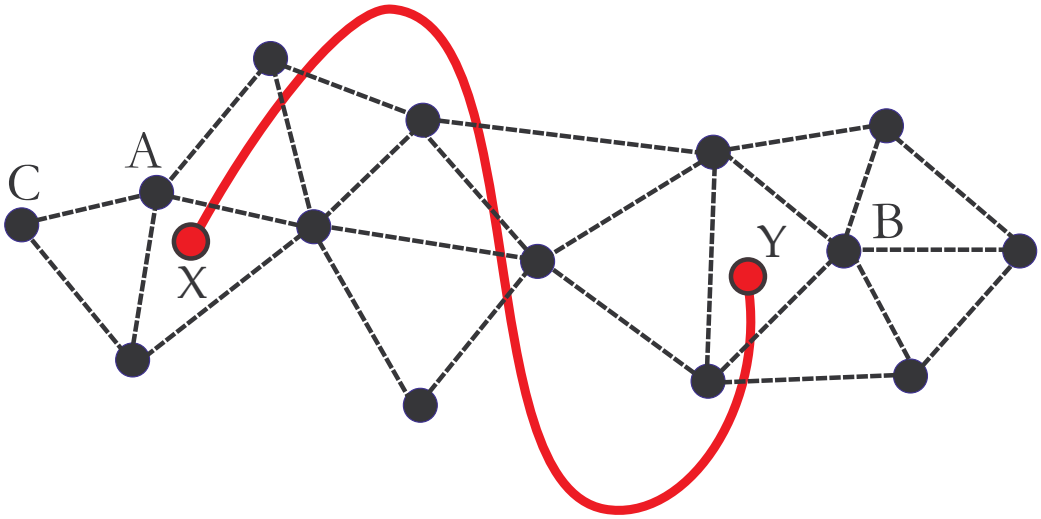
Wormhole Επιθέσεις
σε Ασύρματα Δίκτυα Επικοινωνιών



Διπλωματικη Εργασια

Νίκος Κιούρτης

Αθήνα, Σεπτέμβριος 2004

μΠλ∀

Περίληψη της εργασίας

Σε αυτή την εργασία θα παρουσιάσουμε και θα αναλύσουμε τις wormhole επιθέσεις σε ad–hoc δίκτυα. Σε αυτή την επίθεση ο εχθρός υποκλέβει μηνύματα που στέλνονται σε ένα μέρος του δικτύου και τα προωθεί (ίσως και τροποποιημένα) σε κάποιο άλλο μέρος του ασυρμάτου δικτύου. Ο εχθρός πετυχαίνει με αυτό τον τρόπο να μπερδέψει τους κόμβους του δικτύου ως προς το ποιά είναι η πραγματική θέση τους στο δίκτυο και ποιοί είναι οι πραγματικοί τους γείτονες. Έτσι, αυτές οι επιθέσεις είναι πολύ επικίνδυνες αν εφαρμοστούν εναντίον πολλών πρωτοκόλλων δρομολόγησης για ad–hoc δίκτυα, γιατί αυτές μπορούν να διενεργηθούν χωρίς αυτός να έχει στη διάθεσή του κάποιο κλειδί κρυπτογραφίας ή κάποια συγκεκριμένη πληροφορία για τον τρόπο που επικοινωνούν μεταξύ τους οι κόμβοι του δικτύου, και ακόμα σε αρκετές περιπτώσεις είναι δυνατόν να μην μπορεί ένας κόμβος να ανακαλύψει διαδρομές προς άλλο κόμβο οι οποίες να είναι μεγαλύτερες από δύο βήματα, με αποτέλεσμα να διακόπτονται πολλές από τις επικοινωνίες μεταξύ τους.

Στο κεφάλαιο 1 παρουσιάζουμε κάποια εισαγωγικά στοιχεία τα οποία χρειάζονται για την κατανόηση της υπόλοιπης εργασίας. Στο κεφάλαιο 2 εισάγουμε την έννοια της δρομολόγησης (routing) των μηνυμάτων σε ένα δίκτυο, και παρουσιάζουμε κάποιες μεθόδους και κάποια πρωτόκολλα δρομολόγησης τόσο σε ενσύρματα όσο και σε ασύρματα δίκτυα. Έπειτα, στο κεφάλαιο 3 παρουσιάζουμε την wormhole επίθεση, αναλύουμε τα κύρια χαρακτηριστικά της, και τέλος δείχνουμε πως αυτή εφαρμόζεται με επιτυχία σε πολλά υπάρχοντα (ασφαλή και μη) πρωτόκολλα δρομολόγησης για ασύρματα δίκτυα. Στο τέλος του κεφαλαίου αναφέρουμε και δύο παραλλαγές της επίθεσης, οι οποίες είναι οι επιθέσεις Byzantine Wormhole και Byzantine Overlay Wormhole. Στο επόμενο και τελευταίο κεφάλαιο 4 παρουσιάζουμε τρεις μεθόδους αντιμετώπισης των επιθέσεων wormhole που έχουν προταθεί, και αναλύουμε την τρίτη μέθοδο, η οποία χρησιμοποιεί κατευθυνόμενες κεραίες εκπομπής μηνυμάτων και βασίζεται στη συνεργασία μεταξύ των κόμβων του δικτύου, με σκοπό αυτοί να εξακριβώσουν αν υπάρχει κάποια wormhole στο δίκτυο και να την αποφύγουν. Το κεφάλαιο τελειώνει με μια σύγκριση των τριών μεθόδων και ορισμένες παρατηρήσεις.

Μέρος αυτής της διπλωματικής εργασίας έχει εκπονηθεί στο πανεπιστήμιο Carleton του Καναδά. Για αυτό το λόγο ευχαριστώ τον επιβλέποντα καθηγητή μου κ. Ευάγγελο Κρανάκη για την δυνατότητα που μου έδωσε να επισκεφτώ το πανεπιστήμιο αυτό, αλλά και για την έμπνευση και την βοήθεια που μου παρείχε από την αρχή μέχρι το τέλος της εργασίας.

Περιεχόμενα

Λίστα Σχημάτων

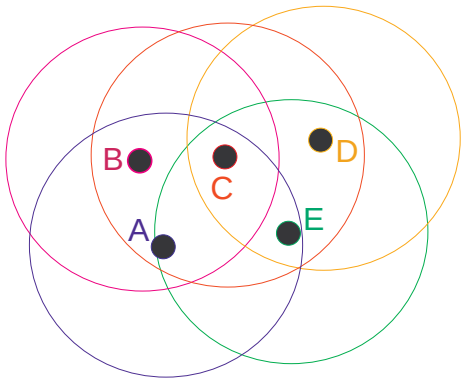
Κεφάλαιο 1

Εισαγωγικά στοιχεία

1.1 Τι είναι ένα Ad–Hoc δίκτυο

Ένα ad–hoc δίκτυο είναι μια συλλογή από συσκευές που είναι εξοπλισμένες με μια ασύρματη σύνδεση, οι οποίες συνεργάζονται με το να προωθούν η μια τα μηνύματα της άλλης μεταξύ τους, έτσι ώστε η κάθε συσκευή να μπορεί να επικοινωνήσει και με άλλες συσκευές που είναι έξω από την εμβέλεια μετάδοσής της. Οι συσκευές αυτές μπορεί να είναι διάφορα είδη υπολογιστών (όπως για παράδειγμα οι φορητοί υπολογιστές ή οι υπολογιστές παλάμης), ασύρματα τηλέφωνα, κλπ. Κάθε τέτοια συσκευή αποτελεί και ένα **κόμβο (node)**, του ad–hoc δικτύου, και υποθέτουμε ότι οι κόμβοι του δικτύου μπορεί να κινούνται. Για αυτό τον λόγο, τα ad–hoc δίκτυα συχνά αναφέρονται ως Mobile Ad–Hoc Networks (MANETs).

Τα ad–hoc δίκτυα δεν απαιτούν να υπάρχει κάποια κεντρική διαχείρηση η συγκεκριμένη υποδομή για να λειτουργήσουν, όπως για παράδειγμα access points. Αυτή είναι και η βασική διαφορά τους με τα ενσύρματα δίκτυα (wired networks) υπολογιστών: ότι δηλαδή σε ένα ad–hoc δίκτυο οι συνδέσεις μεταξύ των κόμβων δεν είναι προκαθορισμένες, και έτσι οι βασικές λειτουργίες του δικτύου, όπως για παράδειγμα η δρομολόγηση των μηνυμάτων, γίνονται με συνεργασία των κόμβων, λόγω της περιορισμένης εμβέλειας στην οποία μπορεί να μεταδώσει μήνυμα ένας κόμβος. Επειδή οι συνδέσεις των κόμβων δεν είναι προκαθορισμένες, σε ένα ad–hoc δίκτυο με τον όρο γείτονες ενός κόμβου A θα εννοούμε όλους τους κόμβους Γ_i οι οποίοι βρίσκονται εντός της εμβέλειας του A.



Σχήμα 1.1: Γείτονες σε ένα ad–hoc δίκτυο

Ο πίνακας που ακολουθεί δείχνει τους γείτονες του κάθε κόμβου στο παραπάνω σχήμα.

ΚΟΜΒΟΣ	ΓΕΙΤΟΝΕΣ
A	B, C, E
B	A, C
C	A, B, D, E
D	C, E
E	A, C, D

Στην υπόλοιπη εργασία θα χρησιμοποιούμε τις έννοιες ad–hoc δίκτυο και ασύρματο δίκτυο χωρίς να τις ξεχωρίζουμε.

Τέλος, όταν θα χρησιμοποιούμε την έννοια **αναμετάδοση (broadcast)**, για παράδειγμα ότι ο κόμβος Α αναμεταδίδει (ή κάνει broadcast) ένα μήνυμα, τότε εννοούμε ότι στο μήνυμα αυτό δεν γράφεται κάποιος συγκεκριμένος κόμβος που αποτελεί τον προορισμό του μηνύματος, δηλαδή όλοι οι κόμβοι που είναι μέσα στην εμβέλεια του Α μόλις λάβουν το μήνυμα θα θεωρήσουν ότι απευθύνεται σε αυτούς χωρίς να το απορρίψουν.

1.2 One–Way Hash Chains

Πριν ορίσουμε τις one–way hash chains θα τον ορισμό της one–way hash function.

1.2.1 One–Way Hash Functions

Μια hash function H ορίζεται ως μια συνάρτηση της οποίας οι τιμές της να υπολογίζονται εύκολα από υπολογιστή, και παίρνει ως είσοδο ένα δυαδικό αριθμό οποιουδήποτε αριθμού ψηφίων και δίνει ως έξοδο ένα δυαδικό αριθμό που έχει καθορισμένο αριθμό ψηφίων, δηλαδή $H : \{0,1\}^* \rightarrow \{0,1\}^n$, όπου ο αριθμός n είναι προκαθορισμένος. Οι hash functions που χρησιμοποιούνται στην πράξη, έχουν τις εξής τρεις ιδιότητες:

1. Θεωρούμε ένα στοιχείο $b \in \{0,1\}^n$. Η πιθανότητα να ισχύει $H(a) = b$ για μια τυχαία τιμή $a\{0,1\}^*$ είναι $\frac{1}{2^n}$.
2. Η συνάρτηση H είναι *strongly collision–free*, δηλαδή είναι υπολογιστικά αδύνατον να βρεθούν δύο στοιχεία $x, x' \in \{0,1\}^*$ έτσι ώστε $x \neq x'$ και $H(x) = H(x')$.
3. Δεδομένης της τιμής $y \in \{0,1\}^n$, είναι υπολογιστικά αδύνατον να βρούμε ένα $x \in \{0,1\}^*$ έτσι ώστε $H(x) = y$.

Ειδικότερα, μια hash function η οποία ικανοποιεί την τρίτη ιδιότητα από τις παραπάνω, ονομάζεται one–way hash function, γιατί είναι υπολογιστικά εύκολο να υπολογίσουμε την τιμή $H(x)$ και υπολογιστικά αδύνατον να υπολογίσουμε την τιμή $H^{-1}(x)$.

1.2.2 One–Way Hash Chains

Σε κάποια από τα πρωτόκολλα δρομολόγησης που θα παρουσιάσουμε παρακάτω θα χρειαστούμε τις one–way hash chains (ή για συντομία hash chains). Μια one–way hash chain κατασκευάζεται ως εξής:

Διαλέγουμε τυχαία μια τιμή h_0 (που θα είναι η πρώτη τιμή της αλυσίδας) και κατόπιν εφαρμόζουμε μια one–way hash function όσες φορές θέλουμε να είναι και τα στοιχεία της αλυσίδας. Έτσι, αν θέλουμε να φτιάξουμε την one–way hash chain $h_0, h_1, h_2, \ldots, h_n$ χρησιμοποιώντας την hash function H, αφού διαλέξουμε τυχαία την τιμή h_0 θα ισχύει ότι:

$$h_1 = H(h_0),\ h_2 = H(h_1) = H(H(h_0)), \quad h_{i+1} = H(h_i),\ \ i = 0, 1, \ldots, n-1$$

Οι one-way hash chains έχουν δύο πολύ σημαντικές ιδιότητες:

1. Αν κάποιος έχει την τιμή h_i, τότε μπορεί να εξακριβώσει αν η τιμή h_j ανήκει στην hash chain και $j > i$, και η εξακρίβωση γίνεται με τον έλεγχο $H^{j-i}(h_i) = h_j$ (δηλαδή εφαρμόζοντας την συνάρτηση H $j - i$ φορές στην τιμή h_i).

2. Επειδή είναι υπολογιστικά αδύνατον να υπολογίσει κανείς την τιμή $H^{-1}(h_i)$, δεν μπορεί κανείς να βρει μια τιμή h'_j, $j > i$, έτσι ώστε $H^{j-i}(h_i) = h'_j$

Έτσι, αυτές τις δύο παραπάνω ιδιότητες μας δίνουν την δυνατότητα να μπορούμε να εξακριβώσουμε πότε οι τιμές που λαμβάνουμε και ανήκουν σε μια hash chain είναι αυθεντικές, αφού αν η h_a ανήκει στην hash chain και λάβουμε μια άλλη τιμή h_b με την ιδιότητα $H^{b-a}(h_a) = h_b$, τότε ξέρουμε ότι και οι δύο τιμές ανήκουν στην ίδια hash chain και μπορεί να μας τις έστειλε μόνο ο δημιουργός της.

1.3 Message Authentication Codes

Όταν δύο κόμβοι του δικτύου ανταλλάσσουν μηνύματα, θέλουν να έχουν την δυνατότητα να ελέγξουν ότι τα μηνύματα αυτά είναι αυθεντικά, ότι δηλαδή όντως τα έστειλε ο κόμβος ο οποίος νομίζουν και ότι από την στιγμή που εστάλησαν μέχρι την στιγμή που παραδώθηκαν στον προορισμό τους δεν έχει αλλοιωθεί το περιεχόμενό τους από κάποιον άλλο κόμβο. Ένα message authentication code είναι ένα μικρό κομμάτι δεδομένων το οποίο προσθέτουμε στο αρχικό μήνυμα πριν το στείλουμε και μας εξασφαλίζει τα παραπάνω. Υπάρχουν διάφορα message authentication codes, και περιγράφουμε μερικά από αυτά παρακάτω.

Αν οι δύο κόμβοι A και B που ανταλλάσσουν μηνύματα μοιράζονται κάποιο κοινό κλειδί κρυπτογράφησης δεδομένων K_{AB}, τότε ένας τρόπος δημιουργίας ενός message authentication code είναι να χρησιμοποιήσουμε μια συνάρτηση F η οποία δέχεται ως είσοδο το μήνυμα M που θέλει να στείλει ο A στον B και το κλειδί K_{AB}, να υπολογίσουμε την τιμή $V = F(M, K_{AB})$. Η τιμή V είναι το message authentication code του μηνύματος M, και αντί να στείλουμε μόνο το M, στέλνουμε το ζεύγος (M, V). Έτσι, μόλις ο B λάβει το μήνυμα (M, V), θα υπολογίσει την τιμή $V' = F(M, K_{AB})$, και αν $V = V'$ τότε θα ξέρει ότι το μήνυμα είναι αυθεντικό.

Ένας άλλος τρόπος δημιουργίας ενός message authentication code είναι να χρησιμοποιήσουμε μια one–way hash function. Έτσι, αν υποθέσουμε ότι οι κόμβοι Α και Β μοιράζονται κάποια μυστική τιμή S και η H είναι μια one–way hash function, τότε ένα message authentication code για το μήνυμα M είναι η τιμή $\text{MAC}_S = H[M\|S]$, όπου με $M\|S$ συμβολίζουμε το μήνυμα που προκύπτει αν προσθέσουμε στο τέλος του μηνύματος M την τιμή S. Η τιμή MAC_S είναι το message authentication code του μηνύματος M, και στον Β θα στείλουμε το ζεύγος (M, MAC_S). Έτσι, μόλις ο Β λάβει το μήνυμα (M, MAC_S), θα υπολογίσει την τιμή $\text{MAC}'_S = H[M\|S]$, και αν $\text{MAC}_S = \text{MAC}'_S$ τότε θα ξέρει ότι το μήνυμα είναι αυθεντικό.

Από εδώ και στο εξής, όταν γράφουμε MAC θα εννοούμε message authentication code.

Στο επόμενο κεφάλαιο θα περιγράψουμε πως γίνεται η δρομολόγηση των μηνυμάτων σε ενσύρματα δίκτυα (wired networks) και σε ασύρματα δίκτυα.

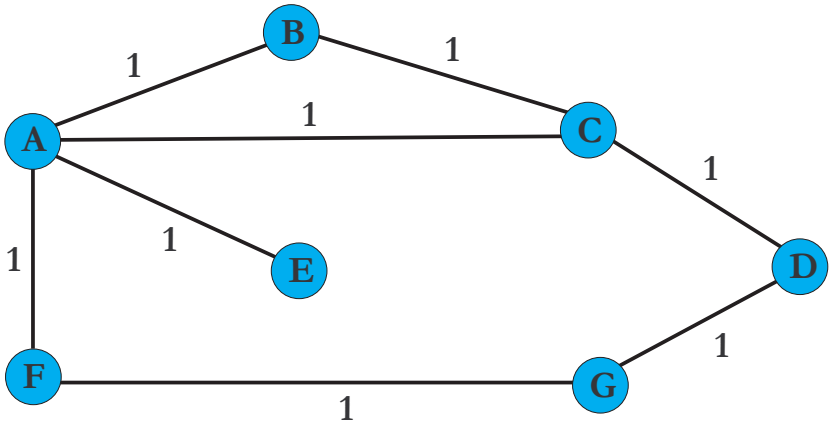
Κεφάλαιο 2

Routing σε Δίκτυα και Ασύρματα Δίκτυα

2.1 Routing σε Ενσύρματα (Wired) Δίκτυα

Όταν βρισκόμαστε σε ένα ενσύρματο δίκτυο, τότε όλες οι συνδέσεις μεταξύ των κόμβων είναι προκαθορισμένες. Ένα από τα προβλήματα τα οποία εμφανίζονται είναι πως θα δρομολογηθεί ένα μήνυμα από ένα κόμβο A στον προορισμό του B. Με την έννοια **δρομολόγηση (routing)** εννοούμε την διαδικασία που εκτελείται για να παραδωθούν μηνύματα μεταξύ δυο μή γειτονικών κόμβων σε ένα δίκτυο. Για να το πετύχουμε αυτό, χρησιμοποιούμε κάποιο **πρωτόκολλο δρομολόγησης (routing protocol)**. Το πρωτόκολλο γενικά είναι ένα σύνολο κανόνων τους οποίους χρησιμοποιούν δύο ή περισσότερες συσκευές για να επικοινωνήσουν μεταξύ τους.

Ας θεωρήσουμε για παράδειγμα το δίκτυο του σχήματος 2.1:



Σχήμα 2.1: Παράδειγμα ενός wired δικτύου

Για να γίνει η δρομολόγηση των μηνυμάτων από τον ένα κόμβο στον άλλο στο παραπάνω δίκτυο, ένας τρόπος είναι κάθε κόμβος να χρησιμοποιεί ένα **πίνακα δρομολόγησης (routing table)**. Αυτοί οι πίνακες συνήθως δεν περιέχουν όλη

την διαδρομή από την αρχή στον προορισμό, αλλά περιέχουν τον επόμενο κόμβο της διαδρομής που θα χρησιμοποιηθεί στο οποίο πρέπει να δρομολογηθεί το μήνυμα για να φτάσει στον προορισμό του, καθώς και το κόστος για να φτάσει το μήνυμα στον προορισμό (δηλαδή το κόστος όλης της διαδρομής). Για παράδειγμα, στο δίκτυο του σχήματος 2.1, αν θέλουμε να χρησιμοποιούμε γενικά την συντομότερη διαδρομή, τότε τα routing tables για τους κόμβους A και F θα είναι τα παρακάτω:

ΓΙΑ ΤΟΝ ΚΟΜΒΟ A:

ΠΡΟΟΡΙΣΜΟΣ	ΕΠΟΜΕΝΟΣ ΚΟΜΒΟΣ	ΚΟΣΤΟΣ
B	B	1
C	C	1
D	C	2
E	E	1
F	F	1
G	F	2

ΓΙΑ ΤΟΝ ΚΟΜΒΟ F:

ΠΡΟΟΡΙΣΜΟΣ	ΕΠΟΜΕΝΟΣ ΚΟΜΒΟΣ	ΚΟΣΤΟΣ
A	A	1
B	A	2
C	A	2
D	G	2
E	A	2
G	G	1

Έτσι, αν ο κόμβος A θέλει να στείλει ένα μήνυμα στον κόμβο G, τότε το routing protocol που χρησιμοποιείται μόλις συμβουλευτεί το routing table του A θα δει ότι πρέπει να προωθήσει το μήνυμά του στον κόμβο F (και ότι το κόστος της διαδρομής είναι 2), και μόλις το μήνυμα φτάσει στον κόμβο F το πρωτόκολλο θα συμβουλευτεί το routing table του F και θα δει ότι για να φτάσει το μήνυμα στον προορισμό του G θα πρέπει να το στείλει κατευθείαν στον G (που είναι γειτονικός κόμβος).

Η διαδικασία που ακολουθείται για να δημιουργηθούν τα rouitng tables καθορίζεται από τον **αλγόριθμο δρομολόγησης (routing algorithm)**. Υπάρχουν τέσσερις βασικοί τύποι σύμφωνα με τους οποίους γίνεται η δρομολόγηση, τους οποίους θα περιγράψουμε συνοπτικά παρακάτω.

Centralized Routing

Με τον όρο **centralized routing** εννοούμε ότι όλες οι πληροφορίες σχετικά με την διασύνδεση των κόμβων που χρειάζονται για να γίνει η δρομολόγηση δημιουργούνται και φυλάσσονται σε μια κεντρική υπηρεσία. Κατόπιν, αυτή η υπηρεσία στέλνει σε όλους τους κόμβους αυτές τις πληροφορίες, έτσι ώστε ο κάθε κόμβος να φτιάξει το δικό του routing table. Ένα τρόπος να φυλάσσονται κεντρικά όλες οι πληροφορίες για τη δρομολόγηση είναι με τη δημιουργία ενός **routing matrix**, ο οποίος για την περίπτωση του δικτύου του σχήματος 2.1 της σελίδας 5 θα είναι ο εξής:

ΠΡΟΟΡΙΣΜΟΣ

ΑΡΧΗ	A	B	C	D	E	F	G
A	—	B	C	C	E	F	F
B	A	—	C	C	A	A	A
C	A	B	—	D	A	A	D
D	C	C	C	—	C	G	G
E	A	A	A	A	—	A	A
F	A	A	A	G	A	—	G
G	F	F	D	D	F	F	—

Μόλις κάθε κόμβος λάβει τον παραπάνω πίνακα, τότε χρησιμοποιεί αρχικά την γραμμή που έχει αυτόν ως αρχή για να φτιάξει τη στήλη ΕΠΟΜΕΝΟΣ ΚΟΜΒΟΣ και μετά ό,τι χρειάζεται από τον υπόλοιπο πίνακα για να φτιάξει τη στήλη ΚΟΣΤΟΣ.

