα το οποίο να αντιστέκεται σε επιθέσεις όπως basic block reordering, branch chaining, loop unrolling.

Επίσης, το μοντέλο αυτό κωδικοποιεί μεγάλες σταθερές στον κώδικα του λογισμικού σε Java οι οποίες είναι καθ όλα μη διαφανείς (unstealthy), καθώς στο 93% των λογισμικών σε Java οι τιμές των σταθερών που χρησιμοποιούνται είναι μικρότερες από το 1000. Η διαδικασία εξαγωγής του υδατογραφήματος είναι χρονοβόρα διαδικασία ιδιαίτερα σε μεγάλης έκτασης εκτελέσεις καθώς πρέπει να επεξεργαστεί κάθε ένα από τα τμήματα του κώδικα (blocks) που εκτελούνται. Επιπλέον, αν προστεθούν, αφαιρεθούν, ή ακόμα τροποποιηθούν branches το υδατογράφημα αλλοιώνεται, ενώ η λειτουργικότητα του κώδικα παραμένει. Το υδατογράφημα που ενσωματώνεται με το μοντέλο αυτό δεν αντιστέκεται σε καμία τροποποίηση σημασιολογικών δομών αν και είναι δυναμικό και τέλος πως το υδατογράφημα που ενσωματώνεται με το μοντέλο αυτό αλλοιώνεται εύκολα από επιθέσεις όπως bogus branches, reordering non depended branches, split or merge branches.

5.1.3 Μοντέλο 3

Το τρίτο κατά σειρά μοντέλο, είναι το προτεινόμενο από την παρούσα διατριβή μοντέλο, το οποίο παρουσιάστηκε για πρώτη φορά το έτος 2013 στην εργασία με τίτλο: A dynamic watermarking model for embedding reducible permutation graphs into software από τους συγγραφείς I. Chionis, M. Chroni και S.D. Nikolopoulos. Η εργασία αυτή, είχε τις βάσεις της σε μία προγενέστερή της με τίτλο: An embedding graph-based model for software watermarking από τους M. Chroni και S.D. Nikolopoulos. Παρακάτω παρατίθενται τα στάδια του αλγορίθμου ενσωμάτωσης υδατογραφήματος σε κώδικα λογισμικού.

1. Η τιμή υδατογράφησης v κωδικοποιείται σε μια αυτοαναστρέφουσα μετάθεση k στοιχείων $(v_0, v_1, \ldots, v_{k-1})$.

2. Η αυτοαναστρέφουσα μετάθεση τροποποιείται σε ένα κατευθυνόμενο γράφημα RPG.

3. Εκτελείται το πρόγραμμα P με είσοδο I παράγοντας δυναμικό γράφημα κλήσεων DCG.

4. Γίνεται ένα προς ένα αντιστοιχία των συναρτήσεων που εκτελέστηκαν με όλους τους κόμβους του RPG.

5. Για κάθε μία κλήση συνάρτησης $call(f_i, f_j)$ που πραγματοποιήθηκε στο DCG:

- Αν αντιστοιχεί σε ακμή του RPG διατηρείται.
- Αλλιώς, αφαιρείται και αντικαθίσταται από το ελάχιστο μονοπάτι μεταξύ των κόμβων που έχουν αντιστοιχηθεί με τις συναρτήσεις f_i, f_j.

Σύμφωνα με αυτά τα στάδια του αλγορίθμου, οι κόμβοι του RPG στο στάδιο 4 πρέπει να είναι σε αριθμό ίσοι ή λιγότεροι από τις συναρτήσεις του κώδικα που εκτελέστηκαν με την ακολουθία εισόδου I. Επιπλέον, οι κλήσεις συναρτήσεων στο στάδιο 5 εισέρχονται εντός συνθηκών ελέγχου οι οποίες από τις εκφράσεις που αποτιμούν επιτρέπουν ή όχι την ροή εκτέλεσης να διέλθει μέσα από αυτά τα blocks προκειμένου να διατηρηθεί η λειτουργικότητα του κώδικα. Μόνον όταν η κλήση συνάρτησης σε μία συγκεκριμένη στιγμή κατά τη διάρκεια εκτέλεσης του κώδικα ανήκει σε ένα ελάχιστο μονοπάτι η ροή εκτέλεσης διέρχεται εντός της συνθήκης ελέγχου εκτελώντας τον εσωτερικό κώδικά της. Παρακάτω παρουσιάζονται και οι ιδιότητες του μοντέλου αυτού.

- Η υδατογράφηση γίνεται σε επίπεδο πηγαίου κώδικα.
- Κάνει δυναμική υδατογράφηση κώδικα λογισμικού.
- Το υδατογράφημα που ενσωματώνει ανήκει στα δυναμικού ίχνους εκτέλεσης υδατογραφήματα (dynamic execution trace watermarks).
- Ενσωματώνει ένα RPG για ένα υδατογράφημα.
- Κωδικοποιεί το υδατογράφημα στο δυναμικό γράφημα κλήσεων που δημιουργείται κατά την διάρκεια εκτέλεσης του κώδικα.
- Το μοντέλο στο στάδιο 5 του αλγορίθμου χρησιμοποιεί πρότυπα τα οποία περιέχουν κλήσεις σε συναρτήσεις, συνθήκες ελέγχου και αναθέσεις τιμών σε μεταβλητές.
- Χειραγωγεί τη ροή εκτέλεσης του κώδικα, προκειμένου να παραμείνει λειτουργικός με προσθήκη ελέγχων.

Τα πλεονεκτήματα του εν λόγω μοντέλου είναι πως στις συνθήκες ελέγχου που προσθέτει, οι οποίες αποτιμούν αδιαφανή κατηγορήματα (opaque predicates), μπορεί και χρησιμοποιεί και μεταβλητές αρχικές του κώδικα. Το μοντέλο αυτό δεν χρησιμοποιεί σημεία σήμανσης καθώς επίσης δεν ενσωματώνει ούτε ανενεργό, ούτε κίβδηλο κώδικα. Απεναντίας αυτό που κάνει είναι να τροποποιεί τον ήδη υπάρχοντα.