Distributed Routing

Με τον όρο **Distributed routing** εννοούμε ότι δεν υπάρχει κάποια κεντρική διαχείρηση των πληροφοριών όπως πριν. Επομένως, κάθε κόμβος πρέπει να καθορίζει και να διατηρεί το routing table του. Αυτό συνήθως γίνεται γνωρίζοντας ποιοί είναι οι γείτονές του, υπολογίζοντας το κόστος όταν στέλνει ένα μήνυμα σε κάποιο γειτονικό κόμβο και μαθαίνοντας το κόστος ενός γείτονα για να στείλει ένα μήνυμα σε ορισμένους προορισμούς. Αυτές τις πληροφορίες τις μαθαίνει ένας κόμβος επικοινωνώντας διαρκώς με τους γείτονές του, και από αυτές φτιάχνει το routing table του.

Static Routing

Με τον όρο **Static routing** εννοούμε ότι μόλις ο κάθε κόμβος φτιάξει το routing table του, τότε δεν το αλλάζει. Αυτό φυσικά είναι λειτουργικό μόνο σε περιπτώσεις όπου οι συνθήκες σύμφωνα με τις οποίες δημιουργήθηκε ο πίνακας δεν αλλάζουν με το χρόνο, και επομένως η γρηγορότερη διαδρομή από ένα κόμβο σε κάποιον άλλο είναι παραμένει σταθερή με τον χρόνο.

Adaptive Routing

Με τον όρο **Adaptive routing** εννοούμε ότι κάθε κόμβος ενημερώνει το routing table του ανάλογα με το πως αλλάζει η συμπεριφορά του δικτύου, όπως για παράδειγμα όταν αλλαζει το κόστος κάποιων συνδέσεων μεταξύ των κόμβων επειδή υπάρχει μεγάλη διεύλευση μηνυμάτων εκείνη τη στιγμή από εκεί.

Στις επόμενες δύο ενότητες θα περιγράψουμε δύο τρόπους δρομολόγησης των μηνυμάτων, το Distance Vector Routing το οποίο χρησιμοποιεί τον αλγόριθμο Bellman-Ford, και το Link State Routing το οποίο χρησιμοποιεί τον αλγόριθμο του Dijkstra.

2.1.1 Distance Vector Routing και το πρωτόκολλο RIP

Ο αλγόριθμος για DISTANCE VECTOR ROUTING βρίσκει την συντομότερη διαδρομή από την αρχή στον προορισμό, δουλεύοντας ανάποδα, δηλαδή ξεκινώντας την εύρεση από τον προορισμό. Για να το κάνει αυτό, χρησιμοποιεί τον αλγόριθμο Bellman-Ford, και βασίζεται στην ακόλουθη αρχή:

Έστω ότι το κόστος της "φθηνότερης" διαδρομής από τον κόμβο A στον κόμβο G είναι $\text{COST}(A, G)$ και ο A συνδέεται απευθείας με τους κόμβους B, C, ..., E. Τότε:

$$\text{COST}(A,G) = \min \begin{cases} \text{COST}(A,B) + \{\text{κόστος φθηνότερης διαδρομής από B σε G}\} \\ \text{COST}(A,C) + \{\text{κόστος φθηνότερης διαδρομής από C σε G}\} \\ \quad \vdots \\ \text{COST}(A,E) + \{\text{κόστος φθηνότερης διαδρομής από E σε G}\} \end{cases}$$

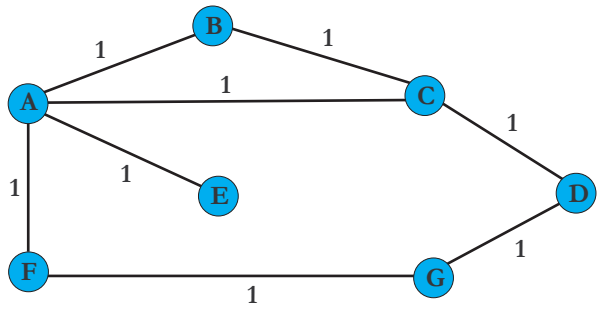
Σύμφωνα με τα προηγούμενα, ο A μπορεί να υπολογίσει την φθηνότερη διαδρομή αν ξέρει τα κόστη της επικοινωνίας με κάθε γείτονα και κάθε γείτονας γνωρίζει την φθηνότερη διαδρομή για τον G. Το πως γνωρίζει κάθε γείτονας την φθηνότερη διαδρομή για τον κόμβο G προκύπτει από την κατασκευή του αλγορίθμου, τον οποίο περιγράφουμε με ψευδοκώδικα παρακάτω:

BELLMAN-FORD ALGORITHM

```
Για κάθε γειτονικό κόμβο X καταχώρησε την εγγραφή
(X, κόστος για αποστολή στο X) στο τρέχον routing table.

Επανέλαβε τις παρακάτω γραμμές περιοδικά
  Για κάθε γειτονικό κόμβο N εκτέλεσε
  {
    πάρε πληροφορίες από το routing table του N;
    για κάθε κόμβο Z στο routing table του N εκτέλεσε
        εαν ο Z είναι στο τρέχον routing table
            πρόσθεσε το ζεύγος (N, "τρέχον κόστος για N" + "κόστος
            για Z από πίνακα του N") στο τρέχον routing table;
        αλλιώς
            εαν "τρέχον κόστος για N" + "κόστος για Z από πίνακα
                του N" < "τρέχον κόστος για Z" τότε αντικατέστησε
                το τρέχον κόστος του Z με την τιμή ("τρέχον κόστος
                για N" + "κόστος για Z από πίνακα του N") και
                όρισε τον N ως πρώτο κόμβο στη route για τον Z;
  }
```

Στον παραπάνω ψευδοκώδικα, όταν λέμε "Επανέλαβε τις παρακάτω γραμμές περιοδικά" εννοούμε ότι τα παρακάτω επαναλαμβάνονται ανά τακτά χρονικά διαστήματα έτσι ώστε τα routing tables των κόμβων να ενημερώνονται κάθε τόσο για να μπορούν να ανταπεξέλθουν στις συνεχείς αλλαγές στις οποίες υπόκειται το δίκτυο (αρχικά δηλαδή η επανάληψη γίνεται μέχρι να μην παρατηρηθεί αλλαγή κανενός από τα routing tables των κόμβων).

Ας υποθέσουμε τώρα ότι στο δίκτυο του σχήματος 2.1 της σελίδας 5 ο κόμβος Α θέλει να στείλει ένα μήνυμα στον κόμβο G και ως routing protocol χρησιμοποιείται το distance vector routing.



Σχήμα 2.2: Η διαδρομή $A \to G$ είναι $A \to F \to G$ (Distance Vector Routing)

Μετά από την εκτέλεση των δύο πρώτων γραμμών του αλγορίθμου Bellman-Ford, το routing table του Α είναι το:

ΠΡΟΟΡΙΣΜΟΣ (ΑΠΟ Α)	B	C	D	E	F	G
ΚΟΣΤΟΣ	1	1	∞	1	1	∞
ΕΠΟΜΕΝΟΣ ΚΟΜΒΟΣ	B	C	—	E	E	—

και γενικά τα κόστη που υπάρχουν στα routing tables όλων των κόμβων συνοψίζονται στον παρακάτω πίνακα:

ΑΡΧΗ / ΠΡΟΟΡΙΣΜΟΣ	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Ύστερα από την εκτέλεση όλων των υπολοίπων βημάτων του αλγορίθμου, οι παραπάνω πίνακες έχουν την τελική μορφή:

ΠΡΟΟΡΙΣΜΟΣ (ΑΠΟ Α)	B	C	D	E	F	G
ΚΟΣΤΟΣ	1	1	2	1	1	2
ΕΠΟΜΕΝΟΣ ΚΟΜΒΟΣ	B	C	C	E	E	F

ΑΡΧΗ / ΠΡΟΟΡΙΣΜΟΣ	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	3
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

Πάνω στο distance vector routing βασίζεται και το Routing Information Protocol (RIP) το οποίο έχει χρησιμοποιηθεί στο Internet στο παρελθόν.

2.1.2 Link State Routing και το πρωτόκολλο OSPF

Ένας άλλος τρόπος δρομολόγησης των δεδομένων ονομάζεται Link State Routing. Αυτός ο τρόπος δρομοόγησης έχει μόνο μια ομοιότητα με το Distance Vector Routing, η οποία είναι ότι και εδώ κάθε κόμβος ανταλάσσει ό,τι πληροφορίες έχει με τους γείτονές του. Η διαφορά όμως είναι στα δεδομένα που ανταλάσσονται. Έτσι, το Link State Routing περιληπτικά βασίζεται στα εξής βήματα:

- Κάθε κόμβος μαζεύει πληροφορίες σχετικά με την κατάσταση των συνδέσεων προς τους γείτονές του. Για παράδειγμα, τέτοιες σημαντικές πληροφορίες μπορεί να περιλαμβάνουν το ρυθμό των bits ανά δευτερόλεπτο που μπορούν να σταλούν σε αυτή την σύνδεση, την ελάχιστη καθυστέρηση προτού μπορεί να αρχίσει η αποστολή μηνυμάτων σε αυτή τη σύνδεση, το πλήθος των μηνυμάτων που βρίσκονται σε ουρά προκειμένου να σταλούν, και το πόσο αξιόπιστη είναι η σύνδεση.

- Κάθε κόμβος δημιουργεί ένα *μήνυμα κατάστασης (link state packet)* για κάθε σύνδεση προς γείτονά του. Με αυτό το μήνυμα μπορού να αναγνωριστούν οι δύο κόμβοι που συνδέονται με αυτή την σύνδεση, και ακόμα το μήνυμα περιέχει όλες τις πληροφορίες που έχει συλλέξει ο κόμβος πιο πριν. Μόλις το μήνυμα δημιουργηθεί, κάθε κόμβος το στέλνει σε εκείνο τον γείτονα του στον οποίο αυτό αναφέρεται.

- Όλοι οι κόμβοι οι οποίοι δέχονται link state packets τα προωθούν σε όλους τους γείτονές τους (εκτός φυσικά από αυτόν από τον οποίο το έλαβαν αρχικά).

- Καθώς οι κόμβοι ανταλάσσουν μεταξύ τους link state packets, κάθε κόμβος τελικά μαθαίνει την τοπολογία του δικτύου αλλά και το κόστος και τη κατάσταση των συνδέσεων μεταξύ των κόμβων. Συνεπώς, μπορεί έπειτα να εκτελέσει ένα αλγόριθμο δρομολόγησης ο οποίος βρίσκει την φθηνότερη διαδρομή (όπως ο αλγόριθμος του Dijkstra που θα περιγράψουμε στη συνέχεια) χρησιμοποιώντας τον εαυτό του ως αρχή, για να βρει την διαδρομή στον προορισμό που θέλει να στείλει το μήνυμά του. Έτσι, βρίσκει και τον επόμενο κόμβο της διαδρομής στον οποίο πρέπει να στείλει το μήνυμα για να προωθηθεί στον προορισμό, και φτιάχνει το δικό του routing table.

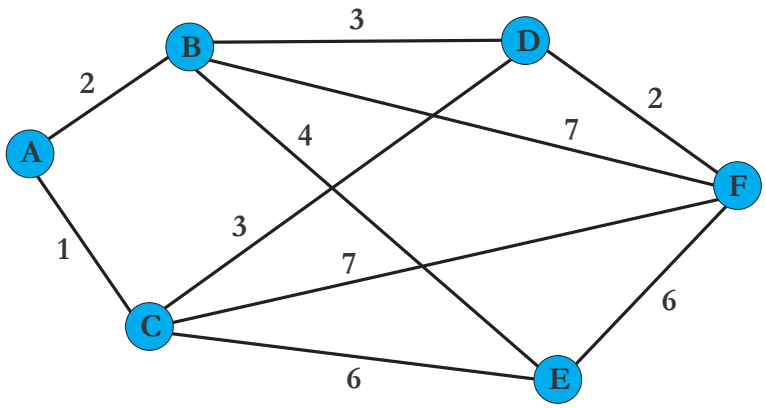
Με αυτό τον τρόπο το routing protocol αντιδρά πιο γρήγορα σε πιθανές αλλαγές του δικτύου σε σχέση με το distance vector routing (για παράδειγμα τέτοιες αλλαγές μπορεί να είναι αυξομειώσεις του κόστους των συνδέσεων μεταξύ των κόμβων λόγω της κίνησης που δημιουργείται στο δίκτυο)).

Στη συνέχεια θα περιγράψουμε με ψευδοκώδικα τον αλγόριθμο του Dijkstra:

Dijkstra's Algorithm

```
Έστω ότι S είναι ένα σύνολο κόμβων το οποίο αρχικά
περιέχει τον κόμβο A.

Ορίζουμε με Cost(X) το κόστος της φθηνότερης διαδρομής από
τον κόμβο A στον X χρησιμοποιώντας μόνο κόμβους από το σύνολο
S (εκτός του X). Αρχικά, η τιμή Cost(X) είναι το κόστος της
σύνδεσης του A με το X. Αν δεν υπάρχει τέτοια απ'ευθείας
σύνδεση, τότε η τιμή Cost(X) είναι μια αυθαίρετα μεγάλη τιμή
(η οποία είναι δηλαδή μεγαλύτερη από οποιαδήποτε διαδρομή μέσα
στο δίκτυο). Για όλους τους κόμβους οι οποίοι έχουν απ'ευθείας
σύνδεση με τον A (είναι δηλαδή γείτονες) ορίζουμε Prior(X) = A.

WHILE (δεν βρίσκονται όλοι οι κόμβοι στο σύνολο S)
DO
  {
        Καθόρισε το σύνολο των κόμβων που δεν βρίσκονται στο S,
        αλλά συνδέονται με ένα κόμβο που ανήκει στο S. Ονόμασε
        αυτό το σύνολο W.

        Διάλεξε τον κόμβο X στο W με τη μικρότερη τιμή Cost(X).
        Πρόσθεσε τον X στο σύνολο S.

        Για κάθε κόμβο V που δεν ανήκει στο S, θέσε ως Cost(V)
        το minimum των τιμών {Cost(V), Cost(X) + "κόστος της
        απευθείας σύνδεσης των X και V"}. Εαν η τιμή Cost(V)
        αλλάξει θέσε Prior(V) = X.
  }
```

Ας υποθέσουμε τώρα ότι έχουμε το δίκτυο του παρακάτω σχήματος 2.3:



Σχήμα 2.3: Παράδειγμα δικτύου για Link State Routing (Dijkstra)

Έστω τώρα ότι ο κόμβος A θέλει να στείλει ένα μήνυμα στον κόμβο F. Μόλις οι κόμβοι του δικτύου ανταλάξουν μεταξύ τους όλα τα πιθανά link state packets, ο κόμβος A θα γνωρίζει, ανάμεσα σε άλλα πράγματα, και το κόστος της κάθε σύνδεσης στο δίκτυο, οπότε θα εκτελέσει τον αλγόριθμο του Dijkstra χρησιμοποιώντας ως σύνολο S τον εαυτό του, δηλαδή αρχικά $S = \{A\}$. Βλέπουμε επίσης ότι οι γειτονικοί κόμβοι που συνδέονται με τον A είναι οι B και C με κόστη 2 και 1 αντίστοιχα, οπότε ο αλγόριθμος θέτει $\text{Cost}(B) = 2$, $\text{Cost}(C) = 1$, $\text{Cost}(D) = \text{Cost}(E) = \text{Cost}(F) = \infty$ και $\text{Prior}(B) = \text{Prior}(C) = A$. Επειδή μόνοι οι κόμβοι B και C συνδέονται με τον A, το σύνολο W περιέχει αρχικά μόνο αυτούς, και θέτει $X = C$ επειδή αυτός έχει το μικρότερο κόστος και τον προσθέτει στο σύνολο S. Έπειτα ο αλγόριθμος εξετάζει τα κόστη $\text{Cost}(V)$ για κάθε $V \notin S$, δηλαδή για τους κόμβους B,D,E και F, και βλέπει αν υπάρχει κάποιος κόμβος V έτσι ώστε να ισχύει ότι το άθροισμα των τιμών $\text{Cost}(C)$ και της τιμής που εκφράζει το κόστος της (απευθείας) σύνδεσης του C με τον V είναι μικρότερο από το ήδη ορισμένο κόστος $\text{Cost}(V)$. Αναλυτικά, έχουμε ότι $\text{Cost}(B) = 2$ και επειδή δεν υπάρχει απευθείας σύνδεση του C με το B δεν αλλάζει η τιμή $\text{Cost}(B)$. Ακόμα, επειδή ισχύουν ότι

$$\text{Cost}(C) + \text{Cost}(\text{“σύνδεση του C και D”}) = 1 + 3 = 4 < \infty = \text{Cost}(D)$$
$$\text{Cost}(C) + \text{Cost}(\text{“σύνδεση του C και E”}) = 1 + 6 = 7 < \infty = \text{Cost}(E)$$
$$\text{Cost}(C) + \text{Cost}(\text{“σύνδεση του C και F”}) = 1 + 7 = 8 < \infty = \text{Cost}(F)$$

ο αλγόριθμος θέτει $\text{Cost}(D) = 4$, $\text{Prior}(D) = C$, $\text{Cost}(E) = 6$, $\text{Prior}(E) = C$, $\text{Cost}(F) = 8$ και $\text{Prior}(F) = C$. Ο παρακάτω πίνακας δείχνει τι θα συμβεί μέχρι να τερματίσει ο αλγόριθμος (μετά το πέρας του πέμπτου βήματος προστίθεται και ο κόμβος F στο σύνολο S και έτσι ο αλγόριθμος τερματίζει):

				συναρτηση Cost					συναρτηση Prior				
βημα	S	W	X	B	C	D	E	F	B	C	D	E	F
1	$\{A\}$	$\{B, C\}$	C	2	1	∞	∞	∞	A	A	$-$	$-$	$-$
2	$\{A, C\}$	$\{B, D, E, F\}$	B	2	1	4	7	8	A	A	C	C	C
3	$\{A, B, C\}$	$\{D, E, F\}$	D	2	1	4	6	8	A	A	C	B	C
4	$\{A, B, C, D\}$	$\{E, F\}$	E	2	1	4	6	6	A	A	C	B	D
5	$\{A, B, C, D, E\}$	$\{F\}$	F	2	1	4	6	6	A	A	C	B	D

Η τελευταία γραμμή του πίνακα μας δείχνει ότι οι φθηνότερες διαδρομές για τους κόμβους B, C, D, E και F έχουν κόστη 2, 1, 4, 6 και 6 αντίστοιχα. Όπως είναι φανερό, ο αλγόριθμος υπολογίζει απ'ευθείας τα ελάχιστα κόστη για τις διαδρομές προς όλους τους κόμβους και όχι τις συγκεκριμένες διαδρομές. Έτσι, για να βρει ο κόμβος A την φθηνότερη διαδρομή θα χρησιμοποιήσει την συνάρτηση Prior. δηλαδή θα υπολογίσει ότι $\text{Prior}(F) = D$ (δηλαδή στην φθηνότερη διαδρομή $A \to F$ ο κόμβος D είναι πριν τον F), $\text{Prior}(D) = C$ και τέλος $\text{Prior}(C) = A$. Άρα, η φθηνότερη διαδρομή από A στο F είναι η $A \to C \to D \to F$ με κόστος $1 + 3 + 2 = 6$.

Πάνω στο link state routing βασίζεται το πρωτόκολλο Open Shortest Path Fisrt (OSPF), το οποίο χρησιμοποιείται ευρέως στο Internet σήμερα.

2.2 Routing σε Ασύρματα (Wireless) Δίκτυα

Στη συνέχεια θα κάνουμε μια σύντομη περιγραφή της λειτουργίας μερικών πρωτοκόλων δρομολόγησης για ασύρματα δίκτυα. Γενικά, τα routing protocols για wireless δίκτυα χωρίζονται σε δυο μεγάλες κατηγορίες: τα **proactive** ή **periodic** και τα **reactive** ή **on–demand**.

Proactive (periodic) routing protocols

Σε ένα proactive routing protocol, οι κόμβοι ανταλάσσουν περιοδικά μεταξύ τους ό,τι πληροφορίες για δρομολόγηση έχουν συγκεντρώσει, με απώτερο στόχο **πάντα να γνωρίζει κάθε κόμβος μια τουλάχιστον διαδρομή προς όλους τους άλλους κόμβους**.

Reactive (on–demand) routing protocols

Όταν χρησιμοποιείται ένα τέτοιο πρωτόκολλο, ένας κόμβος προσπαθεί να ανακαλύψει μια διαδρομή προς κάποιο προορισμό **μόνο αν έχει να στείλει κάποιο μήνυμα σε αυτό τον προορισμό**.

Πειραματικά αποτελέσματα έχουν δείξει ότι τα on–demand routing protocols είναι πιο αποδοτικά από τα periodic routing protocols, με την έννοια ότι επειδή διακινούν μικρότερο αριθμό μηνυμάτων τα οποία σχετίζονται με την δρομολόγηση όταν δεν υπάρχει μεγάλη διακίνηση μηνυμάτων στο δίκτυο, έτσι χρησιμοποιούν λιγότερους πόρους του δικτύου. Άκόμα επειδή στα on–demand routing protocols οι διαδρομές ανακαλύπτονται μόλις κάθε κόμβος θελήσει να στείλει κάποιο μήνυμα, αυτά τα πρωτόκολλα προσαρμόζονται πιο γρήγορα σε αλλαγές της τοπολογίας του δικτύου.

2.3 Routing Protocols για Ασύρματα Δίκτυα

Από τα πρωτόκολλα που θα παρουσιάσουμε παρακάτω, τα δύο πρώτα από αυτά Destination-Sequenced Distance Vector Routing (DSDV) και Secure Efficient Distance Vector Routing (SEAD) είναι proactive, ενώ τα Dynamic Source Routing (DSR) και ARIADNE είναι reactive, και τέλος θα παρουσιάσουμε συνοπτικά το periodic πρωτόκολλο Optimized Link State Routing (OLSR) και το reactive πρωτόκολλο Secure Routing Protocol (SRP).

2.3.1 Τα πρωτόκολλα DSDV και DSDV–SQ

Το πρωτόκολλο DSDV, όπως φαίνεται και από την ονομασία του, είναι ένα πρωτόκολλο που βασίζεται στη μέθοσο DISTANCE VECTOR ROUTING που παρουσιάσαμε σε προηγούμενη ενότητα. Όπως και στο DISTANCE VECTOR ROUTING, το DSDV απαιτεί από τους κόμβους να ανταλάσσουν το routing table τους στους γείτονες τους και τα μηνύματα αυτής της διαδικασίας ονομάζονται routing advertisements ή routing updates. Η διαφορά όμως εδώ σε σχέση με το DISTANCE VECTOR

ROUTING είναι στη μορφή των routing tables του κάθε κόμβου. Έτσι, στο DSDV σε κάθε γραμμή του routing table του κάθε κόμβου υπάρχει και ένας αριθμός ο οποίος ονομάζεται **αριθμός σειράς (sequence number)**, και ουσιαστικά χρησιμεύει στο να αποτρέψει κάποιον κόμβο από το να ενημερώσει το routing table του με βάση κάποιο routing advertisement που έλαβε από γείτονά του το οποίο όμως δεν είναι πια έγκυρο γιατί το έλαβε πολύ καθυστερημένα (αυτό μπορεί να συμβεί πολύ εύκολα σε ένα ασύρματο δίκτυο, επειδή τα μηνύματα συνήθως αποστέλονται σε πολλούς κόμβους και ακολουθούν πολλές διαφορετικές διαδρομές). Κάθε κόμβος λοιπόν έχει ένα **άρτιο** sequence number τον οποίο στέλνει μαζί με κάθε routing update, και σε κάθε γραμμή στο routing table του έχει αποθηκεύσει τον πιο πρόσφατο sequence number που πήρε από το routing update μήνυμα με βάση το οποίο ενημέρωσε την γραμμή.

Όταν ένας κόμβος λάβει ένα routing update μήνυμα, τότε ενημερώνει κάποια εγγραφή στο routing table του αν και μόνο αν το sequence number του routing update μηνύματος είναι μεγαλύτερο από το sequence number της γραμμής του πίνακα. Αν τύχει και τα δύο sequence numbers είναι ίσα, τότε στον πίνακα γράφεται εκείνη η εγγραφή (για τον ίδιο προορισμό) για την οποία το ΚΟΣΤΟΣ είναι μικρότερο. (Προφανώς, αν το sequence number του routing update μηνύματος είναι μικρότερο από το sequence number της γραμμής του πίνακα, τότε το μήνυμα αγνοείται)

Όταν ένας κόμβος διαπιστώσει ότι κάποια σύνδεση προς ένα γείτονά του που βρίσκεται στη στήλη ΕΠΟΜΕΝΟΣ ΚΟΜΒΟΣ σε κάποια διαδρομή του routing table του δεν είναι πια έγκυρη, τότε στέλνει ένα routing update μήνυμα το οποίο περιέχει την γραμμή που αναφέρεται σε αυτή τη σύνδεση, αφού έχει αντικαταστήσει το ΚΟΣΤΟΣ με την τιμή άπειρο ∞ και το sequence number με τον αμέσως επόμενο **περιττό** αριθμό. Μόλις ένας κόμβος λάβει ένα τέτοιο routing update (με περιττό sequence number), αν έχει κάποια σωστή εγγραφή με μεγαλύτερο sequence number στέλνει αμέσως ένα routing update με την διαδρομή αυτή. Είναι φανερό ότι, επειδή τα routing updates που δηλώνουν σωστές διαδρομές έχουν άρτιους αριθμούς και τα routing updates που δηλώνουν προβληματικές διαδρομές έχουν περιττούς αριθμούς, τα routing updates τα οποία περιέχουν "σωστές" διαδρομές θα μπορούν πάντα να χρησιμοποιηθούν για ενημέρωση των routing tables που έχουν ελλιπή στοιχεία (δηλαδή διαδρομές με ∞).

Τέλος, αναφέρουμε μια παραλλαγή του προτοκόλλου που ονομάζεται DESTINATION–SEQUENCED DISTANCE VECTOR ROUTING WITH SEQUENCE NUMBERS (DSDV–SQ). Σε αυτή την παραλλαγή, κάθε κόμβος στέλνει ένα routing update μήνυμα **και κάθε φορά που αλλάζει κάποιο sequence number στο routing table του, σε αντίθεση με το να στέλνει routing updates μόνο ανά τακτά χρονικά διαστήματα**.

2.3.2 Το πρωτόκολλο SEAD

Το πρωτόκολλο SEAD είναι ένα secure proactive routing protocol που βασίζεται πάνω στο πρωτόκολλο DSDV–SQ και χρησιμοποιεί αλγόριθμους συμμετρικής κρυπτογράφησης. Η βασικός στόχος που θέλει να πετύχει το πρωτόκολλο είναι να κάνει τα routing update μηνύματα αυθεντικά, και ακόμα ο παραλήπτης κάποιου τέτοιου μηνύματος να μπορεί να εξακριβώσει ότι έρχεται από τον κόμβο που πι-

στεύει ότι προέρχεται. Αυτό γίνεται με την χρήση hash chains και την προσθήκη μιας hash value σε κάθε εγγραφή του routing update μήνυματος.

Έτσι, σε αυτό το πρωτόκολλο υποθέτουμε ότι η διάμετρος του δικτύου (δηλαδή η μεγίστη δυνατή διαδρομή μεταξύ των κόμβων) είναι $m-1$, επομένως σε όλα τα routing updates οι τιμές της στήλης ΚΟΣΤΟΣ θα είναι μικρότερες από m. Κάθε κόμβος φτιάχνει μια one–way hash chain, διαλέγοντας μια τυχαία αρχική τιμή $x \in \{0,1\}^\rho$, όπου ρ είναι το πλήθος των bits που έχουν οι τιμές που παίρνουμε από την συνάρτηση hash H που χρησιμοποιούμε. Έπειτα, ο κόμβος υπολογίζει τις τιμές $h_0, h_1, h_2, \ldots, h_n$, όπου $h_0 = x$ και $h_i = H(h_{i-1})$, για $0 < i \leq n$, για κάποιο n το οποίο διαιρείται με το m. Ο κόμβος χρησιμοποιεί αυτές τις τιμές στα routing updates του, από τα αριστερά προς τα δεξιά, όπως θα περιγραψουμε παρακάτω. Τέλος, υποθέτουμε ότι η τιμή h_n της hash chain που έχει δημιουργήσει ο κάθε κόμβος έχει μοιραστεί σε όλους τους άλλους χρησιμοποιώντας κάποια ασφαλή μέθοδο.

Η μέθοδος που χρησιμοποιείται στο SEAD για να πιστοποιηθεί η αυθεντικότητα μιας εγγραφής σε κάποιο routing update χρησιμοποιεί το sequence number που υπάρχει σε αυτή την εγγραφή για να καθορίσει μια ομάδα από m συνεχόμενες τιμές της hash chain που αντιστοιχεί σε αυτό τον προορισμό, και ένα αντικείμενο αυτής της ομάδας θα χρησιμοποιηθεί για την πιστοποίηση της αυθεντικότητας του μηνύματος. Το ποιό ακριβώς στοιχείο της ομάδας θα είναι αυτό καθορίζεται από την τιμή ΚΟΣΤΟΣ της εγγραφής. Έτσι, αν η hash chain ενός κόμβου είναι η $h_0, h_1, h_2, \ldots, h_n$ και το n διαιρείται με το m, τότε για το sequence number i θέτουμε $k = \frac{n}{m} - i$ και ένα από τα στοιχεία της ομάδας $h_{km}, h_{km+1}, \ldots, h_{km+m-1}$ θα χρησιμοποιηθεί για να την πιστοποίηση της αυθεντικότητας. Αν τώρα η τιμή του πεδίου ΚΟΣΤΟΣ είναι j, με $0 \leq j < m$, τότε το στοιχείο της ομάδας που θα χρησιμοποιηθεί είναι το h_{km+j}.

Όταν ένας κόμβος στέλνει ένα routing update, αυτός συμπεριλαμβάνει μαζί με κάθε εγγραφή του πίνακα του routing update και μια hash value. Εαν ο κόμβος περιλαμβάνει μια εγγραφή με τον εαυτό του στο μήνυμα, τότε θέτει το ΚΟΣΤΟΣ στην τιμή 0, το sequence number στο επόμενο δικό του sequence number, και την hash value στο πρώτο στοιχείο της ομάδας της hash chain του που αντιστοιχεί σε αυτό το sequence number. Σε σχέση με το παράδειγμα που δώσαμε στην προηγούμενη παράγραφο, αν το sequence number αυτής της εγγραφής είναι i, το hash value θα είναι το στοιχείο h_{km}. Επιπρόσθετα, αν ο κόμβος περιλαμβάνει στο routing update μια εγγραφή για κάποιο άλλο προορισμό (εκτός από τον εαυτό του), τότε σε αυτή την εγγραφή βάζει για ΚΟΣΤΟΣ και sequence number τις τιμές που διαθέτει για αυτό τον προορισμό, και ως hash value την hash value που υπήρχε στο routing update μήνυμα από το οποίο έμαθε αυτή την διαδρομή για εκείνο τον προορισμό.

Επειδή το στοιχείο της hash chain που χρησιμοποιούμε εξαρτάται και από την τιμή sequence number και από την τιμή ΚΟΣΤΟΣ και είναι υπολογιστικά αδύνατον να υπολογιστεί η τιμή $H^{-1}(h_{km})$ (και άρα να υπολογιστούν και τα υπόλοιπα στοιχεία της αλυσίδας), δεν μπορεί κανένας κόμβος να αλλοιώσει το μήνυμα routing update ώστε μια ή περισσότερες εγγραφές να γράφουν μικρότερες τιμές στη στήλη ΚΟΣΤΟΣ ή να έχουν μεγαλύτερο sequence number.

Όταν τώρα ένας κόμβος λάβει ένα routing update μήνυμα, τότε αυτός ελέγχει αν κάθε εγγραφή του μηνύματος είναι αυθεντική, με βάση τις τιμές sequence

number και ΚΟΣΤΟΣ και το τελευταίο hash value της hash chain του κόμβου που έχει λάβει σε προηγούμενο μήνυμα και γνωρίζει ότι είναι αυθεντικό. Έτσι, αν ο κόμβος βασιστεί στις τιμές sequence number και ΚΟΣΤΟΣ του μηνύματος που έλαβε για το hash value το οποίο θεωρεί αυθεντικό, και τις τιμές sequence number και ΚΟΣΤΟΣ του μηνύματος που έλαβε αργότερα και θέλει να ελέγξει αν το hash value του είναι αυθεντικό, τότε σύμφωνα με την περιγραφή δύο παραγράφους πριν, αυτός μπορεί να υπολογίσει πόσες φορές πρέπει να εφαρμόσει την συνάρτηση hash H στο hash value του δεύτερου μηνύματος ώστε να πάρει την hash value που έχει λάβει νωρίτερα και γνωρίζει ότι είναι αυθεντική. Αν αυτό συμβεί, τότε και η δεύτερη hash value που έλαβε είναι αυθεντική, και επομένως το ίδιο ισχύει και για όλο το routing update μήνυμα.

2.3.3 Το πρωτόκολλο OLSR

Αυτό το πρωτόκολλο είναι ένα periodic routing protocol το οποίο βασίζεται στην μέθοδο link state routing. Το OLSR χρησιμοποιεί HELLO μηνύματα για να μπορέσει ο κάθε κόμβος να ανακαλύψει τους γείτονές του και την κατάσταση των συνδέσεων του με αυτούς.

Έτσι, κάθε κόμβος A αναμεταδίδει ανά τακτά χρονικά διαστήματα ένα HELLO μήνυμα το οποίο περιέχει τα εξής:

- Την λίστα με τους γείτονες του κόμβου A με τους οποίους έχει **αμφίδρομη σύνδεση**.
- Την λίστα με τους γείτονικούς κόμβους από τους οποίους ο A έχει λάβει ένα HELLO μήνυμα, αλλά οι δύο κόμβοι δεν έχουν επικυρώσει ακόμα ότι η σύνδεση μεταξύ τους είναι αμφίδρομη. Έτσι, αν ένας κόμβος βρει τον εαυτό του σε αυτή τη λίστα, τότε καταλαβαίνει ότι η σύνδεση του με τον A είναι αμφίδρομη.

Με αυτό τον τρόπο, κάθε κόμβος ξέρει τους κόμβους που βρίσκονται σε απόσταση μέχρι δύο βημάτων (hops).

2.3.4 Το πρωτόκολλο DSR

Το πρωτόκολλο DSR αποτελείται από δύο βασικές διαδικασίες, οι οποίες ονομάζονται Route Discovery και Route Maintenance.

Όταν ένας κόμβος θέλει να στείλει ένα μήνυμα σε κάποιον άλλο κόμβο και δεν έχει στη προσωρινή μνήμη διαδρομών (route cache) καμία διαδρομή προς αυτό τον προορισμό, τότε ξεκινά ένα Route Discovery, το οποίο είναι μια διαδικασία εύρεσης διαδρομής. Ο κόμβος που ξεκινά την διαδικασία ονομάζεται *initiator* της Route Discovery ενώ ο κόμβος στον οποίο θέλουμε να καταλήξει το μήνυμα ονομάζεται *target*. Ο initiator κάνει broadcast ένα μήνυμα που ονομάζεται Route Request και περιέχει αρχικά τον target κόμβο και ένα μοναδικό αριθμό (unique identifier) ο οποίος χρησιμοποιείται για την αναγνώριση αυτού του Route Request. Κάθε κόμβος ο οποίος λαμβάνει το Route Request, εκτελεί μια από τις δυο ακόλουθες ενέργειες:

- Αν έχει ξαναλάβει πάλι ROUTE REQUEST από τον initiator με το ίδιο unique identifier, τότε αγνοεί το μήνυμα (επειδή το έχει ξαναλάβει πιο πριν).

- Αν δεν έχει ξαναλάβει αυτό το ROUTE REQUEST, τότε ο κόμβος προσθέτει τον εαυτό του σε μια λίστα κόμβων που υπάρχει στο ROUTE REQUEST μήνυμα και το κάνει broadcast.

Όταν το ROUTE REQUEST φτάσει στον προορισμό του target, τότε ο target στέλνει ένα μήνυμα ROUTE REPLY στον initiator (χρησιμοποιώντας ως διαδρομή π.χ. τη αντίστροφη λίστα των κόμβων που περιέχει το μήνυμα ROUTE REQUEST). Το μήνυμα ROUTE REPLY περιέχει την λίστα με όλους τους κόμβους του ROUTE REQUEST, η οποία ουσιαστικά είναι και η διαδρομή προς στον target η οποία ανακαλύφθηκε. Μόλις ο initiator λάβει το ROUTE REPLY μήνυμα, αποθηκεύει την διαδρομή που περιέχει στη route cache του.

Από την άλλη μεριά, η ROUTE MAINTENANCE είναι η διαδικασία σύμφωνα με την οποία ένας κόμβος ελέγχει αν η διαδρομή στην οποία στέλνει ένα μήνυμα σε κάποιον προορισμό είναι έγκυρη μέχρι το τέλος της και άρα δεν έχει αλλάξει, το οποίο μπορεί να συμβεί αν για παράδειγμα δύο κόμβοι αυτής της διαδρομής έχουν μετακινηθεί και είναι πια εκτός εμβέλειας. Σύμφωνα με αυτή τη διαδικασία λοιπόν, κάθε κόμβος ο οποίος προωθεί ένα μήνυμα βασιζόμενος σε κάποια διαδρομή (που έχει ανακαλυφθεί προηγουμένως) είναι υπεύθυνος για να ελέγξει ότι ο επόμενος κόμβος στη διαδρομή όντως έλαβε το μήνυμα. Έτσι, το μήνυμα ξαναστέλνεται στον επόμενο κόμβο μέχρι να σταλεί μια βεβαίωση ότι αυτός το έλαβε ή μέχρι αυτή η διαδικασία να φτάσει των μέγιστο αριθμό προσπαθειών που έχει καθοριστεί από πριν. Αυτή η βεβαίωση ότι ο επόμενος κόμβος έλαβε το μήνυμα συνήθως παρέχεται από το Medium Access Control protocol που χρησιμοποιείται, όπως για παράδειγμα το IEEE 802.11 που χρησιμοποιείται στα ασύρματα δίκτυα υπολογιστών. Αν μια τέτοια βεβαίωση δεν σταλεί και έχουμε φτάσει στον μέγιστο αριθμό προσπαθειών, τότε ο κόμβος ο οποίος δεν έλαβε την βεβαίωση στέλνει στον αρχικό κόμβο που έστειλε το μήνυμα ένα ROUTE ERROR, το οποίο είναι ένα μήνυμα που περιέχει εκείνη την σύνδεση μεταξύ των κόμβων στην οποία ήταν αδύνατο να προωθηθεί το μήνυμα. Ο αρχικός κόμβος τότε σβήνει αυτή την διαδρομή από την route cache του, και για να στείλει το μήνυμα που ήθελε ή το στέλνει με βάση κάποια άλλη διαδρομή που έχει στη route cache του, ή, αν δεν υπάρχει καμία άλλη τέτοια, τότε ξεκινάει μια διαδικασία ROUTE DISCOVERY για να βρει καινούργια διαδρομή.

2.3.5 Το πρωτόκολλο ARIADNE

Το πρωτόκολλο ARIADNE είναι ένα secure on–demand routing protocol που βασίζεται πάνω στο προηγούμενο πρωτόκολλο DSR και χρησιμοποιεί αλγόριθμους συμμετρικής κρυπτογράφησης. Με το πρωτόκολλο αυτό ο κόμβος target μιας διαδικασίας ROUTE DISCOVERY μπορεί να εξακριβώσει ότι τα μήνυμα ROUTE REQUEST που λαμβάνει είναι αυθεντικά, δηλαδή ότι κανένας κόμβος δεν μπορεί να αλλοιώσει στην πορεία τα στοιχεία που πρόσθεσαν προηγούμενοι κόμβοι στο ROUTE REQUEST. Ακόμα, ο initiator μπορεί να εξακριβώσει ότι τα μηνύματα

ROUTE REPLY που δέχεται είναι αυθεντικά, δηλαδή ότι κανένας κόμβος δεν μπορεί να αλλάξει κάποια στοιχεία του μηνύματος πριν το προωθήσει στον initiator. Έτσι, το πρωτόκολλο ARIADNE προστατεύει ενάντια σε εχθρικούς κόμβους οι οποίοι θέλουν να αλλάξουν ή να αλλοιώσουν τις πληροφορίες δρομολόγησης, ή ακόμα σε εχθρικούς κόμβους οι οποίοι θέλουν να πάρουν την ταυτότητα κάποιου άλλου.

Έστω ότι ο κόμβος S θέλει να βρει μια διαδρομή προς τον D. Σε αυτό το πρωτόκολλο στο μήνυμα ROUTE REQUEST που θα σταλεί από τον S έχουν προστεθεί μια hash chain και μια λίστα από message authentication codes (MACs). Αρχικά η hash chain εχει την τιμή $\text{MAC}_{SD}(initiator, target, id, timestamp)$, όπου το id είναι το unique identifier που αναφέραμε στο πρωτόκολλο DSR, το timestamp είναι η ώρα της αποστολής και το SD είναι ένα κλειδί που μοιράζονται οι κόμβοι S και D, και η λίστα από MACs είναι κενή. Κάθε κόμβος N ο οποίος λαμβάνει το ROUTE REQUEST, εκτελεί μια από τις δυο ακόλουθες ενέργειες:

- Αν έχει ξαναλάβει πάλι ROUTE REQUEST από τον initiator με το ίδιο unique identifier, τότε αγνοεί το μήνυμα (επειδή το έχει ξαναλάβει πιο πριν).
- Αν δεν έχει ξαναλάβει αυτό το ROUTE REQUEST, τότε ο κόμβος προσθέτει τον εαυτό του στη λίστα κόμβων που υπάρχει όπως στο DSR, αντικαθιστά την hash chain με την τιμή $H[N, \langle hash\ chain \rangle]$ (όπου $hash\ chain$ είναι η προηγούμενη τιμή της hash chain), προσθέτει στη λίστα από MACs ένα message authentication code όλου του μηνύματος ROUTE REQUEST χρησιμοποιώντας ένα κλειδί που του έχει μοιραστεί από πριν και είναι έγκυρο για το διάστημα που γίνεται η διαδικασία, και τέλος κάνει broadcast το μήνυμα που θα προκύψει.

Όταν ο target κόμβος θα λάβει το ROUTE REQUEST μήνυμα, ελέγχει ότι η hash chain είναι ίση με

$$H[k_n, H[k_{n-1}, H[\ldots, H[k_1, MAC_{SD}(initiator, target, id, timestamp)] \ldots]]]$$

όπου $k_n, k_{n-1}, \ldots, k_1$ είναι η λίστα των κόμβων που περιέχει το ROUTE REQUEST. Αν αυτό ισχύει, τότε το μήνυμα ROUTE REQUEST δεν έχει αλλοιωθεί στην πορεία από κάποιο κόμβο και είναι έγκυρο, οπότε ο target κόμβος στέλνει πίσω στον initiator ένα ROUTE REPLY μήνυμα όπως και στο DSR. Επιπλέον, σε αυτό το μήνυμα έχουν προστεθεί η λίστα από MACs του ROUTE REQUEST, ένα MAC του μηνύματος Route Reply που έχει υπολογιστεί με το κοινό (για τους κόμβους S και D) κλειδί DS και μια λίστα κλειδιών κρυπτογραφίας που αρχικά είναι κενή. Κατόπιν το μήνυμα στέλνεται στον initiator μέσω των κόμβων από τους οποίους έφτασε στον target, δηλαδή αντιστρέφοντας την λίστα κόμβων του ROUTE REQUEST, όπως στο DSR. Κάθε ενδιάμεσος κόμβος της διαδρομής που προωθεί το μήνυμα στον initiator, πριν το προωθήσει περιμένει για ένα χρονικό διάστημα (αν χρειάζεται), ώστε το κλειδί το οποίο χρησιμοποίησε πριν για να δημιουργήσει το MAC που πρόσθεσε στο ROUTE REQUEST μήνυμα να πάψει να είναι έγκυρο (και άρα να μπορεί να αποκαλυφθεί), και μόλις συμβεί αυτό το προσθέτει στη λίστα κλειδιών του ROUTE REPLY, και προωθεί το μήνυμα στον επόμενο κόμβο της διαδρομής.

Όταν ο initiator λάβει το Route Reply, ελέγχει ότι όλα τα κλειδιά είναι έγκυρα, ότι η λίστα από MACs είναι έγκυρη (επειδή χρησιμοποιώντας τη λίστα κλειδιών που του δώθηκε μπορεί να αναπαράγει όλα τα στοιχεία της MAC λίστας), και τέλος ελέγχει ότι το MAC του Route Reply μηνύματος είναι επίσης έγκυρο, οπότε ξέρει ότι κανένας κόμβος δεν το έχει αλλοιώσει πριν του το στείλει.

2.3.6 Το πρωτόκολλο SRP

Το πρωτόκολλο SRP είναι και αυτό ένα secure on–demand routing protocol που έχει πολλές ομοιότητες με το πρωτόκολλο DSR, επειδή χρησιμοποιεί και αυτό Route Request μηνύματα. Έτσι, το SRP βασίζεται και αυτό σε συμμετρική κρυπτογράφηση (και προϋποθέτει ότι όλα τα ζεύγη κόμβων μοιράζονται από ένα κλειδί κρυπτογράφησης), και δίνει την δυνατότητα στον initiator του Route Discovery να αναγνωρίζει και να απορρίπτει τα πλαστά Route Reply μηνύματα.

Έτσι, όταν ο κόμβος S θέλει να στείλει ένα Route Request μήνυμα για να βρει μια διαδρομή προς τον κόμβο T, ο S στέλνει μαζί με το Route Request και ένα MAC του Route Request μηνύματος μαζί με το κλειδί $K_{S,T}$ που μοιράζονται οι κόμβοι. Όλοι οι ενδιάμεσοι κόμβοι που προωθούν το Route Request μήνυμα στον προορισμό του μετράνε την συχνότητα των Route Request μηνυμάτων που δέχονται από τους γείτονες τους. Αυτές τις συχνότητες τις χρησιμοποιούν για να βάλουν ένα αριθμό προτεραιότητας σε κάθε κόμβο, και οι αριθμοί προτεραιοτήτων είναι αντιστόφως ανάλογοι με την συχνότητα των Route Request μηνυμάτων που στέλνει ο κάθε κόμβος. Έτσι αν ένας κόμβος στέλνει επίτηδες πολλά Route Request μηνύματα για να καταναλώσει πόρους του δικτύου, αυτός θα εξυπηρετηθεί τελευταίος ή ακόμα και θα αγνοηθεί (ανάλογα με τον αριθμό των μηνυμάτων που στέλνει).