Αν γίνουν αλλαγές στο υδατογράφημα κάνοντας χρήση error ? correcting ιδιοτήτων οι αλλαγές αυτές μπορούν να εντοπιστούν με αρκετά μεγάλη πιθανότητα (πιο μεγάλη από την πρώτη εργασία του C. Collberg) ή ακόμα και να διορθωθούν. Αν τροποποιηθεί ή διαγραφεί κάποια ανάθεση στις μεταβλητές υδατογράφησης ο κώδικας παύει να είναι λειτουργικός. Προκειμένου όμως να λειτουργεί και πάλι ορθά πρέπει να γίνουν αλλαγές σε όλες τις αναθέσεις στις μεταβλητές αυτές από το σημείο που έγινε η τροποποίηση και έπειτα στην εκτέλεση, καθώς επίσης και στις συνθήκες ελέγχου οι οποίες αποτιμούσαν τις τιμές αυτές. Παρ όλα αυτά το ενσωματωμένο υδατογράφημα δεν θα αλλοιωθεί.

Επιπλέον, αν διαγραφεί κάποια κλήση η οποία λάμβανε χώρα σε κάποιο από τα ελάχιστα μονοπάτια, ο κώδικας παύει να είναι λειτουργικός. Ενώ τέλος, το μοντέλο αυτό ενσωματώνει υδατογράφημα το οποίο αντιστέκεται σε επιθέσεις όπως: decompile ? compile, layout obfuscations (όλες), data obfuscations (όλες), control obfuscations (ordering, computation), shrinking, code optimizations (όπως branch optimization, code motion, common sub expression elimination, global register allocation).

Από την άλλη πλευρά τα μειονεκτήματα είναι πως, το μοντέλο αυτό αντικαθιστώντας μία κλήση συνάρτησης με το αντίστοιχο shortest path (αλληλουχία κλήσεων) χρησιμοποιεί patterns τα οποία εμπεριέχουν συνθήκες ελέγχου με πολλαπλές expressions, με αποτέλεσμα στο εκτελέσιμο αρχείο για κάθε μία από αυτές τις expressions αυτές να δημιουργείται και μία εντελή ελέγχου (π.χ. στο Java Bytecode να δημιουργούνται πολλές εντολές if_icmpeg) αυξάνοντας έτσι αρκετά το πλήθος των εντολών αυτών.

Επίσης αν διαγραφεί κάποια κλήση η οποία δεν λάμβανε χώρα σε κάποιο από τα ελάχιστα μονοπάτια, ο κώδικας παύει να είναι λειτουργικός. Προκειμένου όμως να λειτουργεί και πάλι ορθά πρέπει να γίνουν οι κατάλληλες αλλαγές (ολίσθηση προς τα πίσω τόσο όσο θα προστίθονταν στη μεταβλητή υδατογράφησης αν πραγματοποιούνταν η κλήση που διαγράφτηκε) στις συνθήκες ελέγχου από εκείνο το σημείο όπου έγινε η αλλαγή και έπειτα στη διάρκεια εκτέλεσης. Τέλος, το μοντέλο αυτό ενσωματώνει υδατογράφημα το οποίο δεν αντιστέκεται σε επιθέσεις όπως control obfuscations (aggregation such as inline, outline and clone methods).

5.2 Συγκρίσεις Ανθεκτικότητας

Με την αναλυτική παρουσίαση των τριών προηγούμενων μοντέλων υδατογράφησης, γεννιούνται πολλά ερωτήματα σχετικά με το πως αυτά ανταποκρίνονται σε επιθέσεις όπως είναι οι τροποποιήσεις μέσω συσκότισης και οι τροποποιήσεις μέσω βελτιστοποίησης κώδικα. Επί της ουσίας δηλαδή, αν ύστερα από μία ορισμένη επίθεση το ενσωματωμένο υδατογράφημα μπορεί να διατηρείται αναλλοίωτο ή τουλάχιστον μπορεί να ανακατασκευαστεί.

Ορισμένες από τις τεχνικές συσκότισης και βελτιστοποίησης στο υδατογραφημένα λογισμικά έλαβαν χώρα μέσω του αυτόματου εργαλείου ProGaurd [54], ενώ οι υπόλοιπες εξετάστηκαν βάσει των ιδιοτήτων που χαρακτήριζαν τα μοντέλα. Το εργαλείο αυτό, επιτελεί λειτουργίες κυρίως συσκότισης (obfuscation) και βελτιστοποίησης (optimization) προκειμένου να δημιουργήσει πιο συμπυκνωμένο κώδικα, γρηγορότερα εκτελέσιμο και δύσκολα διαχειρίσιμο στην αντίστροφη μηχανική. Αφού ο κώδικας ο υδατογραφημένος περνούσε μέσα από το εργαλείο ProGaurd, ύστερα για να αποδειχθεί αν το ενσωματωμένο υδατογράφημα δέχονταν αλλοίωση ή όχι ο κώδικας περνούσε μέσα από ένα άλλο αυτόματο εργαλείο ονόματι Java Decompiler με σκοπό να γυρίσει ο εκτελέσιμος κώδικας πίσω στον πηγαίο κώδικα (αν αυτό ήταν εφικτό) και στη συνέχεια ο πηγαίος να μπει και να εκτελεστεί μέσα από τον παρακολουθητή (profiler) του NetBeans για να καταγραφεί το ίχνος εκτέλεσης και να εξαχθεί το υδατογράφημα [41].