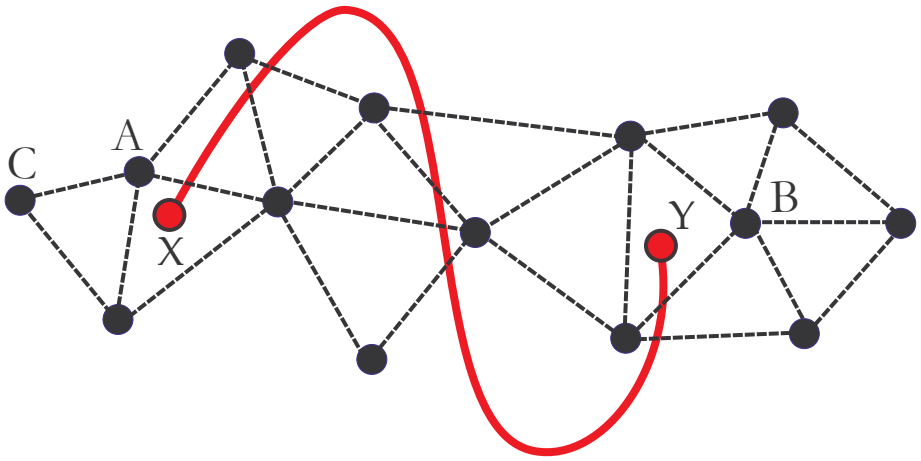
Μόλις ο target κόμβος T λάβει το Route Request μήνυμα, ελέγχει την αυθεντικότητα του υπολογίζοντας το MAC του μηνύματος μαζί με το κλειδί $K_{S,T}$ και ελέγχοντας αν είναι ίσο με το MAC που έλαβε μαζί με το Route Request μήνυμα. Αν το Route Request είναι έγκυρο, ο κόμβος T στέλνει ένα Route Reply μήνυμα στον S χρησιμοποιώντας την διαδρομή που ανακαλύφθηκε και ακόμα στέλνει με το μήνυμα ένα MAC όπως ακριβώς έστειλε ο S με το Route Request. Μόλις αυτό φτάσει στον S, αυτός ελέγχει ότι περιμένει ένα Route Reply μήνυμα με το συγκεκριμένο unique identifier, έπειτα ελέγχει την αυθεντικότητα του μηνύματος με την βοήθεια του MAC που έλαβε (όπως και ο T), και αν το μήνυμα είναι έγκυρο το δέχεται, αλλιώς το απορρίπτει.

Κεφάλαιο 3

Η επίθεση Wormhole

3.1 Περιγραφή της επίθεσης

Η wormhole attack είναι μια επίθεση η οποία στοχεύει στο να μην γίνει σωστά η δρομολόγηση των μηνυμάτων στο ασύρματο δίκτυο, και έπειτα στο να ελέγξει την επικοινωνία μεταξύ όσων περισσοτέρων κόμβων μπορεί. Ας υποθέσουμε ότι έχουμε το ασύρματο δίκτυο του σχήματος 3.1.



Σχήμα 3.1: Wormhole attack

Σε μια wormhole επίθεση, ο εχθρός έχει στην κατοχή του δύο κόμβους X και Y (που δεν είναι ανάγκη να είναι μέρος του ασυρμάτου δικτύου στο οποίο θέλει να επιτεθεί) οι οποίοι συνδέονται μεταξύ τους με ένα σύνδεσμο ο οποίος παρέχει χαμηλό χρόνο αναμονής (latency) και συνήθως υψηλή διαμεταγωγή δεδομένων (bandwith). Ο εχθρός προωθεί ότι μήνυμα συλλαμβάνει ο κόμβος X στον Y και ανάποδα, και με τη σειρά τους οι κόμβοι X και Y αναμεταδίδουν ό,τι μηνύματα λαμβάνει ο ένας από τον άλλο στο υπόλοιπο δίκτυο. Η προώθηση των μηνυμάτων από τον κόμβο X στον Y και αντίστροφα γίνεται χωρίς καμία απολύτως επέμβαση στο περιεχόμενο των μηνυμάτων. Επειδή η σύνδεση μεταξύ των κόμβων X και Y έχει χαμηλό latency, ο χρόνος που απαιτείται για να φτάσει ένα μήνυμα π.χ.

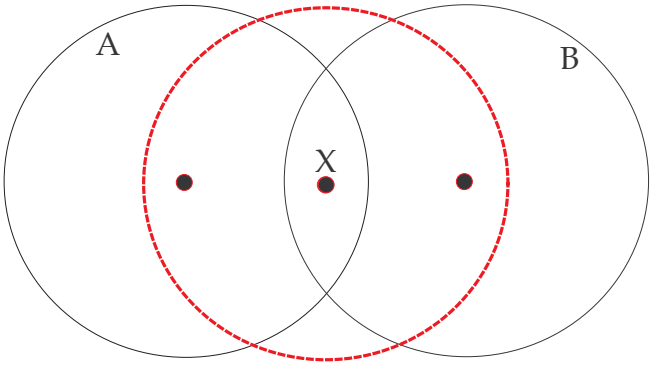
από τον κόμβο A στον κόμβο B αν προωθηθεί μέσω του wormhole $X-Y$ είναι σημαντικά μικρότερος από τον χρόνο που θα χρειαζόταν για να προωθηθεί σιγά σιγά το μήνυμα στον B μέσω κάποιων άλλων κόμβων του αρχικού ασυρμάτου δικτύου. Για αυτό το λόγο και επειδή η προώθηση των μηνυμάτων μέσω του wormhole $X-Y$ γίνεται χωρίς καμία απολύτως επέμβαση στο περιεχόμενο του μηνύματος, οι κόμβοι A και B νομίζουν ότι είναι γειτονικοί κόμβοι. Με αυτό τον τρόπο ο εχθρός κάνει διάφορους μη γειτονικούς κόμβους του δικτύου να νομίζουν ότι είναι γειτονικοί, δηλαδή τους κάνει να νομίζουν ότι η διάταξη των nodes του δικτύου είναι διαφορετική από την πραγματική.

Εκτός όμως από αυτό, ο εχθρός καταφέρνει κάτι ακόμα καλύτερο: να ελέγχει τις επικοινωνίες μεταξύ των κόμβων που ξεγελάει ότι είναι γείτονες, αφού οποιαδήποτε μήνυμα πρόκειται αυτοί να ανταλλάξουν μεταξύ τους θα πρέπει να περάσει από το wormhole $X-Y$. Έτσι, ο εχθρός μπορεί ανα πάσα στιγμή να διακόψει την προώθηση των μηνυμάτων μεταξύ των κόμβων X, Y, με αποτέλεσμα να διακοπεί την επικοινωνία μεταξύ όλων των κόμβων που λανθασμένα πιστεύουν ότι είναι γείτονες, όπως για παράδειγμα οι κόμβοι A και B. Ακόμα καλύτερα, ο εχθρός μπορεί να κάνει το εξής χειρότερο: να προωθεί πάντα όλα τα μηνύματα τα οποία σχετίζονται με όποιο πρωτόκολο δρομολόγησης χρησιμοποιείται για να επικοινωνήσουν μεταξύ τους οι κόμβοι και να απορρίπτει όλα τα data packets, δηλαδή όλα τα μηνύματα τα οποία περιέχουν τις πραγματικές πληροφορίες που θέλουν να ανταλλάξουν μεταξύ τους οι κόμβοι. Έτσι ο εχθρός πετυχαίνει μια μόνιμη επίθεση τύπου Denial of Service, αφού αν π.χ. ο κόμβος C θέλει να επικοινωνήσει με τον B στο σχήμα 3, τότε κάθε φορά που χρησιμοποιεί κάποιο πρωτόκολο δρομολόγησης η συντομότερη διαδρομή που θα ανακαλυφθεί είναι μέσω του A, αφού ο A με τον C είναι γείτονες και ο A νομίζει ότι ο B είναι γειτονικός του κόμβος, και μόλις ο C πάει να στείλει κάποιο μήνυμα το οποίο προορίζεται για τον B, αυτό θα το σταλεί πρώτα στον A, και μόλις πάει να σταλεί στον B μέσω του wormhole το μήνυμα θα απορριφθεί από τον εχθρό και δεν θα φτάσει ποτέ στον προορισμό του.

Γενικότερα, οι wormhole επιθέσεις χρησιμοποιούνται όταν ο εχθρός θέλει να εκμεταλλευτεί καταστάσεις που εμφανίζονται όταν ένας κόμβος εκτελεί κάποια ενέργεια βασισμένος στο πρώτο μήνυμα που θα λάβει για κάποιο συγκεκριμένο θέμα και αγνοεί όλα τα υπόλοιπα μηνύματα που αναφέρονται στο ίδιο θέμα (αυτές οι καταστάσεις ονομάζονται routing race conditions). Σε αυτά τα routing race conditions, ο εχθρός μπορεί να επηρεάσει την τελική τοπολογία του δικτύου αν για παράδειγμα καταφέρει να προωθήσει με τη βοήθεια wormhole σε κάποιους κόμβους συγκεκριμένες πληροφορίες για την δρομολόγηση των μηνυμάτων, όπως για παράδειγμα τα Route Request μηνύματα που χρησιμοποιούν τα on–demand πρωτόκολλα, πολύ πριν την στιγμή που αυτοί θα τις λάμβαναν κανονικά μέσω της πραγματικής διαδρομής στο δίκτυο.

3.2 Παραδείγματα επιθέσεων τύπου wormhole

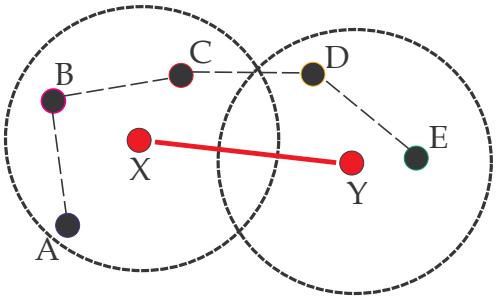
Η πιο απλή επίθεση wormhole είναι η Worawannotai attack η οποία ουσιαστικά είναι μια τετριμμένη έκδοση wormhole, όπου οι δύο άκρες της ταυτίζονται, όπως φαίνεται στο σχήμα 3.2:



Σχήμα 3.2: Worawannotai attack

Σε αυτή την επίθεση ο εχθρικός κόμβος X ουσιαστικά δρα σαν απλώς αναμεταδότης που στέλνει τα μηνύματα του A στον κόμβο B (τα οποία δεν μπορούν να φτάσουν άμεσα στον κόμβο B επειδή ο τελευταίος είναι εκτός εμβέλειας του πρώτου).

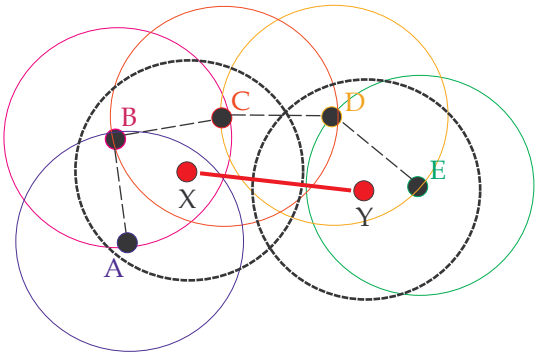
Ένα παράδειγμα μιας μη τετριμμένης επίθεσης wormhole φαίνεται στο επόμενο σχήμα:



Σχήμα 3.3: Παράδειγμα wormhole επίθεσης

Σε αυτό το σχήμα έχουμε τους κόμβους A, B, C, D και E, καθώς και τους κόμβους X και Y που συνδέονται μέσω ενός wormhole και δεν αποτελούν μέρος του αρχικού δικτύου. Οι δύο κύκλοι που σχηματίζονται από τις μαύρες διακεκομμένες γραμμές δείχνουν την εμβέλεια των κόμβων X και Y, ενώ οι μαύρες διακεκομμένες γραμμές μεταξύ των κόμβων A, B, C, D και E περιγράφουν τις συνδέσεις μεταξύ των κόμβων που μπορούν να δημιουργηθούν αν δεν υπάρχει η wormhole X–Y. Το σχήμα 3.4 στην σελίδα 23 δείχνει την εμβέλεια όλων των ασυρμάτων κόμβων του δικτύου.

Οι συνδέσεις λοιπόν που μπορούν να δημιουργηθούν αν δεν υπάρχει η wormhole X–Y όπως φαίνεται από το παραπάνω σχήμα είναι οι A–B, B–C, C–D, D–E. Αν όμως λάβουμε υπόψη μας και τους κόμβους X, Y οι οποίοι συνδέονται μεταξύ τους μεσω wormhole, τότε παρατηρούμε ότι οι κόμβοι A, B και C είναι στην εμβέλεια του κόμβου X, και επομένως οποιαδήποτε μήνυμα εκπέμψουν θα το λάβει και ο κόμβος X, καθώς και οποιαδήποτε μετάδοση από τον κόμβο Y θα ληφθεί από τους κόμβους D και E. Τα ίδια ισχύουν και για τους κόμβους D,

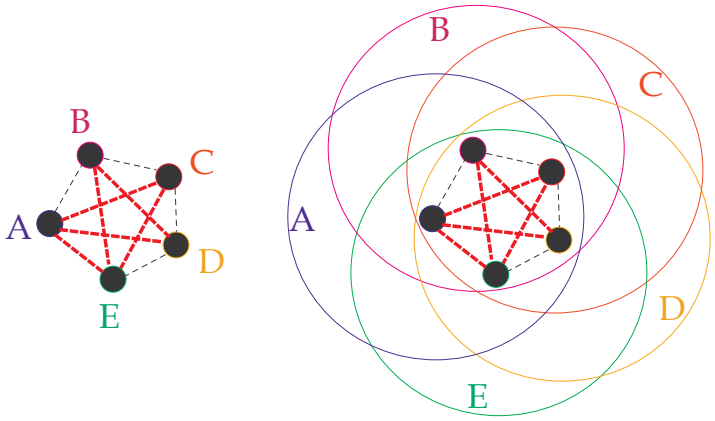


Σχήμα 3.4: Εμβέλεια όλων των κόμβων του δικτύου

E και Y, μιας και οι κόμβοι D και E είναι στην εμβέλεια του κόμβου Y. Έτσι, αφού κάθε μήνυμα που λαμβάνει ο X το προωθεί στον κόμβο Y και αυτός με τη σειρά του το στέλνει σε όσους κόμβους είναι στην εμβέλειά του και αντίστροφα, το αποτέλεσμα είναι οι αποστολές και λήψεις μηνυμάτων μεταξύ των κόμβων A, B, C, D και E να γίνονται ως εξής:

- Τα μηνύματα του A τα λαμβάνουν και οι C, D και E εκτός από τον B
- Τα μηνύματα του B τα λαμβάνουν και οι D και E εκτός από τους A και C
- Τα μηνύματα του C τα λαμβάνουν και οι A και E εκτός από τους B και D
- Τα μηνύματα του D τα λαμβάνουν και οι B και A εκτός από τους C και E
- Τα μηνύματα του E τα λαμβάνουν και οι A, B και C εκτός από τον D

Συνεπώς, η διάταξη του δικτύου όπως την αντιλαμβάνονται οι κόμβοι είναι όπως στο επόμενο σχήμα, όπου με μαύρες διακεκομμένες γραμμές περιγράφονται οι συνδέσεις που μπορούν να δημιουργηθούν απευθείας μεταξύ των κόμβων και με κόκκινες διακεκομμένες γραμμές περιγράφονται οι συνδέσεις που μπορούν να δημιουργηθούν μόνο μέσω της wormhole X–Y:



3.3 Εφαρμογή της επίθεσης σε ορισμένα routing πρωτόκολλα

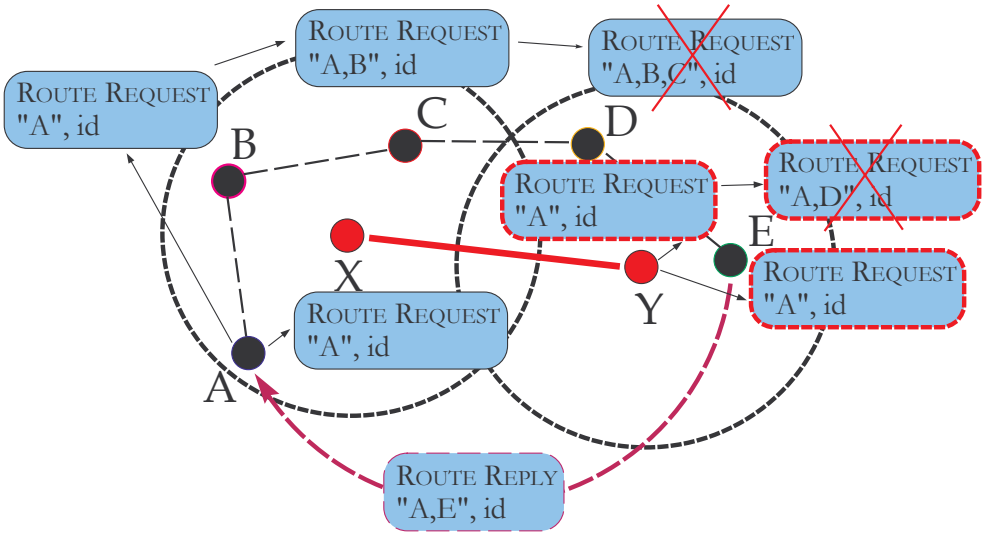
Όπως θα δούμε στη συνέχεια, η επίθεση τύπου wormhole είναι ιδιαίτερα επικίνδυνη όταν εφαρμόζεται εναντίον routing προτοκόλλων για ad hoc δίκτυα, γιατί σε αυτά τα δίκτυα όταν κάποιος κόμβος ακούει ένα μήνυμα από κάποιο άλλο κόμβο τότε θεωρεί αυτόματα ότι ο τελευταίος βρίσκεται στην εμβέλεια του, δηλαδή ότι είναι γειτονικοί κόμβοι.

3.3.1 Επίθεση σε on-demand (reactive) routing πρωτόκολλα

Ας υποθέσουμε ότι έχουμε ένα ad hoc δίκτυο που χρησιμοποιεί ως routing protocol το Dynamic Source Routing protocol (DSR). Τότε μια εφαρμογή της επίθεσης θα είναι να προωθεί ο εχθρός όλα τα Route Request μηνύματα μέσω της worhmole X–Y κατευθείαν στους γειτονικούς κόμβους του προορισμού. Όταν οι γειτονικοί κόμβοι λάβουν τα Route Request μηνύματα θα ακολουθήσουν το πρωτόκολλλο και θα τα προωθήσουν στον κόμβο–προορισμό, απορρίπτοντας όλα τα άλλα Route Request μηνύματα που θα λάβουν αργότερα (σύμφωνα με το πρωτόκολλλο). Με αυτό τον τρόπο ανακαλύπτεται μόνο μια διαδρομή προς τον προορισμό, αυτή που είναι μέσω της wormhole, και ακόμα, αν το ένα άκρο της wormhole είναι αρκετά κοντά στον κόμβο που στέλνει Route Request μηνύματα, τότε αυτό το είδος της επίθεσης μπορεί ακόμα και να εμποδίσει την ανακάλυψη διαδρομών μεγαλύτερων από 2 βήματα από αυτόν τον κόμβο. Κατόπιν, όταν ο αρχικός κόμβος που ζήτησε να βρει δρομολόγιο ξεκινά την αποστολή των μηνύματων που θέλει να στείλει (data packets) στον κόμβο–προορισμό χρησιμοποιώντας την διαδρομή που ανακαλύφθηκε και είναι μέσω της wormhole, ο εχθρός μπορεί να μην προωθεί κανένα από τα αυτά, παρά μόνο όσα είναι τύπου Route Request. Έτσι, τα μηνύματα αυτά θα φαίνονται ότι χάθηκαν, μιας και δεν έφτασαν στον προορισμό τους, και επειδή ο αρχικός κόμβος δεν έχει ανακαλύψει άλλα δρομολόγια προς τον κόμβο–προορισμό, θα ξεκινήσει ένα καινούργιο Route Discovery. Επειδή όμως τα μηνύματα που θα στείλει τώρα είναι τύπου Route Request, ο εχθρός θα τα προωθήσει μέσω της wormhole όπως περιγράψαμε αρχικά, και το αποτέλεσμα είναι ο αρχικός κόμβος να βρίσκει διαδρομές προς τον κόμβο–προορισμό, χωρίς όμως να μπορεί να επικοινωνήσει με αυτόν. Με άλλα λόγια, ο εχθρός πετυχαίνει μια μόνιμη Denial of Service επίθεση από την οποία δεν μπορεί να ξεφύγει ο αρχικός κόμβος, γιατί σε αυτόν φαίνεται ότι υπάρχει απλώς κάποιο πρόβλημα μετάδοσης μερικών από τα μηνύματά του. Εναλλακτικά, αν ο εχθρός μπορεί να διαβάσει το περιεχόμενο των μηνυμάτων πριν τα προωθήσει μέσω της wormhole, δηλαδή αν τα data packets δεν είναι κρυπτογραφημένα, τότε μπορεί ακόμα να μην κάνει Denial of Service αλλά να προωθεί μόνο μερικά από αυτά ή ακόμα και κομμάτια αυτών (αν για παράδειγμα ο στόχος της επίθεσης είναι να μεταδοθούν μερικές αλλά όχι όλες από τις πληροφορίες που αποστέλονται).

Στο σχήμα 3.5 της σελίδας 25 εξετάζουμε ένα παράδειγμα μιας τέτοιας επίθεσης. Σε αυτό το ασύρματο δίκτυο ο κόμβος A θέλει να στείλει ένα μήνυμα στον κόμβο E, οπότε ξεκινάει ένα Route Discovery κάνοντας broadcast Route Request μηνύματα, τα οποία λαμβάνονται από τους κόμβους B και X. Ο κόμ-

βος B ακολουθώντας το πρωτόκολλλο προωθεί το μήνυμα στον κόμβο C κλπ. Ο κόμβος X που ανήκει στον εχθρό θα μεταφέρει το μήνυμα στο άλλο άκρο Y της wormhole, και ο κόμβος Y θα το κάνει broadcast, οπότε θα το λάβουν οι D και E. Ο κόμβος D θα λάβει επίσης και ένα ROUTE REQUEST μήνυμα από τον C λίγο αργότερα το οποίο όμως θα αγνοήσει γιατί θα έχει ήδη λάβει το ROUTE REQUEST μήνυμα του A από την wormhole.



Σχήμα 3.5: ROUTE REQUESTS στο DSR με την παρουσία wormhole

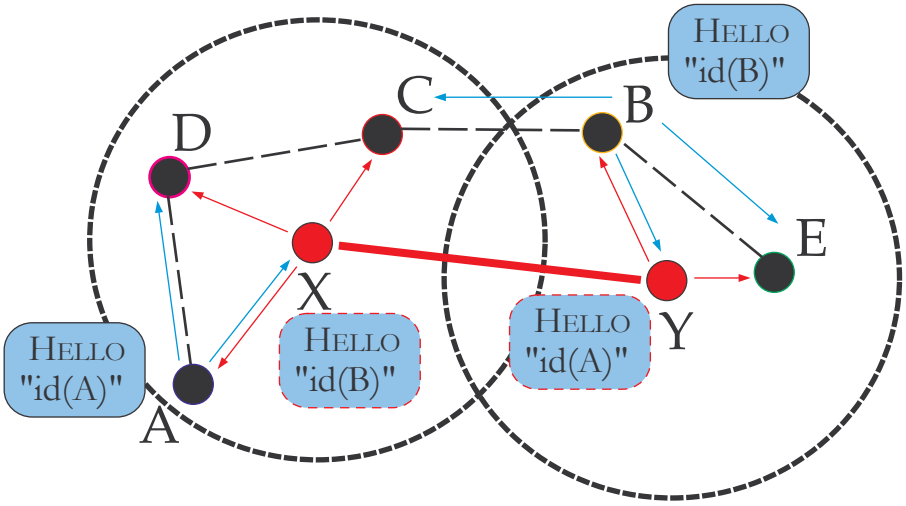
Ακόμα, ο E θα λάβει 2 ROUTE REQUEST μηνύματα (με το ίδιο id πάντα), και το πρώτο που θα λάβει θα είναι από τον A μέσω της wormhole, οπότε και θα απαντήσει με ROUTE REPLY στον A και η διαδρομή θα είναι $A \rightarrow E$ (μέσω της wormhole φυσικά), και έπειτα θα λάβει το ROUTE REQUEST που θα του προωθήσει ο D, το οποίο και θα αγνοήσει και συνεπώς, η μόνη διαδρομή που σχηματίζεται είναι αυτή μέσω της wormhole.