5.2.1 Ανθεκτικότητα σε Συσκότιση Κώδικα

Η τεχνική της συσκότισης κώδικα λογισμικού χωρίζεται σε ορισμένες βασικές κατηγορίες οι οποίες περιλαμβάνουν ενέργειες που ανάλογα με τα στοιχεία του κώδικα που χειρίζονται και επεξεργάζονται κατατάσσονται και στην αντίστοιχη κατηγορία. Οι κατηγορίες αυτές βρίσκουν ευρεία εφαρμογή ιδιαίτερα στις περιπτώσεις όπου είναι επιθυμητή η στρέβλωση είτε μικρή, είτε μεγάλη του κώδικα προς αποφυγή κατανόησής του από ενδεχόμενους κακόβουλους χρήστες. Αυτές είναι οι εξής:

- Συσκότιση διάταξης (layout obfuscation)
- Συσκότιση δεδομένων (data obfuscation)
- Συσκότιση ελέγχου (control obfuscation)

Κάθε μία από τις κατηγορίες αυτές επικεντρώνεται στο δικό της κλοιό ή αλλιώς φάσμα υποψήφιων προς συσκότιση στοιχείων του κώδικα, πάντα υπό την προϋπόθεση πως όποια αλλαγή και να πραγματοποιηθεί δεν θα δημιουργεί αντίκτυπο στην λειτουργικότητα του λογισμικού. Οι πιο απλές και σχετικά με τις υπόλοιπες γρήγορες στην εφαρμογή τους τεχνικές συσκότισης κατατάσσονται στις συσκοτίσεις διάταξης όπου τροποποιούνται πληροφορίες οι οποίες δεν σχετίζονται με την εκτέλεση του κώδικα, οι αμέσως πιο περίπλοκες είναι οι τεχνικές συσκότισης δεδομένων όπου με αυτές τροποποιούνται οι δομές που χειρίζονται τις πληροφορίες του κώδικα και οι περιπλοκότερες είναι οι τεχνικές συσκότισης ελέγχου όπου με αυτές χειραγωγείται η ροή εκτέλεσης και πιο συγκεκριμένα αλλάζουν τα διαγράμματα ελέγχου ροής του κώδικα.

Ο παρακάτω πίνακας εμπεριέχει τις κατηγορίες συσκότισης, οι οποίες χωρίζονται εν συνεχεία σε επιμέρους υποκατηγορίες καθώς και τα τρία μοντέλα με τη σειρά που παρουσιάστηκαν στο προηγούμενο υποκεφάλαιο του τρέχοντος κεφαλαίου. Από τον εν λόγω πίνακα διακρίνονται ποια από τα μοντέλα ενσωματώνουν δομές υδατογράφησης σε κώδικες λογισμικών οι οποίες είναι ανθεκτικές σε ποιες κατηγορίες συσκοτίσεων.

Από τα αποτελέσματα αυτά, εύκολα μπορεί να γίνει ευδιάκριτο πως τις περισσότερες των επιθέσεων μπορεί και τις αντιστέκεται το μοντέλο WaterRPG, έχοντας ως μοναδικό αδύνατο σημείο τις συγχωνεύσεις και τους διαχωρισμούς των συναρτήσεων, ενώ τις αμέσως επόμενες επιθέσεις μπορεί και τις αντιστέκεται το GTW_{SM} παρόλο που χαρακτηρίζεται από την στατική ιδιότητα. Το μοντέλο με τις λιγότερες αντιστάσεις σε συσκοτίσεις κώδικα είναι το Dynamic Path Based το οποίο φαίνεται ιδιαίτερα ευαίσθητο σε τροποποιήσεις συνθηκών ελέγχου. Ίσως όμως, ένα από τα μεγαλύτερα του πλεονεκτήματα του Dynamic Path Based μοντέλου είναι πως δεν είναι απαραίτητα να γίνει εξαγωγή όλου του κώδικα που έχει χρησιμοποιηθεί για την ενσωμάτωση του υδατογραφήματος. Πράγμα, το οποίο για τα υπόλοιπα δύο μοντέλα δεν ισχύει.

Μέχρι στιγμής, δεν έχει βρεθεί κάποιο μοντέλο υδατογράφησης λογισμικού το οποίο να καθίσταται πλήρως ανθεκτικό σε συσκοτίσεις κώδικα. Κάτι τέτοιο θα αποτελούσε αμέσως σταθμό στο τομέα υδατογράφησης λογισμικού. Πάρα ταύτα το προτεινόμενο μοντέλο της εργασίας αυτής βρίσκεται πολύ κοντά σε μία τέτοιου είδους επίτευξη. Το αδύνατό του

σημείο, θωρακίζοντας το μοντέλο με νέες λειτουργίες κατά την λειτουργία ενσωμάτωσης, πιθανότατα να μπορεί να εξαλειφθεί, ξεπερνώντας αναίμακτα με τον τρόπο αυτόν ακόμη και τις επιθέσεις συγχώνευσης και διαχωρισμού.

Πίνακας 5.1: Ανθεκτικότητα σε τεχνικές συσκότισης των μοντέλων 1, 2 και 3

Κύριες Κατηγορίες	**Δευτερεύουσες Κατηγορίες**	**Επιμέρους Υποκατηγορίες**	1	2	3
layout	scramble ids	reuse identifiers for several entities	✓	✓	✓
		introduce syntactic and semantic errors	✓	✓	✓
	change format	–	✓	✗	✓
	dead code el.	–	✓	✓	✓
	dummy code el.	–	✓	✗	✓
data	storage	split variables	✓	✗	✓
		promote scales to object	✓	✓	✓
		convert static data to procedure	✗	✗	✓
		register reassignment	✓	✓	✓
	encoding	change encoding	✗	✗	✓
		change variable lifetime	✓	✓	✓
	aggregation	merge scalar variables	✓	✓	✓
		modify inheritance relations	✓	✓	✓
		split – fold – merge – flattening arrays	✓	✗	✓
		instruction substitution	✗	✓	✓
	ordering	reorder instance variables	✓	✗	✓
		reorder methods	✓	✓	✓
		reorder arrays	✓	✗	✓
control	computation	reducible to non reducible flow graphs	✗	✓	✓
		extend loop condition	✗	✗	✓
		table interpretation	✓	✓	✓
		control flow transformations	✗	✗	✓
		data flow analysis	✗	✗	✓
	aggregation	inline methods	✓	✗	✗
		outline methods	✗	✗	✗
		split – merge methods	✗	✗	✗
		clone methods	✗	✗	✗
		unroll loops	✗	✗	✓
	ordering	reorder statements	✗	✗	✓
		reorder loops	✗	✗	✓
		reorder expressions	✗	✗	✓

5.2.2 Ανθεκτικότητα σε Βελτιστοποίηση Κώδικα

Η τεχνική της βελτιστοποίησης κώδικα όπως και η τεχνική της συσκότισης, χωρίζεται σε επιμέρους κατηγορίες οι οποίες περιλαμβάνουν λειτουργίες όπου ανάλογα με τα εκάστοτε στοιχεία που τροποποιούν κατατάσσονται και στην αντίστοιχη κατηγορία. Υπενθυμίζεται, πως η συσκότιση κώδικα έχει ως κύριο στόχο την στρέβλωση του κώδικα, ενώ η βελτιστοποίηση έχει ως κύριο στόχο την ταχύτητα εκτέλεσης. Οι τεχνικές βελτιστοποίησης κώδικα, συνήθως εφαρμόζονται από μεταγλωττιστές ή διερμηνείς γλωσσών προγραμματισμού με σκοπό να κερδίσουν σε χώρο και σε χρόνο εκτέλεσης.

Το γεγονός αυτό, έμμεσα κρύβει το ότι οι λειτουργίες αυτής της τεχνικής λαμβάνουν χώρα στο εκτελέσιμο αρχείο και μόνο κατά την διάρκεια εκτέλεσής του, επηρεάζοντας το ίχνος που δημιουργεί η εκτέλεσης ενός λογισμικού (π.χ. ποια ενδιάμεσα αποτελέσματα φυλάσσονται σε ποιους καταχωρητές). Αν και αυτό, δείχνει να είναι καταστροφικό για το μοντέλο της παρούσας διατριβής, καθώς ενδεχομένως να πειράζονται οι κλήσεις συναρτήσεων, όπως θα φανεί στη συνέχεια αυτό δεν ισχύει, διότι οι επιπρόσθετες κλήσεις που έχουν προστεθεί έχουν αντικαταστήσει παλιές και με τον τρόπο αυτόν έχουν γίνει αναπόσπαστο κομμάτι της λειτουργικότητας του κώδικα.

Τα δύο πρώτα μοντέλα, επειδή έχουν προσθέσει αμφότερα κίβδηλο κώδικα μέσα στον ήδη υπάρχον τον οποίο τον έχουν περιλάβει εντός συνθηκών ελέγχου εύκολα αλλαγές στις συνθήκες αυτές, είτε στις εκφράσεις που αποτιμούν, είτε στις εσωτερικές τους εντολές αλλοιώνουν τη δομή του υδατογραφήματος δίχως να δημιουργούν επιπτώσεις λειτουργικότητας στην υπόλοιπο κώδικα. Οι αλλοιώσεις αυτές όμως δεν φέρουν το ίδιο αντίκτυπο στο WaterRPG μοντέλο το οποίο και αυτό χρησιμοποιεί συνθήκες ελέγχου αλλά με τέτοιον τρόπο ούτως ώστε να γίνουν κτήμα του πραγματικού κώδικα και να παίζουν καθοριστικό ρόλο στη λειτουργικότητά του.

Επιπλέον, τροποποιήσεις σε δεδομένα όπως είναι οι εκφράσεις, είτε αυτές είναι μαθηματικές, είτε είναι λογικές προκειμένου το πρόγραμμα να εκτελείται ταχύτερα ενώ δημιουργούν προβλήματα αναγνώρισης του υδατογραφήματος στα δύο πρώτα μοντέλα, δεν συμβαίνει το ίδιο και στο μοντέλο που προτείνεται από την παρούσα εργασία. Αυτό είναι αλήθεια, διότι τα δύο πρώτα μοντέλα, στηρίζονται στον κώδικα σε πρώτο επίπεδο και όποιες επιθέσεις και να λάβουν χώρα στο επίπεδο αυτό είναι καταστροφικές για αυτά. Με τον όρο πρώτο επίπεδο εννοείται τα συστατικά που αποτελούν τον κώδικα. Ενώ, το τρίτο μοντέλο το οποίο δεν στηρίζεται στο πρώτο αλλά στο δεύτερο επίπεδο του κώδικα που είναι η ροή εκτέλεσης και συγκεκριμένα η ροή που έχει ο κώδικας διερχόμενος από τις συναρτήσεις του κώδικα, δεν δημιουργούν προβλήματα εντοπισμού του υδατογραφήματος.

Παρακάτω παρουσιάζεται ένας παρόμοιος συγκριτικός πίνακας, με τον αμέσως προηγούμενο, των τριών μοντέλων που παρουσιάστηκαν στο προηγούμενο υποκεφάλαιο του τρέχοντος κεφαλαίου. Από τον εν λόγω πίνακα γίνεται διάκριση των μοντέλων αυτών σχετικά με το ποια από τα μοντέλα ενσωματώνουν υδατογραφήματα σε κώδικες λογισμικών τα οποία μπορούν να αντισταθούν σε ποιες κατηγορίες βελτιστοποίησης. Όπως απορρέει από τον εν λόγω πίνακα, μονάχα το μοντέλο WaterRPG αντιστέκεται στις περισσότερες βελτιστοποιήσεις κώδικα και αυτό διότι δεν στηρίζεται επιφανειακά στα στοιχεία του κώδικα

όπως κάνουν τα υπόλοιπα δύο μοντέλα, αλλά στην ροή των κλήσεων που πραγματοποιούνται, πράγμα το οποίο δύσκολα έως καθόλου αλλάζει ειδικά αν οι κλήσεις αυτές είναι πλέον λειτουργικό τμήμα του κώδικα.