3.3.2 Επίθεση σε periodic (proactive) routing πρωτόκολλλα

Ας υποθέσουμε τώρα ότι έχουμε ένα ασύρματο δίκτυο το οποίο χρησιμοποιεί κάποιο proactive routing πρωτόκολλλο, το οποίο ή χρησιμοποιεί κάποιο μηχανισμό ώστε οι κόμβοι να ανακαλύψουν τους γείτονές τους, όπως για παράδειγμα το OLSR, ή στέλνει routing update μηνύματα όπως το DSDV. Αυτά τα πρωτόκολλα βασίζονται κυρίως στη σωστή λήψη των broadcast μηνυμάτων που στέλνουν οι κόμβοι μεταξύ τους και τα χρησιμοποιούν είτε για να ανακαλύψει κάθε κόμβος τους γείτονές του ή για να ενημερώσει ο κάθε κόμβος το routing table του, και επομένως είναι τρωτά σε wormhole επιθέσεις.

Έτσι, μια wormhole επίθεση στο πρωτόκολλο OLSR που χρησιμοποιεί HELLO μηνύματα για ανακάλυψη γειτόνων είναι πολύ απλή: ο εχθρός απλά προωθεί όλα τα HELLO μηνύματα του κόμβου A στον κόμβο B (που είναι εκτός της εμβέλειας του A) μέσω της wormhole που ελέγχει, και αντίστροφα προωθεί όλα τα HELLO

μηνύματα του κόμβου B στον κόμβο A, με αποτέλεσμα οι A και B να νομίζουν ότι είναι γείτονες. Αυτό όμως δεν ισχύει στην πραγματικότητα, και έτσι το routing πρωτόκολλο δεν θα μπορεί να βρει σωστά δρομολόγια, αφού θα βασίζεται σε λάθος δεδομένα.



Σχήμα 3.6: Επίθεση σε πρωτόκολλλα που χρησιμοποιούν HELLO μηνύματα

Για το πρωτόκολλο DSDV, η επίθεση μπορεί να γίνει ως εξής: ο εχθρός χρησιμοποιεί την wormhole που ελέγχει για να προωθεί κάθε routing advertisement που στέλνει ο κόμβος A στον κόμβο B και αντίστροφα, οπότε οι A και B νομίζουν ότι είναι γείτονες. Εαν όμως, όπως π.χ. στο σχήμα 3.6, οι A και B δεν είναι ο ένας στην εμβέλεια του άλλου, είναι προφανές ότι δεν μπορούν να επικοινωνήσουν. Επιπρόσθετα, μπορούμε να δείξουμε και το εξής: αν η καλύτερη υπάρχουσα έγκυρη διαδρομή από τον A στον B αποτελείται από τουλάχιστον $2n+2$ βήματα, τότε κάθε κόμβος που βρίσκεται από 1 εως n βήματα μακριά του A δεν θα μπορεί να επικοινωνήσει με τον B, και κάθε κόμβος που βρίσκεται από 1 εως n βήματα μακριά του B δεν θα μπορεί να επικοινωνήσει με τον A. Για να το δείξουμε αυτό, ας υποθέσουμε ότι ο κόμβος C απέχει το πολύ n βήματα από τον κόμβο A και μπορεί να επικοινωνήσει με τον κόμβο B, δηλαδή υπάρχει μια έγκυρη διαδρομή από τον C στον B. Επειδή ο A στα routing updates του ανακοινώνει ότι έχει διαδρομή με προορισμό τον B που αποτελείται από 1 βήμα, ο C μόλις λάβει αυτή την ανακοίνωση έχει ένα δρομολόγιο $n+1$ βημάτων για τον κόμβο B. Έτσι, αφού ο C απέχει το πολύ n βήματα από τον κόμβο A, έχουμε ότι μια διαδρομή από τον A στον B μέσω του C μήκους το πολύ $n+n+1 = 2n+1$ βήματα, που είναι άτοπο γιατί υποθέσαμε αρχικά ότι η καλύτερη διαδρομή είναι $2n+2$.

3.3.3 Επίθεση σε secure on-demand και periodic πρωτόκολλα

Ας υποθέσουμε τώρα ότι έχουμε ένα ασύρματο δίκτυο το οποίο χρησιμοποιεί κάποιο secure routing protocol. Μια wormhole επίθεση μπορεί να εφαρμοστεί με επιτυχία και σε αυτή την περίπτωση, ανεξάρτητα από το αν το secure routing

protocol είναι on-demand ή periodic, και παρακάτω θα αναφέρουμε τρία τέτοια παραδείγματα επιθέσεων, στα πρωτόκολλα ARIADNE, SEAD και SRP.

Επίθεση στο πρωτόκολλο ARIADNE

Όπως είδαμε σε προηγούμενη ενότητα, το πρωτόκολλο ARIADNE είναι ένα on-demand routing πρωτόκολλο που βασίζεται στο DSR και χρησιμοποιεί αλγόριθμους συμμετρικής κρυπτογράφησης. Επειδή ακριβώς το ARIADNE βασίζεται στο DSR, μοιράζεται με αυτό την ίδια αδυναμία προστασίας από την wormhole επίθεση. Αυτό γίνεται επειδή τα δεδομένα που προστατεύονται με message authentication codes στα ROUTE REQUEST και ROUTE REPLY μηνύματα είναι **η λίστα των κόμβων που αποτελεί την διαδρομή και όχι ο τύπος του μηνύματος**, αν δηλαδή είναι ROUTE REQUEST, ROUTE REPLY, ROUTE MAINTENANCE, κλπ. Αυτό έχει ως αποτέλεσμα ο εχθρός να μπορεί να καταλάβει αν κάποιος κόμβος έχει ξεκινήσει κάποια διαδικασία ROUTE DISCOVERY και να μπορεί να ξεχωρίσει ποιά μηνύματα ανήκουν σε αυτή την διαδικασία. Έτσι, ένας εχθρός που ελέγχει κάποιο wormhole μπορεί να προωθήσει τα ROUTE REQUEST και ROUTE REPLY που ανταλλάσσονται μεταξύ των κόμβων ακριβώς όπως στην περίπτωση που χρησιμοποιείται το πρωτόκολλο DSR, δηλαδή η επίθεση είναι ουσιαστικά ίδια με αυτήν που περιγράφουμε στην ενότητα 3.3.1 της σελίδας 24.

Επίθεση στο πρωτόκολλο SEAD

Όπως είδαμε σε προηγούμενη ενότητα, το πρωτόκολλο SEAD είναι ένα periodic routing πρωτόκολλο που βασίζεται στο Destination-Sequenced Distance Vector πρωτόκολλο (DSDV). Το SEAD χρησιμοποιεί one–way hash chains για να προστατέψει τα sequence number και ΚΟΣΤΟΣ του κάθε μηνύματος που χρησιμοποιείται για ενημέρωση των routing tables των υπολοίπων κόμβων από αλλοιώσεις που πιθανόν να γίνουν από εχθρικούς κόμβους. Όπως και στην περίπτωση του ARIADNE, επειδή το SEAD βασίζεται στο πρωτόκολλο DSDV και η κρυπτογράφηση των δεδομένων δεν εμποδίζει τον εχθρό να ξεχωρίσει ποιά μηνύματα είναι routing updates, μια wormhole επίθεση στο πρωτόκολλο SEAD είναι ουσιαστικά ίδια με μια επίθεση στο πρωτόκολλο DSDV, όπως την περιγράψαμε στην ενότητα 3.3.2 της σελίδας 26.

Επίθεση στο πρωτόκολλο SRP

Όπως είδαμε πιο πριν σε προηγούμενη ενότητα, το πρωτόκολλο SRP είναι και αυτό ένα πρωτόκολλο το οποίο έχει κατασκευαστεί ώστε να είναι συμβατό με τα on-demand routing πρωτόκολλα, και χρησιμοποιεί και αυτό ROUTE REQUEST και ROUTE REPLY μηνύματα. Όπως και με το πρωτόκολλο ARIADNE, η προστασία των δεδομένων των routing μηνυμάτων που γίνεται με κρυπτογράφηση δεν εμποδίζει τον εχθρό να ξεχωρίσει ποιά μηνύματα είναι ROUTE REQUEST και ποια ROUTE REPLY με αποτέλεσμα να μπορεί να τα προωθήσει κανονικά μέσω της wormhole που ελέγχει και να εφαρμοστεί η επίθεση με επιτυχία.

3.4 Δυνατά σημεία της επίθεσης

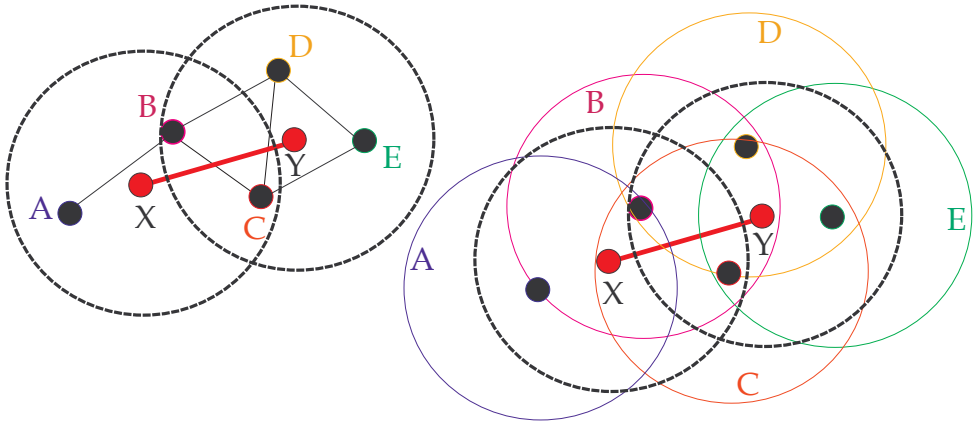
Ολοκληρώνοντας την ανάλυση της επίθεσης wormhole συνοψίζουμε τα δυνατά σημεία της τα οποία την κάνουν πιο εύκολη να εφαρμοστεί από τον εχθρό και πιο δύσκολη να αντιμετωπιστεί από τους κόμβους του ασύρματου δικτύου. Αυτά είναι τα εξής:

1. **Ο εχθρός δεν χρειάζεται να έχει στην κατοχή του ειδικές πληροφορίες για το πως επικοινωνούν μεταξύ τους οι κόμβοι**, όπως για παράδειγμα κάποιο κλειδί κρυπτογραφίας το οποίο πιθανόν να χρησιμοποιείται για να κρυπτογραφηθούν οι επικοινωνίες. Αυτό είναι φανερό, αφού η επίθεση βασίζεται σε απλή προώθηση μηνυμάτων χωρίς να χρειάζεται να διαβαστεί το περιεχόμενό τους.

2. **Η επικοινωνία μέσω wormhole δεν γίνεται αντιληπτή από τους κόμβους που την χρησιμοποιούν**, γιατί η προώθηση των μηνυμάτων γίνεται χωρίς κάποια αλλαγή του περιεχομένου τους.

3. **Μια wormhole δεν επηρεάζει μόνο τις επικοινωνίες των κόμβων που "ξεγελάει" στο να είναι γείτονες, αλλά και τις επικοινωνίες των γειτόνων τους.** Σε πειράματα όπου οι κόμβοι του δικτύου διανέμονται τυχαία σε μια επιφάνεια ενός ορθογωνίου, παρατηρήθηκε ότι μια wormhole με άκρα τα οποία έχουν τοποθετηθεί τυχαία επηρεάζει αρνητικά πάνω από το 5% **ΟΛΩΝ** των διαδρομών (routes) μεταξύ των κόμβων του δικτύου.

   Ακόμα, επειδή η σύνδεση μεταξύ των 2 κόμβων που αποτελούν μια wormhole είναι συνήθως πολύ γρήγορη (λόγω του χαμηλού latency), είναι φανερό ότι ένας πιο έξυπνος εχθρός θα μπορούσε να τοποθετήσει τα άκρα της wormhole του σε συγκεκριμένες τοποθεσίες έτσι ώστε να ελέγξει σχεδόν όλες τις δυνατές επικοινωνίες μεταξύ ενός κόμβου (και ίσως και περισσοτέρων) και των υπόλοιπων κόμβων του δικτύου.



Σχήμα 3.7: Απομόνωση ενός κόμβου υπό την παρουσία wormhole (στο αριστερό σχήμα φαίνεται η εμβέλεια της wormhole ενώ στο δεξί σχήμα φαίνεται και η εμβέλεια του κάθε κόμβου)

Για παράδειγμα, στο σχήμα 3.7 βλέπουμε ότι ο κόμβος A μπορεί να επικοινωνήσει με τον κόμβο B απευθείας. Αν τώρα δεν υπήρχε η wormhole X–Y, ο κόμβος A θα επικοινωνούσε και με τους κόμβους C, D και E μέσω του κόμβου B. Όταν όμως υπάρχει η wormhole X–Y τότε ο A θα επικοινωνήσει με τον κόμβο B που είναι στην εμβέλειά του, αλλά θα φανεί σε αυτόν ότι **και οι κόμβοι C, D και E είναι μέσα στην εμβέλειά του** (μιας και ό,τι μηνύματα στέλνει θα προωθηθούν μέσω της wormhole), οπότε θα επικοινωνεί και με αυτούς "κατευθείαν". Στην πραγματικότητα όμως θα μπορεί να επικοινωνήσει μόνο με τον κόμβο B (αν ο εχθρός διακόψει την προώθηση μηνυμάτων) με αποτέλεσμα ο αριθμός των κόμβων που μπορεί να επικοινωνήσει ο A να μειώνεται από τους 4 στους 1 (δηλαδή μείωση 75%), και ακόμα χειρότερα, να απομονώνεται από τους υπόλοιπους κόμβους του δικτύου.

4. Ο εχθρός είναι δυνατόν να μπορεί σε κάποιες περιπτώσεις να διαβάζει το περιεχόμενο των μηνυμάτων που προωθεί (όταν για παράδειγμα το περιεχόμενο των μηνυμάτων δεν είναι κρυπτογραφημένο), το οποίο σημαίνει ότι **θα μπορεί να διαβάζει ΟΛΑ τα μηνύματα αυτών των επικοινωνιών μεταξύ των κόμβων** που μεσολαβεί, γιατί αν υπάρχει μια διαδρομή μέσω κάποιας wormhole τότε όπως είδαμε αυτή είναι πιο σύντομη και έτσι χρησιμοποιείται μόνο αυτή.

5. Όπως είδαμε, η επίθεση πραγματοποιείται ακόμα και όταν χρησιμοποιείται κάποιο secure on-demand routing protocol ή secure periodic routing protocol.

3.5 Byzantine Wormhole Attacks

Οι επιθέσεις στα ασύρματα δίκτυα στις οποίες ο εχθρός έχει τον πλήρη έλεγχο ενός ή και περισσοτέρων κόμβων **οι οποίοι όμως αποτελούν μέρος του αρχικού δικτύου (authenticated nodes)**, δεν έχουν δηλαδή προστεθεί στο δίκτυο από τον εχθρό, ονομάζονται **Byzantine επιθέσεις**. Τελειώνοντας αυτό το κεφάλαιο, θα αναφέρουμε δυο παραλλαγές της επίθεσης wormhole, οι οποίες είναι η επίθεση *Byzantine Wormhole* και η επίθεση *Byzantine Overlay Wormhole*.

3.5.1 Byzantine Wormhole Επίθεση

Η Byzantine Wormhole επίθεση είναι μια επίθεση wormhole στην οποία όμως οι δύο κόμβοι που την σχηματίζουν και ελέγχονται από τον εχθρό είναι authenticated nodes. Σε αυτή η επίθεση ο εχθρός μπορεί να χρησιμοποιήσει και το ήδη υπάρχον δίκτυο για να δημιουργήσει την wormhole, δηλαδή για παράδειγμα μπορεί να ξεκινήσει ένα Route Discovery για να βρει διαδρομή από τον ένα κόμβο στον άλλο και έπειτα να "διαφημίζει" ότι αυτοί οι δύο κόμβοι είναι γειτονικοί, για να πετύχει να προωθούνται μηνύματα μέσω αυτής της σύνδεσης που έχει πετύχει. Φυσικά, τα παραπάνω δεν αποκλείουν το γεγονός να χρησιμοποιήσει ο εχθρός άλλα μέσα για να δημιουργήσει την wormhole, όπως για πράδειγμα μια γρήγορη ενσύρματη σύνδεση, κατευθυνόμενες κεραίες, κλπ.

3.5.2 Byzantine Overlay Wormhole Επίθεση

Αυτή η επίθεση είναι παρόμοια με την Byzantine Wormhole επίθεση που περιγράψαμε πριν, με την διαφορά ότι εδώ ο εχθρός ελέγχει παραπάνω από δυο authenticated κόμβους, τους οποίους χρησιμοποιεί για να δημιουργήσει ένα overlay δίκτυο. Όταν λέμε ότι αυτοί οι κόμβοι σχηματίζουν ένα overlay δίκτυο ουσιαστικά εννοούμε αυτοί σχηματίζουν ένα δικό τους "ιδιωτικό" δίκτυο χρησιμοποιώντας πιθανόν συνδέσεις και κόμβους του αρχικού δικτύου (και έτσι φαίνεται σαν αυτό το overlay δίκτυο να επικαλύπτει το αρχικό). Ο εχθρός χρησιμοποιεί αυτό το overlay δίκτυο για να δημιουργήσει όλες τις δυνατές συνδέσεις μεταξύ των κόμβων που έχει καταλάβει, και έπειτα κάνει να φαίνεται στους υπόλοιπους κόμβους (και άρα και στο routing protocol που χρησιμοποιείται) ότι όλοι οι κόμβοι που ελέγχει είναι γείτονες. Με αυτό τον τρόπο ο εχθρός αυξάνει σημαντικά τις πιθανότητες να προωθούνται τα μηνύματα μέσω του overlay δικτύου που ελέγχει, τα οποία μπορεί έπειτα να διαχειριστεί όπως και στην απλή wormhole επίθεση. Αυτή είναι και η πιο δυνατή wormhole επίθεση που έχει περιγραφεί μέχρι στιγμής.

Στο κεφάλαιο που ακολουθεί παρουσιάζουμε κάποιες μεθόδους αντιμετώπισης των wormhole επιθέσεων.

Κεφάλαιο 4

Αντιμετωπίζοντας τις επιθέσεις Wormhole

Όπως είδαμε παραπάνω οι επιθέσεις με wormholes βασίζονται στο γεγονός ότι γενικά οι κόμβοι ενός ασύρματου δικτύου δεν γνωρίζουν την ακριβή θέση τους σε σχέση με τους γείτονές τους. Για αυτόν ακριβώς το λόγο, πρωτόκολλα δρομολόγησης τα οποία βασίζονται σε πληροφορίες που σχετίζονται με τις θέσεις του κάθε κόμβου είναι δυνατόν να προστατέψουν το δίκτυο από τετοιες επιθέσεις. Έχουν προταθεί τρεις κύριες μέθοδοι για την αντιμετώπιση wormholes:

1. Η χρήση των **packet leashes**, ο οποίος είναι ένας τρόπος να περιορίσουμε την μέγιστη απόσταση που μπορεί να διανύσει ένα μήνυμα πριν παραδοθεί στον προορισμό του.

2. Η χρήση του πρωτοκόλλου On–Demand Secure Byzantine Routing (ODSBR), το οποίο αντιμετωπίζει και τις παραλλαγές Byzantine Wormhole και Byzantine Overlay Wormhole της "κλασσικής" wormhole επίθεσης.

3. Η χρήση **directional antennas με συγκεκριμένους αλγορίθμους ανίχνευσης γειτόνων**, οι οποίοι χρησιμοποιούν κατευθυνόμενες κεραίες για να εντοπίσουν από ποιά κατεύθυνση γίνεται η λήψη των μηνυμάτων, και κατόπιν χρησιμοποιούν αυτές τις πληροφορίες για να ανιχνεύσουν αν η επικοινωνία των κόμβων γίνεται μέσω κάποιας wormhole.

Στις επόμενες ενότητες θα περιγράψουμε περιληπτικά τους δύο πρώτους τρόπους αντιμετώπισης των wormholes και θα αναλύσουμε την τρίτη μέθοδο.

4.1 Αντιμετώπιση wormholes με χρήση των packet leashes

Με τον όρο packet leashes αναφερόμαστε σε οποιεσδήποτε επιπλέον πληροφορίες προσθέτονται σε κάθε μήνυμα το οποίο στέλνει ένας κόμβος, με σκοπό να χρησιμοποιηθούν στο να περιορίσουν την μεγίστη απόσταση στην οποία μπορεί να μεταδοθεί το μήνυμα. Υπάρχουν δύο κατηγορίες packet leashes, τα *geographical leashes* και τα *temporal leashes*.

Geographical leashes

Με την βοήθεια ενός geographical leash, όπως δηλώνει και το όνομά του, πετυχαίνουμε να είναι ο προορισμός ενός μηνύματος μέσα σε μια προκαθορισμένη απόσταση από τον αποστολέα, δηλαδή έτσι βάζουμε ένα άνω φράγμα στην μέγιστη απόσταση που μπορεί να διανύσει το μήνυμα.

Temporal leashes

Από την άλλη μεριά, με την βοήθεια ενός temporal leash, μπορούμε να βάλουμε ένα άνω φράγμα στο "χρόνο ζωής" ενός μηνύματος, δηλαδή ορίζουμε ένα χρονικό διάστημα ύστερα από το πέρας του οποίου το μήνυμα δεν θα είναι έγκυρο, οπότε και θα απορριφθεί από όποιο κόμβο το λάβει.

Στις επόμενες υποενότητες περιγράφουμε περιληπτικά πως κατασκευάζουμε το κάθε είδος packet leash.

4.1.1 Κατασκευή ενός Geographical Leash

Για να κατασκευάσει ένας κόμβος ένα geographical leash, πρέπει **να ξέρει την τοποθεσία του**, και όλοι οι κόμβοι πρέπει να έχουν **χαλαρά συγχρονισμένα ρολόγια (loosely synchronized clocks)**. Όταν λέμε ότι οι κόμβοι A και B έχουν loosely synchronized clocks, εννοούμε ότι η διαφορά της ώρας που δείχνουν τα ρολόγια τους είναι το πολύ $\pm\Delta$, όπου το Δ εξαρτάται από τα A και B. Έτσι, όταν ο κόμβος A θέλει να στείλει ένα μήνυμα στον B, θα στείλει μαζί με το μήνυμα την θέση του p_a και την ώρα t_a που έστειλε το μήνυμα. Μόλις ο B λάβει το μήνυμα θα ελέγξει αυτά τα δεδομένα με την θέση του p_b και τη ώρα t_b που έλαβε το μήνυμα.

Αν λοιπόν θεωρήσουμε ότι ο κάθε κόμβος κινείται με ταχύτητα το πολύ v, τότε ο κόμβος B μπορεί να υπολογίσει ένα άνω φράγμα της απόστασης d_{ab} μεταξύ του A και του ιδίου, βασισμένος στα παραπάνω δεδομένα, και έτσι, αν η πραγματική απόσταση των κόμβων είναι μεγαλύτερη από αυτό το άνω φράγμα, τότε πιθανόν το μήνυμα έχει φτάσει στον προορισμό του μέσω wormhole και θα αγνοηθεί. Συγκεκριμένα, αν δ είναι το μέγιστο δυνατό λάθος στην μέτρηση της θέσης ενός κόμβου, τότε θα ισχύει ότι το άνω φράγμα της απόστασης d_{ab} μεταξύ των κόμβων είναι $d_{ab} \leq \|p_a - p_b\| + 2v \cdot (t_b - t_a + \Delta) + \delta$, είναι δηλαδή το άθροισμα της αρχικής απόστασης $\|p_a - p_b\|$ των δύο κόμβων, του σφάλματος δ στη μέτρηση των θέσεων τους και της απόστασης $2v \cdot (t_b - t_a + \Delta)$ την οποία πιθανόν να κάλυψαν με την κίνησή τους μέχρι το μήνυμα να φτάσει από τον A στον B. Τέλος, αναφέρουμε ότι για την προστασία των δεδομένων t_a και p_a (δηλαδή για να μην αυτά αλλοιωθούν από κάποιο εχθρικό κόμβο στην πορεία), χρησιμοποιείται κάποια μέθοδος ψηφιακής υπογραφής τους, όπως για παράδειγμα η μέθοδος RSA.