Πίνακας 5.2: Ανθεκτικότητα σε τεχνικές βελτιστοποίησης των μοντέλων 1, 2 και 3

Κατηγορίες	1	2	3
branch optimization	✗	✗	✓
code motion	✓	✓	✓
common subexpression elimination	✗	✗	✓
algebraic simplification	✓	✓	✓
operator strength reduction	✓	✓	✓
constant propagation	✓	✓	✓
constant folding	✓	✗	✓
dead code elimination	✗	✗	✓
peephole optimizations	✓	✓	✓
dead store elimination	✗	✗	✓
global register allocation	✓	✓	✓
instruction scheduling	✗	✗	✓
interprocedural analysis	✗	✗	✗
invariant IF code floating (unswitching)	✗	✗	✓
reassociation	✓	✓	✓
store motion	✓	✓	✓
value numbering	✗	✗	✓

Κεφαλαιο 6

Συμπερασματα

6.1 Καταληκτικές Παρατηρήσεις

6.2 Μελλοντική Εργασία

6.1 Καταληκτικές Παρατηρήσεις

Ολοκληρώνοντας τη συγγραφή της προκειμένης διατριβής προέκυψαν ορισμένα σεβαστά συμπεράσματα, τόσο στον ευρύτερο τομέα της προστασίας λογισμικού, όσο και στον εξειδικευμένο τομέα της υδατογράφησης λογισμικού όπου προστατεύονται τα πνευματικά δικαιώματα.

Πρώτο γενικό συμπέρασμα είναι πως αν η προστασία που τίθεται σε ένα λογισμικό ξεφύγει από τα επιτρεπτά όρια (αυξηθεί σε μεγάλο βαθμό) τότε το λογισμικό παύει να θεωρείται ασφαλές καθώς θα έχει επιβαρυνθεί πολύ από τον επιπρόσθετο κώδικα ασφαλείας με αποτέλεσμα αυτό να προκαλεί πολλαπλά κενά. Το λογισμικό καθίσταται πλέον αργό, μεγάλο, δύστροπο στην συντήρησή του και οι κακόβουλοι χρήστες δεν χρειάζεται να καταβάλουν σημαντική προσπάθεια προκειμένου να αναιρέσουν την όποια προστασία του. Επομένως, η προστασία σε λογισμικά επιβάλλεται να είναι διακριτική και τόση όσο χρειάζεται δίχως υπερβολές.

Δεύτερο γενικό συμπέρασμα είναι πως αν οι εταιρίες οι οποίες παράγουν λογισμικά δεν σταματήσουν να τα διαθέτουν στην αγορά σε τόσο υψηλές τιμές, η πειρατεία των λογισμικών συνεχώς θα αυξάνεται. Το γεγονός αυτό είναι αλήθεια και ως τώρα απολύτως επιβεβαιωμένο καθώς οποιαδήποτε προστασία και να τεθεί στα λογισμικά είναι δεδομένο πως θα εμπεριέχει κενά ασφαλείας, με αποτέλεσμα αν κάποιος δεν επιθυμεί να αγοράσει ένα λογισμικό διότι η τιμή του, του το απαγορεύει τότε θα προσπαθήσει είτε να το αποκτήσει παράνομα αλλοιώνοντάς του την προστασία του ως πρωτογενής παράγοντας, είτε να το αποκτήσει στην πειρατική του μορφή μέσω άλλων ως δευτερογενής παράγοντας, συμβάλλοντας είτε με τον έναν είτε με τον άλλον τρόπο στην άνθηση της πειρατείας.

Τρίτο γενικό συμπέρασμα είναι πως προστασία λογισμικού μονάχα από μία οπτική γωνία (π.χ. της υδατογράφησης) σίγουρα δεν καθίσταται αρκετή προκειμένου να θέσει τις κατάλληλες δικλίδες ασφαλείας έναντι κακόβουλων χρηστών. Η ασφάλεια πρέπει να τίθεται με περισσότερους από έναν τρόπους, οι οποίοι θα είναι αλληλένδετοι και θα μπορούν να επικοινωνούν άμεσα (π.χ. υδατογράφηση, μαζί με θωράκιση και συσκότιση).

Τέταρτο γενικό συμπέρασμα είναι πως το νομοθετικό πλαίσιο γύρο από την προστασία λογισμικού στην Ελλάδα αλλά και σε πολλές χώρες στο εξωτερικό είναι πεπαλαιωμένο και μάλιστα σε αρκετές περιπτώσεις λάθος διαχειρίσιμο και χρήζει άμεσης ανανέωσης προκειμένου οι εταιρίες και γενικά οι πνευματικοί ιδιόκτητες λογισμικών να μπορούν να βρίσκουν το δίκιο τους προσφεύγοντας στα δικαστήρια.

Από την πλευρά της υδατογράφησης λογισμικού συγκεκριμένα, πρώτο συμπέρασμα είναι πως η υδατογράφηση ανεξάρτητα από το τι είδους πληροφορία ενσωματώνει στον κώδικα ενός λογισμικού δεν πρέπει σε καμία των περιπτώσεων να είναι ορατή και να επιβαρύνει το λογισμικό αισθητά τόσο σε χώρο όσο και σε χρόνο καθώς αν εντοπιστεί το υδατογράφημα από κακόβουλους χρήστες είναι βέβαιο πως έχει καταστραφεί.

Δεύτερο συμπέρασμα για τα υδατογραφήματα λογισμικού είναι πως αυτά πρέπει να μπορούν να εξάγονται από το λογισμικό στο οποίο έχουν ενσωματωθεί ακόμα και μετά το πέρας διαφόρων τροποποιήσεων. Όσες περισσότερες επιθέσεις είναι σε θέση το ενσωματωμένο υδατογράφημα να αποκρούει τόσο πιο αποτελεσματικό καθίσταται. Βέβαια, κανένα μοντέλο υδατογράφησης ως τώρα δεν μπορεί να αντισταθεί σε όλες τις ευρέως γνωστές τροποποιήσεις για αυτόν τον λόγο και η εν λόγω τεχνική δεν μπορεί να σταθεί από μόνη της προκειμένου να προστατεύσει πλήρως τα πνευματικά δικαιώματα σε λογισμικά.

Τρίτο συμπέρασμα είναι πως η δημιουργία και η ενσωμάτωση υδατογραφήματος σε λογισμικά είναι καθαρά λειτουργίες οι οποίες χρήζουν άπλετης φαντασίας προκειμένου να λάβουν υπόψη τους όλες τις πιθανές τροποποιήσεις οι οποίες θα λάβουν χώρα στον κώδικα του λογισμικού με απώτερο σκοπό να αλλοιώσουν την προστασία του. Για αυτό χρειάζονται διαύγεια και εμπειρία προκειμένου να φέρουν τα βέλτιστα αποτελέσματα.

Τέταρτο και τελευταίο συμπέρασμα είναι πως το μοντέλο υδατογράφησης λογισμικού είναι αναγκαίο να κάνει χρήση πραγματικού κώδικα (ήδη υπάρχοντος) προκειμένου να κωδικοποιήσει σε αυτόν πληροφορία στοχεύοντας με τον τρόπο αυτόν στην υδατογράφηση του κώδικα από το ίδια του στοιχεία, μην αφήνοντας τους κακόβουλους χρήστες να εκμεταλλευτούν το κενό που υπεισέρχεται για να μπορέσουν να κάνουν την διάκριση μεταξύ αυθεντικού κώδικα και κώδικα υδατογραφήματος.

6.2 Μελλοντική Εργασία

Δοθείσας αυτής της διατριβής, ως μελλοντική εργασία προκύπτουν αρκετές εναλλακτικές επιλογές. Μία από αυτές θα μπορούσε να είναι η ενσωμάτωση πολλαπλών υδατογραφημάτων στο λογισμικό χρησιμοποιώντας για κάθε έναν από αυτά βέβαια και διαφορετική ακολουθία εισόδου. Αλλη, είναι η περαιτέρω μελέτη του μοντέλου WaterRPG έναντι τροποποιήσεων οι οποίες θα μπορούσαν να λάβουν χώρα στον κώδικα του λογισμικού καθώς επίσης η

ενσωμάτωση υδατογραφημάτων στηριζόμενη στην ίδια ιδέα αλλά με λιγότερες επιπρόσθετες εντολές. Επίσης, ως μελλοντική εργασία θα μπορούσε κάλλιστα να είναι η ενσωμάτωση αναγώγιμου μεταθετικού γραφήματος κωδικοποιώντας το σε δυναμικό γράφημα κλήσεων συναρτήσεων, με τη διαφορά πως το αναγώγιμο μεταθετικό γράφημα δεν θα καλύπτει ή αλλιώς δεν θα αντιστοιχεί σε ένα προς ένα αντιστοιχία σε ολόκληρο το δυναμικό γράφημα κλήσεων.

Βιβλιογραφια

[1] AEPI Hellenic Society for the Protection of Intellectual Property, Retrieved September 2013, from the site http://www.aepi.gr/index.php?option=$com_content$&$task$ = $blogcategory$&id = 67&$Itemid$ = 122.

[2] G. Arboit, A method for watermarking java programs via opaque predicates, *In Proc. 5th International Conference on Electronic Commerce Research (ICECR-5)*, 2002.

[3] A. Balakrishnan, and C. Schulze, Code obfuscation literature survey, *CS701 Construction of Compilers*, 2005.

[4] Lucila M. S. Bento, and Davidson R. Boccardo, and Raphael C. S. Machado, and Vinícius Gusmão Pereira de Sá, and Jayme Luiz Szwarcfiter, Towards a provably robust graph-based watermarking scheme, *CoRR*, 2013.

[5] BSA Global Software Piracy Study - 2011 Edition, Retrieved September 2013, from the site http://portal.bsa.org/globalpiracy2011.

[6] BSA Global Software Piracy Study - 2013 Edition, Retrieved September 2013, from the site http://ww2.bsa.org/country/News%20and%20Events/News%20Archives/global/05152013-SoftwareValueStudy.aspx.

[7] CALLSTACK How the Java virtual machine handles method invocation and return, Retrieved August 2013, from the site http://www.javaworld.com/jw-06-1997/jw-06-hood.html.

[8] J.T Chan, Jien-Tsai, and W. Yang, Advanced obfuscation techniques for java bytecode, *J. Syst. Softw.*, pp. 1-10, 2004.

[9] I. Chionis, and M. Chroni, and A. Fylakis, and S.D. Nikolopoulos, Software, image and audio watermarking algorithmic techniques, *2th International Symposium on Computing in Informatics and Mathematics (ISCIM'13)*, 2013.

[10] I. Chionis, and M. Chroni, and S.D. Nikolopoulos, A dynamic watermarking model for embedding reducible permutation graphs into software, *10th International Conference on Security and Cryptography (SECRYPT'13)*, pp. 74-85, 2013.

[11] I. Chionis, and M. Chroni, and S.D. Nikolopoulos, Watermarking Java application programs using the WaterRpg dynamic model, *14th International Conference on Computer Systems and Technologies (CompSysTech'13)*, pp. 291-298, 2013.

[12] I. Chionis, and M. Chroni, and S.D. Nikolopoulos, Evaluating the WaterRpg software watermarking model on Java application programs, *17th Panhellenic Conference on Informatics (PCI'13)*, pp. 144-151, 2013.

[13] M. Chroni, and S.D. Nikolopoulos, Efficient encoding of watermark numbers as reducible permutation graphs, *CoRR abs/1110.1194*, 2011.

[14] M. Chroni, and S.D. Nikolopoulos, An embedding graph based model for software watermarking, *Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'12)*, 2012.

[15] M. Chroni, and S.D. Nikolopoulos, An efficient graph codec system for software watermarking, *36th Int'l Conference on Computers, Software, and Applications (COMPSAC'12), Workshop STPSA'12, IEEE Proceedings*, 2012.

[16] M. Chroni, and S.D. Nikolopoulos, Design and evaluation of a graph codec system for software watermarking, *2nd Int'l Conference on Data Management Technologies and Applications (DATA'13)*, 2013.

[17] M. Chroni, and A. Fylakis, and S.D. Nikolopoulos, Watermarking images in the frequency domain by exploiting self-inverting permutations, *Journal of Information Security*, 2013.

[18] M. Chroni, and A. Fylakis, and S.D. Nikolopoulos, A watermarking system for teaching intellectual property rights: implementation and performance, *11th Int'l Conference on Information Technology Based Higher Education and Training (ITHET'12)*, 2012.