4.1.2 Κατασκευή ενός Temporal Leash

Για να κατασκευάσει ένας κόμβος ένα temporal leash, πρέπει όλοι οι κόμβοι να έχουν **αυστηρά συγχρονισμένα ρολόγια (tightly synchronized clocks)**. Όταν λέμε ότι οι κόμβοι A και B έχουν tightly synchronized clocks, εννοούμε

ότι η διαφορά της ώρας που δείχνουν τα ρολόγια τους είναι το πολύ $\pm\Delta$, όπου το Δ είναι το ίδιο για όλα τα ζεύγη κόμβων (δηλαδή είναι ανεξάρτητο από τους A και B). Ακόμα, η τιμή Δ πρέπει να είναι γνωστή από όλους τους κόμβους του δικτύου (και γενικά για τα temporal leashes πρέπει να είναι κάποια microseconds ή nanoseconds). Έτσι λοιπόν, όταν ο κόμβος A θέλει να στείλει ένα μήνυμα στον B, στέλνει μαζί με το μήνυμα και την χρονική στιγμή t_a την οποία το μεταδίδει (το t_a ονομάζεται timestamp). Μόλις ο κόμβος B λάβει το μήνυμα, ελέγχει αυτή την τιμή με την χρονική τιμή t_b που έλαβε το μήνυμα.

Συνεπώς, επειδή τα μηνύματα ταξιδεύουν το πολύ με την ταχύτητα του φωτός, ο B μπορεί να εξακριβώσει αν το μήνυμα που έλαβε έχει ταξιδέψει πολύ μακριά (δηλαδή πέρα από την πραγματική εμβέλεια του κόμβου A). Ακόμα, ο αποστολέας A μπορεί να κατασκευάσει ένα temporal leash ως εξής: βασιζόμενος στο γεγονός ότι τα μηνύματα ταξιδεύουν το πολύ με την ταχύτητα του φωτός, ο A μπορεί να υπολογίσει πόσο χρόνο t'_a χρειάζεται ένα μήνυμα για να ταξιδέψει την μέγιστη απόσταση στην οποία θέλει να μεταδοθεί, και έτσι αντί για την χρονική στιγμή της μετάδοσης του μηνύματος θα στείλει αυτή την χρονική στιγμή t'_a (που ονομάζουμε expiration time), ύστερα από το πέρας της οποίας όποιος κόμβος λάβει το μήνυμα θα πρέπει να το απορρίψει. Τέλος, όπως και στην περίπτωση των geographic leashes, για την προστασία των δεδομένων t_a και t'_a (δηλαδή για να μην αυτά αλλοιωθούν από κάποιο εχθρικό κόμβο στην πορεία), χρησιμοποιείται κάποια μέθοδος ψηφιακής υπογραφής τους, όπως για παράδειγμα η μέθοδος RSA.

4.2 Αντιμετώπιση wormholes με χρήση του πρωτοκόλλου ODSBR

Το προτόκολλο ODSBR ανήκει στην κατηγορία των on–demand routing protocols, και αυτό μόλις ανιχνεύσει την ύπαρξη κάποιας Byzantine επίθεσης προσπαθεί να βρει μια άλλη διαδρομή προς τον προορισμό η οποία δεν διέρχεται από κανέναν από τους κόμβους οι οποίοι συμβάλλουν στο να εκτελεστεί αυτή η επίθεση.

Συγκεκριμένα, οι σχεδιαστές του πρωτοκόλλου ODSBR παρατηρούν ότι το κυρίως πρόβλημα δεν είναι η δημιουργία της wormhole (μιας και αυτό στην ουσία θα διευκόλυνε τις συνδέσεις μεταξύ των κόμβων του δικτύου), αλλά το γεγονός ότι ο εχθρός δεν προωθεί τα μηνύματα με τα δεδομένα μέσω της wormhole που ελέγχει. Άρα, **μπορούμε να δούμε την wormhole σαν μια προβληματική σύνδεση μεταξύ δύο γειτονικών κόμβων της διαδρομής που έχει ανακαλυφθεί, και να προτιμήσουμε μια άλλη διαδρομή.**

Έτσι, περιληπτικά μπορούμε να πούμε ότι το πρωτόκολλο ODSBR λειτουργεί σε τρεις διαδοχικές φάσεις:

1. Route Discovery με ανιχνευση προβληματικων συνδεσεων

   Στην πρώτη φάση της λειτουργίας του, το πρωτόκολλο προσπαθεί να ανακαλύψει μια διαδρομή προς το επιθυμητό προορισμό με βάση τα κόστη των συνδέσεων μεταξύ των κόμβων τα οποία γνωρίζει, ανεξάρτητα από το αν μερικές αυτές τις συνδέσεις διέρχονται μέσω wormhole.

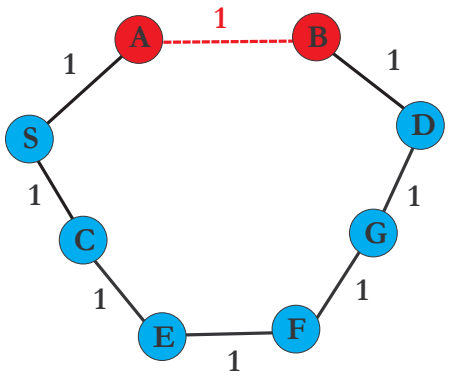
2. ΑΝΙΧΝΕΥΣΗ BYZANTINE ΕΠΙΘΕΣΕΩΝ

   Σε αυτή τη φάση το πρωτόκολλο προσπαθεί να βρει αν υπάρχουν προβληματικές συνδέσεις στην διαδρομή που ανακάλυψε, το οποίο μπορεί να σημαίνει ότι κάποιοι κόμβοι διενεργούν μια Byzantine wormhole επίθεση. Αυτό γίνεται χρηγσιμοποιώντας μια προσαρμοσμένη τεχνική όπως στο ROUTE MAINTENANCE του DSR η οποία δέχεται ως είσοδο τη διαδρομή και μας δίνει ως έξοδο την (πρώτη) προβληματική σύνδεση της διαδρομής.

3. ΔΙΑΧΕΙΡΙΣΗ ΤΟΥ ΚΟΣΤΟΥΣ ΤΩΝ ΓΝΩΣΤΩΝ ΣΥΝΔΕΣΕΩΝ

   Σε αυτή τη φάση το πρωτόκολλο διαχειρίζεται τη λίστα με τα κόστη των συνδέσεων που έχουν ανακαλυφθεί προηγουμένως ότι είναι προβληματικά από την φάση ΑΝΙΧΝΕΥΣΗ BYZANTINE ΕΠΙΘΕΣΕΩΝ. Έτσι, το πρωτόκολλο αυξάνει προοδευτικά το κόστος κάθε τέτοιας προβληματικής σύνδεσης, και η λίστα με αυτά τα κόστη χρησιμοποιείται πάλι από την πρώτη φάση για να αποφύγει την χρήση της ίδιας διαδρομής και να ανακαλύψει μια καλύτερη (δηλαδή πιο ασφαλή) διαδρομή.

Για να δείξουμε πως δουλεύει το πρωτόκολλο ενάντια στις wormhole επιθέσεις, θεωρούμε για παράδειγμα το ακόλουθο δίκτυο:



Σχήμα 4.1: Παράδειγμα χρήσης του πρωτοκόλλου ODSBR (οι κόμβοι A και B δημιουργούν μια byzantine wormhole $A \rightsquigarrow B$)

Στο δίκτυο αυτό ας υποθέσουμε ότι ο κόμβος S θέλει να στείλει ένα μήνυμα στον κόμβο D, και ότι κάθε κόστος σύνδεσης μεταξύ των κόμβων είναι αρχικά 1. Έτσι, λόγω της ύπαρξης της επίθεσης byzantine wormhole μεταξύ των κόμβων A και B (που είναι μέρος του αρχικού δικτύου αλλά τους ελέγχει ο εχθρός), η συντομότερη διαδρομή που θα ανακαλυφθεί είναι η $S \xrightarrow{1} A \overset{1}{\rightsquigarrow} B \xrightarrow{1} D$, με κόστος 3. Παρατηρούμε ότι η μοναδική "πραγματική" διαδρομή που υπάρχει στο δίκτυο είναι η $S \xrightarrow{1} C \xrightarrow{1} E \xrightarrow{1} G \xrightarrow{1} D$, με κόστος 5.

Μόλις ο κόμβος S χρησιμοποιήσει την διαδρομή μέσω της wormhole, το πρωτόκολλο θα δεί ότι στο μονοπάτι που έχει διαλέξει δεν παραδίδονται τα μηνύματα και θα μπει στη δεύτερη φάση όπου θα ψάξει να βρει την προβληματική σύνδεση. Μόλις βρει ότι αυτή είναι η σύνδεση $A \rightsquigarrow B$, η τρίτη φάση του πρωτοκόλλου θα διπλασιάσει το κόστος της σύνδεσης αυτής. Έπειτα το πρωτόκολλο

θα ξεκινήσει ένα δεύτερο ROUTE DISCOVERY, αυτή τη φορά όμως η διαδρομή $S \xrightarrow{1} A \overset{2}{\rightsquigarrow} B \xrightarrow{1} D$ θα έχει κόστος $1+2+1=4$. Επειδή $4<5$ το πρωτόκολλο θα διαλέξει πάλι αυτή την διαδρομή, και μόλις δει ότι τα μηνύματα δεν παραδίδονται θα βρει πάλι ότι η σύνδεση $A \rightsquigarrow B$ είναι προβληματική. Έτσι, θα διπλασιάσει πάλι το κόστος αυτής της σύνδεσης, το οποίο θα γίνει 4. Στο ROUTE DISCOVERY λοιπόν (που θα εκτελεστεί αμέσως μετά), η διαδρομή $S \xrightarrow{1} A \overset{4}{\rightsquigarrow} B \xrightarrow{1} D$ θα έχει κόστος $1+4+1=6$, ενώ η διαδρομή $S \xrightarrow{1} C \xrightarrow{1} E \xrightarrow{1} G \xrightarrow{1} D$ θα έχει κόστος 5, επομένως αυτή τη φορά θα προτιμηθεί η "πραγματική" διαδρομή του δικτύου. Με αυτόν τον τρόπο το πρωτόκολλο ODSBR αποφεύγει όλα τα wormhole attacks.

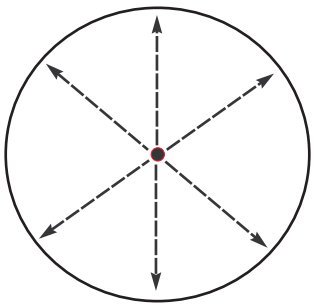
Στις επόμενες ενότητες θα παρουσιάσουμε και θα αναλύσουμε την μέθοδο αντιμετώπισης των wormhole επιθέσεων με directional antennas. Όπως θα δούμε παρακάτω, αυτή η μέθοδος αντιμετωπίζει αποτελεσματικά μόνο μια επίθεση wormhole που πιθανόν να υπάρχει στο δίκτυο, και όχι την byzantine overlay wormhole επίθεση.

4.3 Αντιμετώπιση wormholes με χρήση directional antennas

Στις παρακάτω ενότητες θα περιγράψουμε πως μπορούμε να αντιμετωπίσουμε επιθέσεις wormhole με την χρήση directional antennas. Αρχικά, στην ενότητα 4.4 θα περιγράψουμε το μοντέλο της κεραίας που θα χρησιμοποιήσουμε και έπειτα στην ενότητα 4.5 θα παρουσιάσουμε τρεις αλγορίθμους εύρεσης γειτόνων για κόμβους του δικτύου οι οποίοι μας δίνουν την δυνατότητα να εντοπίσουμε αν υπάρχει κάποιο wormhole.

4.4 Το μοντέλο κεραίας που θα χρησιμοποιήσουμε

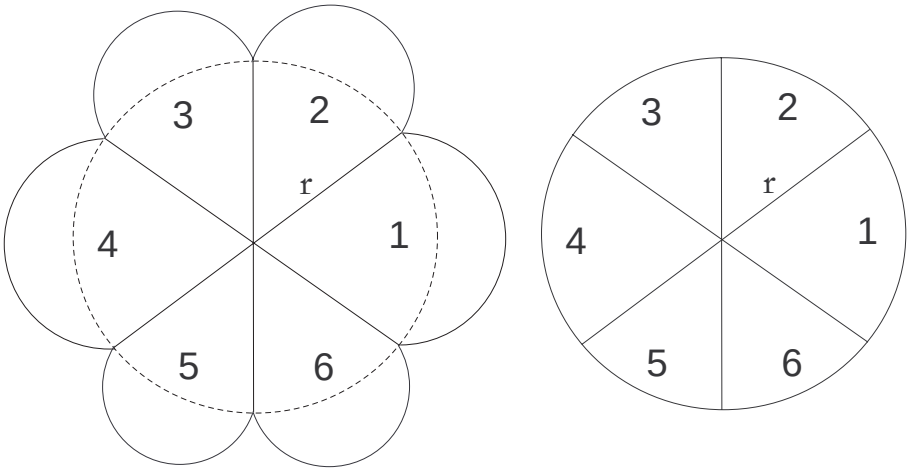
Συνήθως για την επικοινωνία μεταξύ των κόμβων στα ασύρματα δίκτυα χρησιμοποιούνται κεραίες που στέλνουν και δέχονται το σήμα σε όλες τις κατευθύνσεις (omnidirectional antennas), όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 4.2: Omnidirectional Antenna όπου το σήμα μεταδίδεται προς όλες τις κατευθύνσεις

Για την αντιμετώπιση των wormholes μπορούμε να χρησιμοποιήσουμε ένα άλλο είδος κεραιών, τις κατευθυνόμενες κεραίες (directional antennas). Αυτές χωρίζονται σε 2 μεγάλες κατηγορίες, τις **περιστρεφόμενες (steered)** και τις **μεταγώγιμες (switched)**. Οι steered antennas ουσιαστικά αποτελούνται από μια στοιχειώδη κεραία η οποία μπορεί να μεταδίδει προς μια κατεύθυνση και ένα μηχανισμό για να περιστρέφει την κεραία ώστε να καλύπτονται όλες οι κατευθύνσεις. Μια switched directional antenna αποτελείται από N στοιχειώδεις κεραίες. Κάθε μια από αυτές τις στοιχειώδεις κεραίες στέλνει το σήμα σε μια επιφάνεια που για απλοποίηση θεωρούμε ότι είναι το εμβαδόν ενός κυκλικού τομέα με επίκεντρη γωνία $\frac{2\pi}{N}$ rad, έτσι ώστε να καλύπτονται όλες οι κατευθύνεις του επιπέδου και οι επιμέρους επιφάνειες που στέλνουν το σήμα οι στοιχειώδεις κεραίες να μην έχουν κανένα κοινό σημείο, όπως φαίνεται στο σχήμα 4.3.

Στο αριστερό σχήμα βλέπουμε ότι η πραγματική επιφάνεια που περιγράφει την εμβέλεια της κεραίας έχει το σχήμα "μαργαρίτας", και με την διακεκομμένη γραμμή συμβολίζουμε την μεγίστη κυκλική επιφάνεια που είναι υποσύνολο της πρώτης. Επειδή στα πρωτόκολλα που θα παρουσιάσουμε αργότερα δεν μας ενδιαφέρει το βεληνεκές του μηνύματος που στέλνουμε προς κάποια κατεύθυνση να είναι μέγιστο, στην υπόλοιπη εργασία θα χρησιμοποιούμε το δεξί σχήμα για



Σχήμα 4.3: Κατευθυνόμενη κεραία με 6 στοιχειώδεις κεραίες, όπου η γωνία αποστολής προς κάθε κατεύθυνση είναι 60 μοίρες

να συμβολίζουμε μια switched directional antenna. Όπως λοιπόν βλέπουμε στο σχήμα 4.3, κάθε μια από τις στοιχειώδεις κεραίες έχει ένα αριθμό από 1 μέχρι 6, και αυτή η αρίθμηση είναι ακριβώς ίδια για όλες τις κατευθυνόμενες κεραίες που αποτελούνται από 6 στοιχειώδεις κεραίες. Αυτό το πετυχαίνουμε με τη βοήθεια μιας μαγνητικής βελόνας που περιέχει η κεραία, η οποία παραμένει συγγραμική με το μαγνητικό πεδίο της γης. Έτσι μπορούμε να εξασφαλίσουμε για παράδειγμα ότι η στοιχειώδης κεραία με νούμερο 1 δείχνει πάντα προς την ανατολή. Οι κύριες διαφορές των δύο μοντέλων των κεραιών είναι ότι οι steered antennas μας δίνουν καλύτερη ακρίβεια ως προς το σε ποιά κατεύθυνση θα στείλουμε το μήνυμα, ενώ οι switched antennas είναι πιο φτηνές στην υλοποίηση τους, ειδικά σε ad–hoc δίκτυα, και για αυτό στους αλγορίθμους που θα παρουσιάσουμε αργότερα χρησιμοποιούνται switched antennas.

Η κεραία έχει δύο μεθόδους λειτουργίας, τις Omni και Directional, και η εναλλαγή μεταξύ των δύο λειτουργιών γίνεται σε αμελητέο χρόνο. Η κεραία βρίσκεται σε λειτουργία Omni όταν δεν στέλνει κάποιο σήμα, και σε λειτουργία Directional στην αντίθετη περίπτωση. Όπως και στις omnidirectional antennas, οι directional antennas έχουν την δυνατότητα να λαμβάνουν σήμα από όλες τις κατευθύνσεις. Η διαφορά τους είναι ότι οι directional antennas μπορούν να στέλνουν σήμα μόνο προς μια κατεύθυνση κάθε φορά. Όταν η κεραία βρίσκεται στην λειτουργία Omni, μόλις γίνει η λήψη κάποιου σήματος, η κεραία αρχικά προσδιορίζει σε ποιά στοιχειώδη κεραία το εισερχόμενο σήμα είναι πιο δυνατό, και έπειτα χρησιμοποιεί αυτή για να λάβει την υπόλοιπη μετάδοση. Όταν η κεραία βρίσκεται στη λειτουργία Directional, τότε χρησιμοποιείται μόνο μια από τις στοιχειώδεις κεραίες για την αποστολή δεδομένων. Έτσι, αν στην λειτουργία Omni το σήμα λαμβάνεται με ισχύ G^o, στην λειτουργία Directional το σήμα αποστέλεται με ισχύ $G^d \geq G^o$. Ακόμα, επειδή στην αποστολή δεδομένων μπορεί να χρησιμοποιηθεί μόνο μια στοιχειώδης κεραία, όταν θέλουμε να μεταδώσουμε κάποιο μήνυμα προς όλες τις κατευθύνσεις, τότε στέλνουμε το μήνυμα από όλες τις

στοιχειώδεις κεραίες, χρησιμοποιώντας μια κάθε φορά. Έτσι, αν για παράδειγμα θέλουμε να στείλουμε ένα μήνυμα προς όλες τις κατευθύνσεις χρησιμοποιώντας την κατευθυνόμενη κεραία του σχήματος 4.3, θα στείλουμε το μήνυμα 6 φορές, όσες φορές δηλαδή είναι και οι στοιχειώδεις κεραίες από τις οποίες αποτελείται.

4.5 Αλγόριθμοι ανακάλυψης γειτόνων οι οποίοι παρέχουν ανίχνευση wormholes

Οι αλγόριθμοι για την ανίχνευση wormholes που θα παρουσιάσουμε στη συνέχεια, μας παρέχουν προστασία ενάντια σε μεμονομένες wormhole επιθέσεις, όταν δηλαδή ο εχθρός ελέγχει μόνο 2 κόμβους που συνδέονται με γρήγορη σύνδεση, και βασίζονται στο να γνωρίζουν όλοι οι κόμβοι το ακριβές σύνολο γειτόνων τους. Είναι φανερό ότι τότε ο εχθρός δεν θα μπορεί δημιουργήσει κανένα wormhole, αφού σε αυτή την περίπτωση οι αρχικοί κόμβοι του δικτύου θα μπορούν να ξεχωρίζουν τα άκρα της wormhole από τους υπόλοιπους "πραγματικούς" γείτονές τους, εξακριβώνοντας ποιοί είναι αυτοί. Κατά συνέπεια, οι κόμβοι του δικτύου θα μπορούν τότε απλά να αγνοούν όλα τα μηνύματα που προέρχονται από κόμβους οι οποίοι δεν ανήκουν στην λίστα γειτόνων τους. Για να γίνει αυτό όπως θα δούμε, οι κόμβοι θα χρησιμοποιήσουν το γεγονός ότι όταν επικοινωνούν με κατευθυνόμενη κεραία, τότε γνωρίζουν την κατεύθυνση από την οποία λαμβάνουν το μήνυμα.

Αρχικά θα κάνουμε μερικές υποθέσεις σχετικά με τις επικοινωνίες μεταξύ των κόμβων του δικτύου. Υποθέτουμε ότι όλες οι επικοινωνίες μεταξύ των κόμβων του δικτύου είναι αμφίδρομες και ότι σε κάθε ζεύγος κόμβων έχει μοιραστεί ένα κλειδί το οποίο μπορεί να χρησιμοποιηθεί με κάποιο αλγόριθμο συμμετρικής κρυπτογράφησης, έτσι ώστε όλα τα κρίσιμα μηνύματα που ανταλλάσσονται να είναι κρυπτογραφημένα. Το τελευταίο μπορεί να γίνει μιας και έχουν προταθεί αρκετοι αποδοτικοί μηχανισμοί για ανταλαγή κλειδιών σε ad–hoc δίκτυα.

Στις επόμενες ενότητες, με $zone(A, B)$ θα συμβολίζουμε την ζώνη της κατευθυνόμενης κεραίας του κόμβου A στην οποία γίνεται η λήψη ενός μηνύματος που του στέλνει ο κόμβος B. Ακόμα, με $\hat{}zone(A, B)$ θα συμβολίζουμε την αντίθετη ζώνη της $zone(A, B)$, όπου οι αντίθετες ζώνες μιας κατευθυνόμενης κεραίας ορίζονται ως εξής:

$zone$	$\hat{}zone$
1	4
2	5
3	6

Τέλος, με το συμβολισμό neighbors$(A, zone)$ θα εννοούμε τους κόμβους που βρίσκονται σε απόσταση ενός βήματος από τον κόμβο A, και βρίσκονται στη ζώνη $zone$ της κατευθυνόμενης κεραίας του A.

Οι κόμβοι χρησιμοποιούν οποιοδήποτε από τους αλγορίθμους ως εξής: Αμέσως μόλις οι κόμβοι τοποθετηθούν στις θέσεις τους, το σύνολο με τους γείτονές τους θα είναι κενό, και κάθε ένας θα διαλέξει τυχαία μια χρονική στιγμή όπου θα χρησιμοποιήσει τον αλγόριθμο για να δημιουργήσει το σύνολο με τους γείτονες του. Έπειτα, κάθε κόμβος θα χρησιμοποιεί περιοδικά τον ίδιο αλγόριθμο για να

ενημερώσει το σύνολο με τους γείτονες του, στην περίπτωση που αυτοί έχουν αλλάξει (π.χ. λόγω μετακινήσεων των κόμβων, διαφόρων εμποδίων που ίσως να μπλοκάρουν τις επικοινωνίες, κλπ.) Σε κάθε εφαρμογή του αλγορίθμου, θα ονομάζουμε **εκφωνητή (announcer)** τον κόμβο που έχει ξεκινήσει την διαδικασία εφαρμογής.