[19] M. Chroni, and A. Fylakis, and S.D. Nikolopoulos, Watermarking images using 2D representations of self-inverting permutations, *8th Int'l Conference on Web Information Systems and Technologies (WEBIST'12)*, 2012.

[20] M. Chroni, and A. Fylakis, and S.D. Nikolopoulos, A watermarking system for teaching students to respect intellectual property rights, *4th Int'l Conference on Computer Supported Education (CSEDU'12)*, 2012.

[21] C. Collberg, and E. Carter, and S. Debray, and A. Huntwork, and J. Kececioglu, and C. Linn,and M. Stepp, Dynamic path based software watermarking, *Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation (PLDI'04)*, pp. 107-118, 2004.

[22] C. Collberg, and A. Huntwork, and E. Carter, and G. Townsend, Graph theoretic software watermarks: implementation, analysis, and attacks, *In Proceedings of the 6th Workshop on Information Hiding*, pp. 192-207, 2005.

[23] C. Collberg, and S. Kobourov, and E. Carter, and C. Thomborson, Error correcting graphs for software watermarking, *In Proc. 29th Workshop on Graph-Theoretic Concepts in Computer Science (WG'03)*, pp. 156-167, 2004.

[24] C. Collberg, and J. Nagra, Surreptitious Software, *Addison-Wesley*, 2010.

[25] C. Collberg, and C. Thomborson, and D. Low, Manufacturing cheap, resilient, and stealthy opaque constructs, *Principles of programming language (POPL'98)*, 1998.

[26] C. Collberg, and C. Thomborson, and D. Low, A functional taxonomy of obfuscating transformations, *Technical Report 148, Department of Computer Science, University of Auckland*, 1997.

[27] C. Collberg, and C. Thomborson, On the limits of software watermarking, *Computer Science Technical Reports 164*, 1998.

[28] C. Collberg, and C. Thomborson, Software watermarking: models and dynamic embeddings, *In Proc. 26th ACM SIGPLAN-SIGACT on Principles of Programming Languages (POPL'99)*, pp. 311-324, 1999.

[29] C. Collberg, and C. Thomborson, Watermarking, tamperproofing, and obfuscation - tools for software protection, *IEEE Transactions on Software Engineering*, pp. 735-746, 2002.

[30] C. Collberg, and S. Sha, and D. Tomko, and H. Wang, UWStego: A general architecture for software watermarking, *Retrieved September 2013, from the site http://www.mcs.gold.ac.uk/ mas01sd/papers/BriefSoftwareWatermarking.pdf*, 2001.

[31] P. Cousot, and R. Cousot, An abstract interpretation based framework for software watermarking, *Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 173-185, 2004.

[32] D. Curran, and N.J. Hurley, and M.Ó Cinnéide Securing java through software watermarking, *Proceedings of the 2nd international conference on Principles and practice of programming in Java (PPPJ'03)*, pp. 145-148, 2003.

[33] R. Davidson, and N. Myhrvold, Method and system for generating and auditing a signature for a computer program, *US5559884 A*, 1996.

[34] Call What is a function call, Retrieved September 2013, from the site http://eversystems.eu/Document/25/The_Cost_-_A_Function_Call.

[35] A. Gaul, Function call overhead benchmarks with MATLAB, Octave, Python, Cython and C, *CoRR*, 2012.

[36] S. Ghosh, and J. Hiser, and J.W. Davidson, Software protection for dynamically-generated code, *Proceedings of the 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop (PPREW'13)*, 2013.

[37] S. L. Graham, and P. B. Kessler, and M. K. Mckusick, Gprof: A call graph execution profiler, *Proceedings of the 1982 SIGPLAN symposium on Compiler construction (SIGPLAN'82)*, pp. 120-126, 1982.

[38] D. Gong, and F. Liu, and B. Lu, and P. Wang, and L.Ding, Hiding information on java class files, *International Symposium on Computer Science and Computational Technology*, pp. 160-164, 2008.

[39] J. Hamilton, and S. Danicic, A survey of static software watermarking, *World Congress on Internet Security (WorldCIS'11)*, pp. 100-107, 2011.

[40] F. Husain, A survey of digital watermarking techniques for multimedia data, *International Journal of Electronics and Communication Engineering*, pp. 37-43, 2002.

[41] Java Decompiler, Retrieved February 2013, from the site http://jd.benow.ca/.

[42] Java Inspiration, Retrieved September 2013, from the site http://en.wikipedia.org /wiki/Java_(programming_language).

[43] N. Kisserli, and D. Schellekens, and B. Preneel, Self encrypting code to protect against analysis and tampering, *1st Benelux Workshop on Information and System Security (WISSec'06)*, 2006.

[44] S. Krahmer, Generating runtime call graphs, 2006.

[45] F. Liu, and B. Lu, and X. Luo, A chaos based robust software watermarking, *Proceedings of the Second international conference on Information Security Practice and Experience (ISPEC'06)*, pp. 355-366, 2006.

[46] A. Mishra, and R. Kumar, and P.P. Chakrabarti, A method based whole program watermarking scheme for Java class files, *Journal Article*, 2008.

[47] A. Monden, and H. Iida, and k. Matsumoto, and K. Torii, and K. Inoue, A practical method for watermarking Java programs, *In Proc. 24th International Computer Software and Applications Conference (COMPSAC'00)*, pp. 191-197, 2000.

[48] G. Myles, and C. Collberg, Software watermarking via opaque predicates: implementation, analysis, and attacks, *Electronic Commerce Research*, pp. 155-171, 2006.

[49] J. Nagra, and C. Thomborson, and C. Collberg, A functional taxonomy for software watermarking, *Proceedings of the twenty-fifth Australasian conference on Computer science - Volume 4 (ACSC'02)*, 2002.

[50] J. Nagra, and C. Thomborson, and C. Collberg, Threading Software Watermarking, *PhD Thesis*, 2007.

[51] J. Palsberg, and S. Krishnaswamy, and M. Kwon, and D. Ma, and Q. Shao, and Y. Zhang, Experience with software watermarking, *Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, 2000.

[52] S. Praveen, and L. P. Sojan, Array data transformation for source code obfuscation, *Proceedings of World Academy of Science: Engineering & Technology*, 2007.

[53] M. D. Preda, and G. Roberto, and V. Enrico, Hiding software watermarks in loop structures, *Proceedings of the 15th international symposium on Static Analysis (SAS'08)*, pp. 174-188, 2008.

[54] ProGuard, Retrieved January 2013, from the site http://proguard.sourceforge.net/.

[55] G. Preda, and M. Potkonjak, Hiding software watermarks in loop structures, *Information Hiding*, pp. 348-367, 1999.

[56] S. P. Reiss, and M. Renieris, Generating java trace data, *Proceedings of the ACM 2000 Conference on Java Grande*, pp. 71-77, 2000.

[57] P. Shah, Code obfuscation for prevention of malicious reverse engineering attacks, *http://www.cs.ucsb.edu/ koc/ns/projects/02Reports/S2.pdf*.

[58] M.N. Shehzad, and Najam-Ul-Sahar, Text watermarking techniques, *http://www.scribd.com/doc/92569786/Text-Watermarking-Text-watermarking-Techniques-Survey-text-watermarking*.

[59] J. Stern, and G. Hachez, and F. Koeune, and J. Quisquater, Robust object watermarking: application to code, *Information Hiding, Springer Berlin Heidelberg, Lecture Notes in Computer Science*, pp. 368-378, 2000.

[60] Tax Law, Retrieved September 2013, from the site http://www.taxlaw.gr/el/idika-themata/168-software-copyright.

[61] S. Thaker, Software Watermarking via Assembly Code Transformations, *Msc Thesis, Department of Computer Science, University of San Jose State*, 2004.

[62] Important of stealth, Retrieved October 2013, from the site http://jameshamilton.eu/content/importance-stealth-software-watermarking.

[63] Technical Legislation To Computer Engineering, Retrieved September 2013, from the site http://el.wikibooks.org/wiki.

[64] R. Venkatesan, and V. Vazirani, and S. Sinha, A graph theoretic approach to software watermarking, *In 4th International Information Hiding Workshop, Pittsburgh, PA*, 2001.

[65] WIPO World Intellectual Property Organization, Retrieved September 2013, from the site http://www.wipo.int/portal/index.html.en.

[66] Z. Yu, and C. Thomborson and J. Wang and S. Lian and A.V. Vasilakos, Algorithms to watermark software through register allocation, *Softw., Pract. Exper.*, pp. 409-430, 2011.

[67] C. Zhang and J. Wang and C. Thomborson and C. Wang and C. Collberg, A semi dynamic multiple watermarking schemefor java applications, *Proceedings of the nineth ACM workshop on Digital rights management (DRM'09)*, pp. 59-72, 2009.

[68] W. Zhu, and C. Thomborson, Algorithms to watermark software through register allocation, *Digital Rights Management. Technologies, Issues, Challenges and Systems, Lecture Notes in Computer Science, Springer Berlin Heidelberg*, pp. 180-191, 2006.

ΔΗΜΟΣΙΕΥΣΕΙΣ

1. I. Chionis, M. Chroni, and S.D. Nikolopoulos, A dynamic watermarking model for embedding reducible permutation graphs into software, 10th International Conference on Security and Cryptography (SECRYPT'13), SciTePress Digital Library, pp. 74-85, 2013

2. I. Chionis, M. Chroni, and S.D. Nikolopoulos, Watermarking Java application programs using the WaterRpg dynamic model, 14th International Conference on Computer Systems and Technologies (CompSysTech'13), Association for Computing Machinery, pp. 291-298, 2013

3. I. Chionis, M. Chroni, and S.D. Nikolopoulos, Evaluating the WaterRpg software watermarking model on Java application programs, 17th Panhellenic Conference on Informatics (PCI'13), Association for Computing Machinery, pp. 144-151, 2013

Συντομο Βιογραφικο

Ο Ιωάννης Χιόνης έλαβε το έτος 2012 πτυχίο από το τμήμα Πληροφορικής, στη σχολή Θετικών Επιστημών του Πανεπιστήμιου Ιωαννίνων. Η πτυχιακή του εργασία κυρίως εστίαζε στην κατασκευή και αξιολόγηση δομών υδατογράφησης λογισμικού και έφερε τον τίτλο: Αλγοριθμικές Τεχνικές Γραφημάτων Υδατογράφησης Λογισμικού. Έλαβε μεταπτυχιακό δίπλωμα ειδίκευσης στην Πληροφορική το έτος 2014 από το τμήμα Μηχανικών Η/Υ και Πληροφορικής του Πανεπιστημίου Ιωαννίνων, όπου η διατριβή του εστίαζε στην εφαρμογή και αξιολόγηση μοντέλων υδατογράφησης σε κώδικες λογισμικών και έφερε τον τίτλο: Ένα Δυναμικό Μοντέλο Υδατογράφησης Λογισμικού βασισμένο σε Γραφήματα Κλήσεων Συναρτήσεων. Κατά την διάρκεια φοίτησής του, αρχικά σε προπτυχιακό επίπεδο ήταν μέλος της ομάδας υποστήριξης συστημάτων του τμήματος Πληροφορικής. Μετέπειτα σε μεταπτυχιακό επίπεδο διετέλεσε μέλος του "Εργαστηρίου Τεχνολογίας Αλγορίθμων" κάνοντας έρευνα στην αλγοριθμική θεωρία για απόκρυψη πληροφορίας καθώς και στην προστασία λογισμικού μέσω τεχνικών όπως είναι η υδατογράφηση, η συσκότιση και η θωράκιση κώδικα όπου και δημοσίευσε αποτελέσματά του. Παράλληλα με την έρευνα, άσκησε επικουρικό έργο στο εργαστήριο του μαθήματος ΠΛΥ402 – Σχεδίαση και Ανάλυση Αλγορίθμων.