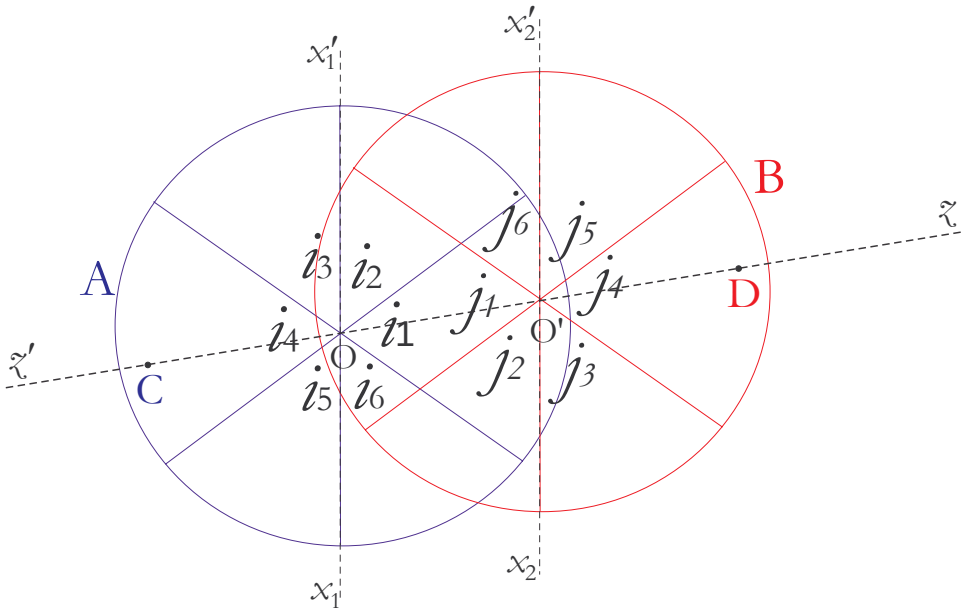
4.5.1 Ανακάλυψη Γειτόνων με βάση την Κατεύθυνση Λήψης (Directional Neighbor Discovery)

Αυτός ο αλγόριθμος αποτελεί την βάση και για τους υπόλοιπους δυο αλγορίθμους που θα παρουσιάσουμε, και για αυτό τον λόγο δεν αποτρέπει όλες τις πιθανές wormhole επιθέσεις. Αρχικά όμως θα δείξουμε το εξής:

Πρόταση 4.1. *Αν έχουμε δύο κόμβους A και B με κατευθυνόμενες κεραίες, τότε αν ο A βρίσκεται στη ζώνη $zone(B, A)$ της κεραίας του B, τότε και ο B βρίσκεται στη ζώνη $\hat{}zone(B, A)$ του A, δηλαδή $zone(A, B) = \hat{}zone(B, A)$ και έτσι*

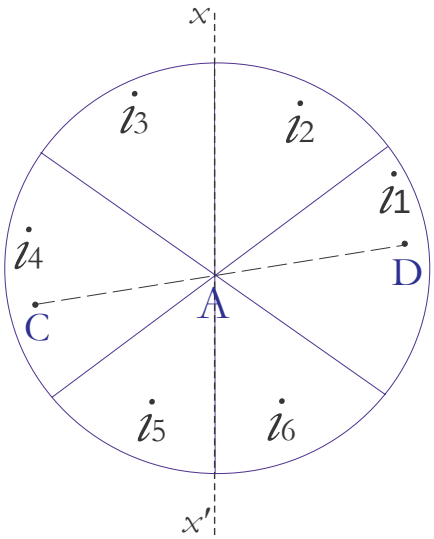
$$A \in neighbors(B, zone(B, A)) \Longleftrightarrow B \in neighbors(A, zone(A, B))$$

Απόδειξη. Έστω ότι ο κόμβος A βρίσκεται στην ζώνη j_1 του κόμβου B και ότι ο κόμβος B βρίσκεται στη ζώνη i_1 του κόμβου A, όπου τα $\{i_1, i_2, i_3, i_4, i_5, i_6\}$ και $\{j_1, j_2, j_3, j_4, j_5, j_6\}$ είναι κάθε ένα κάποιο από τα ακόλουθα συνόλα: $\{1, 2, 3, 4, 5, 6\}$, $\{2, 3, 4, 5, 6, 1\}$, $\{3, 4, 5, 6, 1, 2\}$, $\{4, 5, 6, 1, 2, 3\}$, $\{5, 6, 1, 2, 3, 4\}$, $\{6, 1, 2, 3, 4, 5\}$.



Έστω ότι O, O' είναι οι θέσεις των κόμβων A και B. Θεωρούμε τους κατακόρυφους άξονες x'_1x_1 και x'_2x_2 που διέρχονται από τα σημεία O και O' αντίστοιχα. Ακόμα, θεωρούμε την ευθεία zz' που ορίζεται από το ευθύγραμμο τμήμα OO', και τα σημεία C και D έτσι ώστε να ισχύει $CO = OO' = O'D$. Έπειτα κάνουμε την

εξής παρατήρηση: αν ένα σημείο K βρίσκεται στην ζώνη $zone$ της κεραίας ενός κόμβου N έτσι ώστε η γωνία $\widehat{xNK}$ να είναι θ, τότε το σημείο K' που ορίζεται από το ευθύγραμμο τμήμα $K'N = KN$ και την γωνία $\widehat{xNK'} = \pi + \theta$ βρίσκεται στη ζώνη $\hat{}zone$ της ίδιας κεραίας.



Σχήμα 4.4: Το σημείο K βρίσκεται στην ζώνη i_1, το σημείο K' στην ζώνη i_4 (που είναι η $\hat{}i_1$) και για τις γωνίες έχουμε ότι $\widehat{xAK'} = \pi + \widehat{xAK}$

Σκεφτόμαστε τώρα ως εξής: για τις γωνίες $\widehat{x_1OO'}$ και $\widehat{x_1OC}$ ισχύει ότι $\widehat{x_1OC} = \pi + \widehat{x_1OO'}$, και ακόμα παρατηρούμε ότι το σημείο C, το οποίο ορίζεται από το ευθύγραμμο τμήμα OC το οποίο σχηματίζει γωνία $\widehat{x_1OC} = \pi + \widehat{x_1OO'}$ με τον άξονα $x_1'x_1$, βρίσκεται στην αντίθετη ζώνη της ζώνης i_1. Ακόμα, επειδή οι οι ευθείες $x_1'x_1$ και $x_2'x_2$ είναι παράλληλες, ισχύει ότι $\widehat{x_1OO'} = \widehat{x_2O'D}$. Επειδή οι όμως κεραίες των κόμβων A και B έχουν τον ίδιο προσανατολισμό, τότε και για το σημείο που θα ορίζεται από ένα ευθύγραμμο τμήμα του οποίου το ένα άκρο είναι το O', μήκους ίσο με OC και που θα σχηματίζει γωνία $\pi + \widehat{x_1OO'}$ με τον άξονα $x_2'x_2$ θα ισχύουν τα ίδια με το σημείο C, δηλαδή αυτό θα βρίσκεται σε ζώνη της κεραίας του B που θα είναι αντίθετη με την ζώνη i_1 της κεραίας του A. Επειδή όμως $\widehat{x_2O'O} = \pi + \widehat{x_2O'D} = \pi + \widehat{x_1OO'}$ και $O'O = OC$, έχουμε ότι το σημείο αυτό είναι το O, δηλαδή το σημείο O βρίσκεται σε ζώνη της κεραίας του B που είναι αντίθετη με την ζώνη i_1 της κεραίας του A. Τελικά, αφού το σημείο O (και άρα ο κόμβος A) βρίσκεται στην ζώνη j_1 της κεραίας του B, έχουμε ότι $i_1 = \hat{}j_1$, δηλαδή $zone(A, B) = \hat{}zone(B, A)$. □

Αλγόριθμος Directional Neighbor Discovery

Ο αλγόριθμος λοιπόν αποτελείται από τα 3 ακόλουθα βήματα:

1. $A \rightarrow Region \quad$ HELLO $| \, ID_A$

   Σε αυτό το βήμα ο κόμβος A (Announcer) κάνει broadcast ένα HELLO μήνυμα το οποίο περιέχει την ταυτότητά του ID_A, το στέλνει δηλαδή σε όλες τις ζώνες της κεραίας του.

2. $N \rightarrow A \quad ID_N | E_{K_{NA}}(ID_A | R | zone(N, A))$

   Σε αυτό το βήμα όλοι οι κόμβοι που λαμβάνουν το HELLO μήνυμα του A στέλνουν ένα μήνυμα–απάντηση στην ζώνη της κεραίας τους όπου έλαβαν το HELLO μήνυμα του A. Το μήνυμα που στέλνουν περιέχει αρχικά την ταυτότητα ID_N του κόμβου N και το υπόλοιπο είναι κρυπτογραφημένο με το κοινό κλειδί K_{NA} που γνωρίζουν οι κόμβοι N και A. Το περιεχόμενο του κρυπτογραφημένου μηνύματος είναι η ταυτότητα ID_A του announcer, ένα τυχαίο αριθμό R που ονομάζουμε nonce και την ζώνη όπου ο κόμβος N έλαβε το μήνυμα του A.

3. $A \rightarrow N \quad R$

   Ο announcer αποκρυπτογραφεί το μήνυμα και ελέγχει ότι περιέχει την ταυτότητά του ID_A. Έπειτα ελέγχει ότι η ζώνη που το έλαβε είναι η αντίθετη από αυτή που του έστειλε το μήνυμα ο πιθανός γείτονας N, δηλαδή ότι ισχύει ότι $zone(A, N) = \hat{}zone(N, A)$. Αν ναι, τότε προσθέτει τον κόμβο N στο σύνολο των γειτόνων του για την ζώνη $zone(A, N)$ και στέλνει το αποκρυπτογραφημένο nonce R στον κόμβο N στη $zone(A, N)$, αλλιώς αγνοεί το μήνυμα που έλαβε. Μόλις ο N λάβει το R, προσθέτει και αυτός με την σειρά του τον A στο σύνολο neighbors$(N, zone(N, A))$.

Πρόταση 4.2. *Ο αλγόριθμος directional discovery δουλεύει σωστά, δηλαδή μετά την εκτέλεση και των 3 βημάτων του, οι κόμβοι A και N που αναγνωρίζονται ως γείτονες είναι όντως γειτονικοί κόμβοι.*

Απόδειξη. Εξετάζοντας το 3ο βήμα παρατηρούμε ότι για τους κόμβους A και N αναγνωρίζονται ως γείτονες αν και μόνο αν ισχύει ότι $zone(A, N) = \hat{}zone(N, A)$, το οποίο μας εξασφαλίζει ότι κάθε κόμβος βρίσκεται μέσα σε κάποια ζώνη της κεραίας του άλλου. Άρα ο ένας είναι στην εμβέλεια του άλλου, δηλαδή οι A και N είναι όντως γειτονικοί κόμβοι. □

Κάθε κόμβος που χρησιμοποιεί τον αλγόριθμο directional neighbor discovery τροποποιεί το σύνολο των γειτόνων του για την κάθε ζώνη της κεραίας του μόνο ύστερα από εφαρμογή του αλγορίθμου, είτε από απαντήσεις σε ανακοινώσεις που κάνει ο ίδιος ή από απαντήσεις σε ανακοινώσεις που του στέλνουν οι γείτονές του. Έτσι, όταν ένας κόμβος λάβει κάποιο μήνυμα το οποίο δεν σχετίζεται με τον αλγόριθμο αυτό, τότε ο κόμβος θα ελέγξει αν ο αποστολέας του μηνύματος βρίσκεται στο σύνολο γειτόνων για την ζώνη στην οποία έλαβε το μήνυμα και αν ναι, τότε δέχεται το μήνυμα, αλλιώς το απορρίπτει. Έτσι, αν τα σύνολα γειτόνων διατηρούνται και ενημερώνονται με ακρίβεια από τον κάθε κόμβο με τον τρόπο

που περιγράψαμε, τότε μια επίθεση wormhole δεν είναι δυνατή, αφού οι κόμβοι δεν θα δεχτούν μηνύματα από κόμβους οι οποίοι δεν είναι γείτονές τους (και άρα όχι σε κάποιο από τα σύνολα των γειτόνων τους). Αυτό όμως προϋποθέτει ότι εφαρμόζεται κάποια μέθοδος ελέγχου της αυθεντικότητας των μηνυμάτων, π.χ. με κρυπτογράφηση των μηνυμάτων μεταξύ των κόμβων χρησιμοποιώντας τα ζεύγη κλειδιών τα οποία τους έχουν μοιραστεί, γιατί στην αντίθετη περίπτωση ένας εχθρός θα μπορούσε να στείλει ο ίδιος κάποιο παρόμοιο μήνυμα ή να υποκλέψει και να αλλάξει κάποιο από τα μηνύματα που στέλνονται έτσι ώστε να πάρει τη θέση κάποιου από τους "νόμιμους" γείτονες.

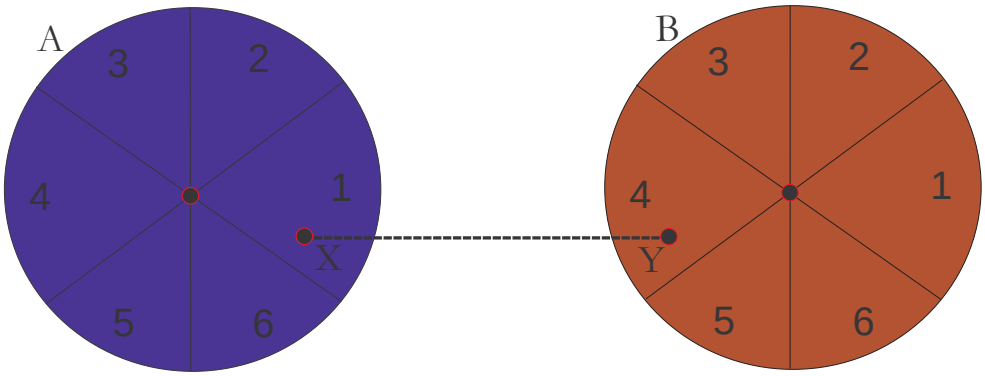
Πρόταση 4.3. *Ο αλγόριθμος directional neighbor dicovery δεν επιτρέπει να δημιουργηθούν "ψεύτικοι" γείτονες, δηλαδή κόμβοι που νομίζουν ότι είναι γείτονες αλλά ουσιαστικά επικοινωνούν μέσω wormhole, εκτός από την περίπτωση που τα άκρα της wormhole βρίσκονται σε αντίθετες ζώνες των κεραιών των 2 κόμβων.*

Απόδειξη. Έστω ότι μεταξύ των κόμβων A και B παρεμβάλεται μια wormhole με το ένα άκρο στην ζώνη $zone_A$ της κεραίας του κόμβου A και το άλλο άκρο στην ζώνη $zone_B$ της κεραίας του κόμβου B έτσι ώστε $zone_A \neq \hat{}zone_B$. Υποθέτουμε τώρα ότι έχει εκτελεστεί ο αλγόριθμος και ότι οι κόμβοι A και B έχουν προσθέσει ο ένας τον άλλο στα σύνολα γειτόνων τους για τις κατάλληλες ζώνες. Εξετάζοντας το 3ο βήμα του αλγορίθμου, βλέπουμε ότι ο A προσθέτει τον B στο σύνολο neighbors$(A, zone_A)$ αν και μόνο αν $zone_A = \hat{}zone_B$ και ο B προσθέτει τον A στο σύνολο neighbors$(B, \hat{}zone)$ αν και μόνο αν λάβει το αποκρυπτογραφημένο nonce R που του έχει στείλει. Αφού λοιπόν οι A και B έχουν προσθέσει ο ένας τον άλλο στα σύνολα γειτόνων τους για της κατάλληλες ζώνες, αυτό θα σήμαινε ότι $zone_A = \hat{}zone_B$ και ότι ο B έχει λάβει το nonce R, τα οποία είναι άτοπα και τα δύο, γιατί έχουμε ότι $zone_A \neq \hat{}zone_B$ από την υπόθεση, και ακόμα λόγω του προηγουμένου ο A δεν στέλνει το nonce R στον B. Επιπλέον, ο B δεν μπορεί να το έλαβε από κανένα άλλο επειδή το κλειδί K_{AB} που χρησιμοποιείται για να κρυπτογραφηθεί το μήνυμα στο 2ο βήμα του αλγορίθμου είναι μυστικό, και επομένως μόνο ο A θα μπορούσε να αποκρυπτογραφήσει το μήνυμα $ID_B|E_{K_{AB}}(ID_A|R|\hat{}zone)$ με το nonce που του έστειλε ο B. Τελικά, βλέπουμε ότι οι A και B δεν μπορεί να έχουν αποδεχτεί ο ένας τον άλλο ως γείτονες μέσω του directional neighbor discovery αλγορίθμου. □

Αδυναμίες του αλγορίθμου Directional Neighbor Discovery

Όπως μας προϊδεάζει και η πρόταση 4.3, ο αλγόριθμος directional neighbor discovery δεν αποτρέπει όλες τις πιθανές wormhole επιθέσεις. Συγκεκριμένα, αν ανάμεσα σε 2 κόμβους παρεμβάλεται μια wormhole έτσι ώστε τα άκρα της να βρίσκονται σε αντίθετες ζώνες των κεραιών των A και B, τότε και τα 3 βήματα του αλγορίθμου εκτελούνται χωρίς πρόβλημα παρά το γεγονός ότι οι A και B δεν είναι πραγματικά γείτονες.

Στο σχήμα 4.5 δείχνουμε την αδυναμία του directional neighbor discovery αλγορίθμου να εντοπίσει την wormhole X–Y από το να πείσει τους κόμβους A και B ότι είναι γείτονες. Αυτή την επίθεση την ονομάζουμε directional attack και παρακάτω περιγράφουμε τι θα ακριβώς θα συμβεί σε αυτή την περίπτωση σε κάθε βήμα του αλγορίθμου.



Σχήμα 4.5: Directional Attack

1ο Βήμα: Αρχικά ο κόμβος A θα στείλει ένα HELLO μήνυμα μαζί με την ταυτότητά του ID_A στη ζώνη 1. Κατόπιν, το ένα άκρο X της wormhole θα συλλάβει το μήνυμα και θα το προωθήσει στο άλλο άκρο Y, το οποίο θα το κάνει broadcast (δηλαδή θα το στείλει σε όλες τις κατευθύνσεις. Ο κόμβος B θα ακούσει το μήνυμα του A (HELLO$|ID_A$) στην ζώνη 4 της κεραίας του, και σε αυτό το σημείο παρατηρούμε ότι οι ζώνες 1 και 4 είναι αντίθετες.

2ο Βήμα: Ο κόμβος B θα στείλει το μήνυμα $ID_B|E_{K_{AB}}(ID_A|R|\,4)$ στην ζώνη 4 της κεραίας του. Έπειτα το άκρο Y της wormhole θα συλλάβει το μήνυμα και θα το προωθήσει στο άλλο άκρο X, το οποίο θα το κάνει πάλι broadcast. Ο κόμβος A λοιπόν θα ακούσει το μήνυμα του B στην ζώνη 1 της κεραίας του, όπως φαίνεται και από το σχήμα 4.5.

3ο Βήμα: Ο κόμβος A αποκρυπτογραφεί το κομμάτι $E_{K_{AB}}(ID_A|R|\,4)$ του μηνύματος και βλέπει τα στοιχεία $ID_A|R|\,4$. Διαπιστώνει αμέσως μετά ότι το μήνυμα περιέχει την ταυτότητά του ID_A και ότι $zone(B,A) = 4$. Έπειτα κάνει τον έλεγχο $zone(A,B) = \hat{}zone(B,A)$ και βλέπει ότι ισχύει η ισότητα γιατί $zone(A,B) = 1$ και $\hat{}zone(B,A) = 1$, οπότε προσθέτει τον B στο σύνολο neighbors$(A,1)$ και στέλνει το nonce R στον B. Έτσι, ο B αφού έλαβε το σωστό nonce προσθέτει τον A στο σύνολο neighbors$(B,4)$.

Έτσι λοιπόν, βλέπουμε ότι με τη βοήθεια του αλγορίθμου αυτού μπορούμε να αποτρέψουμε ένα μεγάλο ποσοστό wormhole επιθέσεων μεταξύ 2 κόμβων. Συγκεκριμένα, επειδή μόνο οι wormhole επιθέσεις όπου τα άκρα της wormhole είναι σε αντίθετες ζώνες είναι επιτυχείς, και επειδή υπάρχουν 6 ζεύγη αντιθέτων ζωνών και 36 ζεύγη ζωνών συνολικά, μπορούμε να πούμε ότι κατά μέσο όρο με αυτόν τον αλγόριθμο μόνο το $\frac{1}{6}$ ή το $16,6\%$ των wormhole επιθέσεων ενάντια 2 κόμβων είναι επιτυχημένο. Αυτό όμως δεν σημαίνει μείωση της αποδοτικότητας της επίθεσης κατά ένα παρόμοιο ποσοστό, γιατί όπως έχουμε δει πιο στην περιγραφή της επίθεσης, οι κόμβοι που βρίσκουν συντομότερες διαδρομές μέσω της wormhole θα "διαφημίσουν" αυτά τα σύντομα δρομολόγια που ανακάλυψαν και στους γείτονές τους, με αποτέλεσμα να αυξήσουν την κίνηση των μηνυμάτων μέσω της wormhole χωρίς να το θέλουν. Έτσι, ένας έξυπνος εχθρός εξακολουθεί να μπορεί να προκαλέσει μεγάλο πρόβλημα στις επικοινωνίες μεταξύ των κόμβων αν βάλει

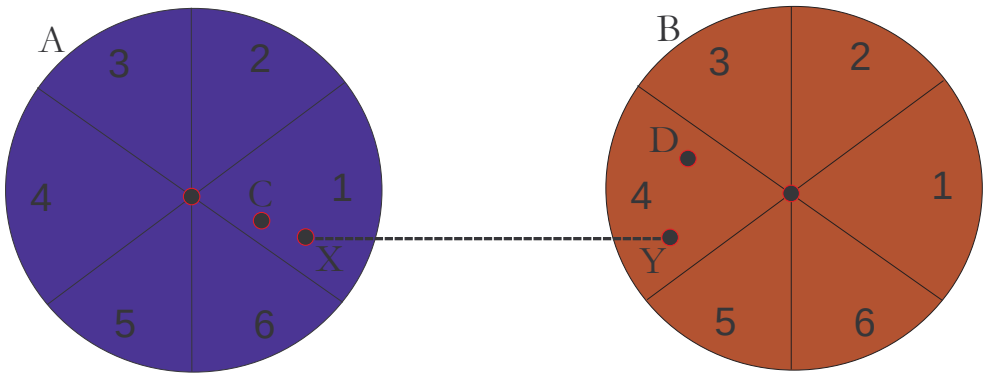
στρατηγικά τα άκρα της wormhole, όπως είδαμε για παράδειγμα στο σχήμα 3.7. Με άλλα λόγια, ακόμα και ένας μικρός αριθμός wormhole (μερικές φορές αρκεί και μια!) επηρεάζει σημαντικά τις επικοινωνίες μεταξύ των κόμβων.

4.5.2 Ανακάλυψη Γειτόνων με Επαλήθευση (Verified Neighbor Discovery)

Όπως είδαμε στην προηγούμενη ενότητα, η directional attack μπορεί να πείσει τους κόμβους A, B ότι είναι γείτονες μόνο αν τα άκρα της έχουν τοποθετηθεί με ένα κατάλληλο τρόπο (ο οποίος είναι να βρίσκονται σε ζώνες των κεραιών των κόμβων που είναι αντίθετες). Αυτό όμως σημαίνει ότι αν οι κόμβοι συνεργαστούν με κάποιον τρίτο κόμβο ο οποίος μπορεί να επικοινωνήσει και με τους δυο, τότε αυτός θα μπορούσε να εξακριβώσει αν οι κόμβοι είναι αρκετά κοντά ώστε να είναι όντως γείτονες. Αν θεωρήσουμε ότι οι A και B είναι γείτονες και ο B ακούει τον κόμβο A στη ζώνη *zone*, τότε αν χρησιμοποιήσουμε την πρόταση 4.1 της σελίδας 39, μπορούμε να δούμε ότι όλοι οι γείτονες του B στη ζώνη *zone* θα είναι γείτονες του A και αντίστροφα, δηλαδή ότι:

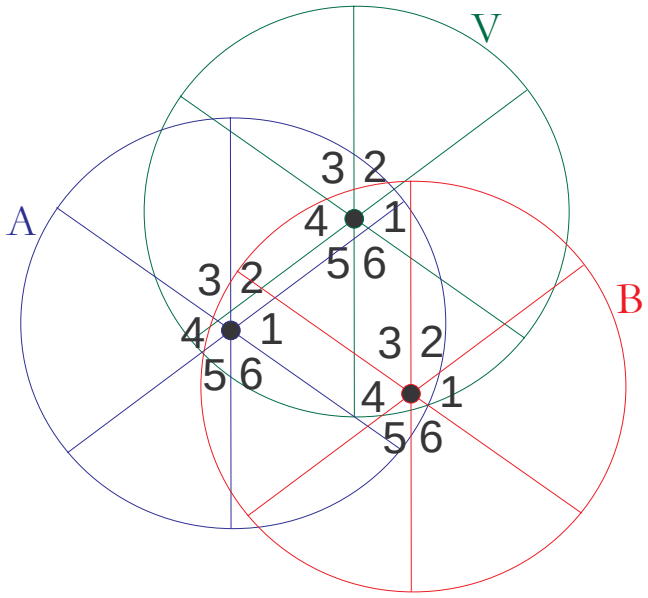
$$i \in \text{neighbors}(B, zone(B, A)) \Longleftrightarrow i \in \text{neighbors}(A, \hat{}zone(B, A))$$

Αυτοί οι κόμβοι, παρά το γεγονός ότι επικοινωνούν και με τους δυο κόμβους δεν μπορούν όλοι να χρησιμοποιηθούν για τη συνεργασία που περιγράψαμε παραπάνω, όπως βλέπουμε και στο ακόλουθο σχήμα:



Σχήμα 4.6: Οι κόμβοι C και D δεν μπορούν να χρησιμεύσουν για τον εντοπισμό της wormhole X–Y

Όπως βλέπουμε και στο προηγούμενο σχήμα οι κόμβοι C και D δεν μπορούν να χρησιμοποιηθούν για να εντοπίσουν αν παρεμβάλεται η wormhole X–Y στην επικοινωνία των κόμβων A και B, αφού κάθε επικοινωνία του κόμβου C προς τον B ή του D προς τον A θα πρέπει να περάσει αναγκαστικά μέσω της wormhole X–Y, μιας και οι C,B και οι D,A είναι εκτός εμβέλειας. Αυτό που θέλουμε από τον τρίτο κόμβο που θα συνεργαστεί με τους A, B είναι να επικοινωνεί και με τους δυο χωρίς οι επικοινωνίες αυτές να περνούν μέσω wormhole, όπως στο παρακάτω σχήμα:



Σχήμα 4.7: Ο κόμβος V μπορεί να χρησιμοποιηθεί για να εντοπίσει αν παρεμβάλεται wormhole μεταξύ A και B

Σε αυτό το σχήμα, παρατηρούμε ότι για τον κόμβο V ισχύουν τα παρακάτω:

$$zone(A, B) = 1 \neq 2 = zone(A, V) \tag{4.1}$$
$$zone(A, B) = 1 \neq 6 = zone(V, B) \tag{4.2}$$

Τέτοιοι κόμβοι, όπως ο κόμβος V του σχήματος, που ικανοποιούν τις δύο συνθήκες $zone(A, B) \neq zone(A, V)$ και $zone(A, B) \neq zone(V, B)$, τους ονομάζουμε **κόμβους επαλήθευσης (verifiers)**. Συγκεκριμένα, ένας κόμβος V είναι verifier για τους κόμβους A και B όταν:

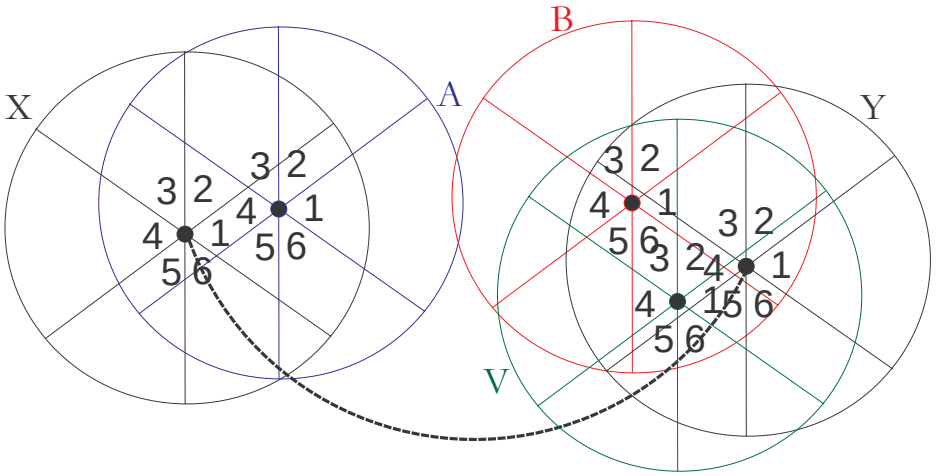
1. $zone(A, B) \neq zone(A, V)$. Αυτό σημαίνει ότι ο κόμβος A δέχεται τα μηνύματα των κόμβων B και V σε διαφορετικές ζώνες της κεραίας του, και επομένως δεν μπορεί να επικοινωνεί και με τους δύο μέσω της ίδιας wormhole.

2. $zone(A, B) \neq zone(V, B)$. Επειδή ισχύει ότι $zone(A, B) = \hat{}zone(B, A)$ και ότι $zone(V, B) = \hat{}zone(B, V)$ (από την πρόταση 4.1 σελίδα 39), έχουμε ότι ο κόμβος B δέχεται τα μηνύματα των κόμβων A και V σε διαφορετικές ζώνες της κεραίας του, και επομένως δεν μπορεί να επικοινωνεί και με τους δύο μέσω της ίδιας wormhole.

Βλέπουμε δηλαδή ότι ο verifier κόμβος V ουσιαστικά "δένει" τους κόμβους A και B, επαληθεύει δηλαδή ότι είναι αρκετά κοντά ώστε να είναι ο ένας στην εμβέλεια του άλλου. Παρακάτω θα δώσουμε μερικά σχήματα–παραδείγματα για να φανεί τι μας εξασφαλίζει η κάθε μια από τις δύο συνθήκες.

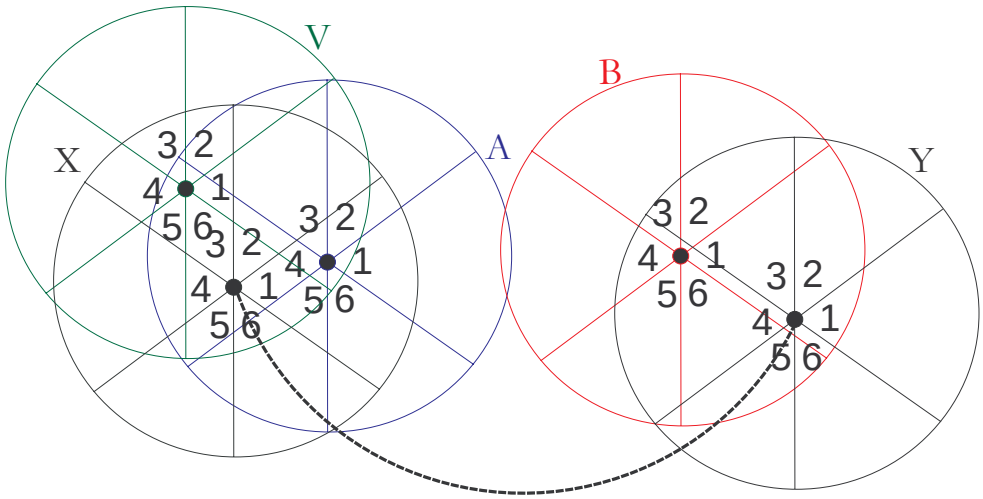
Στο επόμενο σχήμα 4.8 βλέπουμε ότι ο εχθρός προσπαθεί να πετύχει μια directional attack χρησιμοποιώντας την wormhole X–Y και έτσι έχουμε $zone(A, B) =$

4 και $zone(B, A) = 1$. Ακόμα, έχουμε ότι $zone(A, V) = 4$, δηλαδή βλέπουμε ότι $zone(A, B) = zone(A, V)$.



Σχήμα 4.8: Περίπτωση $zone(A, B) = zone(A, V)$

Όπως φαίνεται και από το σχήμα, ο B είναι μέσα στην εμβέλεια του κόμβου V ενώ ο κόμβος A δεν είναι. Άρα, αν ο A ακούει τον B και τον πιθανό verifier κόμβο V από την ίδια κατεύθυνση, δεν μπορούμε να χρησιμοποιήσουμε τον πιθανό verifier κόμβο για να ελέγξουμε αν οι επικοινωνίες των A και B είναι μέσω wormhole, αφού υπάρχει η περίπτωση να χρειαστεί να περάσουν τα μηνύματά του V από την wormhole X–Y για να επικοινωνήσει με τον A.

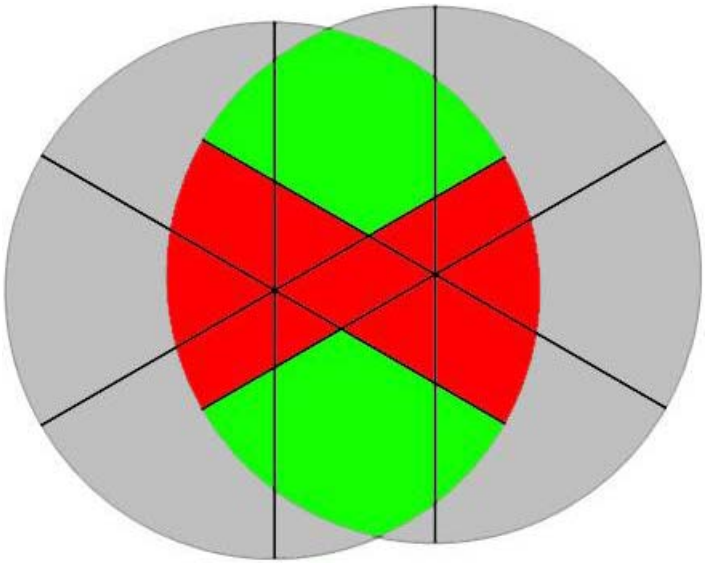


Σχήμα 4.9: Περίπτωση $zone(A, B) = zone(V, B)$

Στο σχήμα 4.9 βλέπουμε ότι ο εχθρός πάλι προσπαθεί να πετύχει μια directional attack χρησιμοποιώντας την wormhole X–Y με $zone(A, B) = 4$ και $zone(B, A) = 1$. Όπως φαίνεται και από το σχήμα, ο A είναι μέσα στην εμβέ-

λεια του κόμβου V ενώ ο κόμβος B όχι. Άρα, αν ο B ακούει τον A και τον πιθανό verifier κόμβο V από την ίδια κατεύθυνση, δεν μπορούμε να χρησιμοποιήσουμε τον πιθανό verifier κόμβο για να ελέγξουμε αν οι επικοινωνίες των A και B είναι μέσω wormhole, αφού υπάρχει η περίπτωση να χρειαστεί να περάσουν τα μηνύματά του V από την wormhole X–Y για να επικοινωνήσει με τον B.

Έτσι λοιπόν οι επιτρεπτές περιοχές όπου μπορεί να υπάρχει ένας verifier κόμβος για τους κόμβους A και B σύμφωνα με τον VERIFIED NEIGHBOR DISCOVERY αλγόριθμο φαίνονται στο παρακάτω σχήμα:



Σχήμα 4.10: Επιτρεπτές περιοχές για Verifiers (VERIFIED NEIGHBOR DISCOVERY)

Στο σχήμα 4.10 με γκρι χρώμα απεικονίζουμε τις περιοχές που αν υπάρχει πιθανός verifier κόμβος V δεν θα μπορεί να επικοινωνήσει και με τους δύο κόμβους A και B. Με κόκκινο χρώμα απεικονίζουμε τις περιοχές που αν υπάρχει πιθανός verifier κόμβος V, τότε θα ισχύει είτε $zone(A, B) = zone(A, V)$ είτε $zone(A, B) = zone(V, B)$ και τέλος με πράσινο χρώμα απεικονίζουμε τις περιοχές που αν ανήκει ένας κόμβος V, τότε είναι verifier κόμβος σύμφωνα με τις συνθήκες του VERIFIED NEIGHBOR DISCOVERY αλγορίθμου.

Στην επόμενη σελίδα θα περιγράψουμε τον επόμενο αλγόριθμο, ο οποίος αυτή τη φορά χρησιμοποιεί και verifier κόμβους για να διαπιστώσει αν οι κόμβοι επικοινωνούν μέσω κάποιας wormhole.

Αλγόριθμος Verified Neighbor Discovery

1. $A \rightarrow Region$ HELLO $|\, ID_A$

   Όπως και στο directional neighbor discovery, ο κόμβος A (Announcer) κάνει broadcast ένα HELLO μήνυμα το οποίο περιέχει την ταυτότητά του ID_A.

2. $N \rightarrow A$ $ID_N|E_{K_{NA}}(ID_A|R|zone(N,A))$

   Όπως και στο directional neighbor discovery, όλοι οι κόμβοι που λαμβάνουν το HELLO μήνυμα του A στέλνουν ένα μήνυμα–απάντηση στην ζώνη της κεραίας τους όπου έλαβαν το HELLO μήνυμα του A. Το μήνυμα που στέλνουν περιέχει αρχικά την ταυτότητα ID_N του κόμβου N και το υπόλοιπο είναι κρυπτογραφημένο με το κοινό κλειδί K_{NA} που γνωρίζουν οι κόμβοι N και A. Το περιεχόμενο του κρυπτογραφημένου μηνύματος είναι η ταυτότητα ID_A του announcer, ένα τυχαίο αριθμό R που ονομάζουμε nonce και την ζώνη όπου ο κόμβος N έλαβε το μήνυμα του A.

3. $A \rightarrow N$ R

   Όπως και στο directional neighbor discovery, ο announcer αποκρυπτογραφεί το μήνυμα και ελέγχει ότι περιέχει την ταυτότητά του ID_A. Έπειτα ελέγχει ότι η ζώνη που το έλαβε είναι η αντίθετη από αυτή που του έστειλε το μήνυμα ο πιθανός γείτονας N, δηλαδή ότι $zone(A,N) = \hat{}zone(N,A)$. Αν ναι, τότε στέλνει το αποκρυπτογραφημένο nonce R στον κόμβο N στη $zone(A,N)$, αλλιώς αγνοεί το μήνυμα που έλαβε.

   Μετά το τέλος αυτού του βήματος, οι κόμβοι A και N γνωρίζουν ο ένας σε ποιά κατεύθυνση βρίσκεται ο άλλος, αλλά δεν έχουν εξακριβώσει ότι επικοινωνούν χωρίς οι επικοινωνίες τους να προωθούνται μέσω κάποιας wormhole.

4. $N \rightarrow Region - \{zone(N,A), \hat{}zone(N,A)\}$ INQUIRY$|ID_N|ID_A|zone(N,A)$

   Σε αυτό το βήμα, όλοι οι πιθανοί γείτονες του announcer του βήματος 3 κάνουν broadcast σε όλες τις ζώνες εκτός από τις $zone(N,A)$, $\hat{}zone(N,A)$ ένα μήνυμα αναζήτησης (INQUIRY), για να βρεθεί κάποιος verifier. Έτσι, αν για παράδειγμα ο κόμβος N έλαβε το HELLO μήνυμα του A στη ζώνη 1, τότε θα στείλει το INQUIRY στις ζώνες 2, 3, 5 και 6. Όπως βλέπουμε, το μήνυμα περιέχει τη ζώνη $zone(N,A)$, έτσι ώστε οι πιθανοί verifier κόμβοι που θα λάβουν το μήνυμα να μπορούν να ελέγξουν αν ικανοποιούν τις δύο συνθήκες που περιγράψαμε πιο πάνω, οπότε θα είναι όντως verifiers για τους A και N.

5. $V \rightarrow N$ $ID_V|E_{K_{VN}}(ID_A|zone(V,N))$

   Εδώ, όσοι κόμβοι λάβανε το μήνυμα INQUIRY αρχικά εκτελούν τον αλγόριθμο directed neighbor discovery ώστε να δουν αν μπορούν να επικοινωνήσουν με τον κόμβο ο οποίος έχει το ID_A που έχουν λάβει. Έπειτα, ελέγχουν αν ικανοποιούν τις δύο συνθήκες που περιγράψαμε πριν, και αν ναι τότε είναι verifiers για τους κόμβους A και N, οπότε απαντούν στον κόμβο N με ένα κρυπτογραφημένο μήνυμα που περιέχει την ταυτότητά τους, την

ταυτότητα του κόμβου A που αρχικά έστειλε το HELLO μήνυμα και την ζώνη $zone(V, N)$ όπου λάβανε το μήνυμα του N.

Σε αυτό το σημείο, για να εκτελεστεί το 6ο βήμα του αλγορίθμου πρέπει ο κόμβος N να λάβει **τουλάχιστον ένα** μήνυμα από κάποιον verifier κόμβο. Αν αυτό συμβεί, τότε:

6. $N \rightarrow A \quad ID_N|E_{K_{AN}}(ID_A|\text{ACCEPT})$

   Ο κόμβος N αποδέχεται τον A ως γείτονα του, δηλαδή προσθέτει τον A στο σύνολο neighbors$(N, zone(N, A))$ και στέλνει στον A ένα κρυπτογραφημένο μήνυμα που περιέχει τις ταυτότητες τους και τη λέξη ACCEPT, οπότε και ο A δέχεται τον N ως γείτονά του στη ζώνη $zone(A, N)$.

Αν στο 4ο βήμα δεν υπάρχει κανένας κόμβος ο οποίος να ικανοποιεί τις συνθήκες 4.1 και 4.2 και έτσι δεν σταλεί στον N κανένα μήνυμα όπως έχει περιγραφεί στο βήμα 5, τότε ο αλγόριθμος σταματάει και οι κόμβοι A και N δεν θεωρούν ότι είναι γειτονικοί.

Πρόταση 4.4. *Αν οι κόμβοι A και B απέχουν απόσταση μεγαλύτερη ή ίση από $2r$ μεταξύ τους (όπου r είναι η ακτίνα του κύκλου που περιγράφει την εμβέλεια της κεραίας τους) τότε ο αλγόριθμος verified neighbor discovery δεν επιτρέπει να δημιουργηθούν "ψεύτικοι" γείτονες.*

Απόδειξη. Έστω ότι οι κόμβοι A και B απέχουν μεταξύ τους απόσταση $d \geq 2r$ και ότι μεταξύ των κόμβων A και B υπάρχει κάποια wormhole η οποία προωθεί τα μηνύματα του A στον B και αντίστροφα. Υποθέτουμε τώρα ότι έχει εκτελεστεί ο αλγόριθμος και ότι οι κόμβοι A και B έχουν προσθέσει ο ένας τον άλλο στα σύνολα γειτόνων τους για τις κατάλληλες ζώνες.

Εξετάζοντας το 3ο βήμα του αλγορίθμου διαπιστώνουμε ότι τα άκρα της wormhole πρέπει να βρίσκονται σε ζώνες των κεραιών των A και B έτσι ώστε οι ζώνες αυτές να είναι αντίθετες. Εξετάζοντας τώρα τα βήματα 4, 5 και 6, διαπιστώνουμε ότι θα υπάρχει κάποιος authenticated κόμβος V[1] ο οποίος θα λειτουργεί ως verifier και θα έχει στείλει ένα μήνυμα στον B, όπως περιγράφεται στο βήμα 5. Αυτό όμως θα σήμαινε ότι ο κόμβος V θα μπορεί να επικοινωνήσει και με τον A και με τον B. Η εμβέλεια του V περιγράφεται και αυτή από ένα κύκλο ακτίνας r, οπότε η μεγίστη απόσταση που μπορούν να έχουν δύο κόμβοι οι οποίοι βρίσκονται μέσα στην εμβέλειά του είναι όσο η διάμετρος του κύκλου που περιγράφει την εμβέλειά του, δηλαδή $2r$. Διακρίνουμε τις εξής δύο περιπτώσεις:
Για την απόσταση d μεταξύ τους ισχύει ότι $d > 2r$
Τότε είναι προφανές ότι ο κόμβος V δεν θα μπορεί να επικοινωνήσει και με τους δύο κόμβους, γιατί αν μπορούσε, τότε θα είχαμε ότι $d_{AV}, d_{BV} \leq r$ και από την τριγωνική ανισότητα για το τρίγωνο με άκρα τους κόμβους A,B και V έχουμε ότι

$$d_{AB} \leq d_{AV} + d_{BV} \leq r + r \;\Rightarrow\; d \leq 2r$$

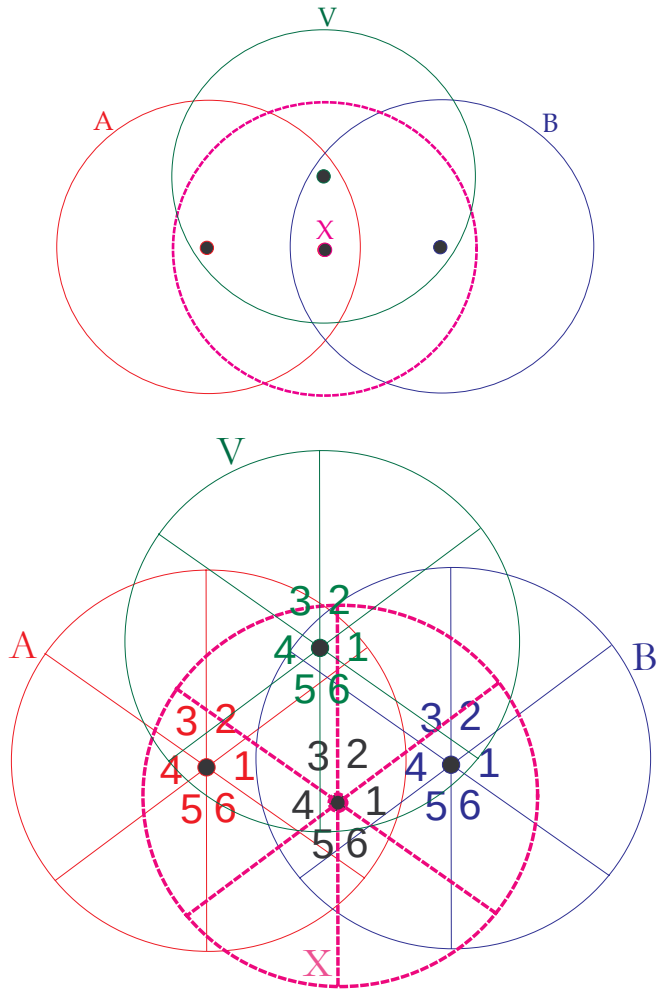
το οποίο είναι άτοπο από την υπόθεση.

[1] γιατί το μήνυμα που στέλνει ο verifier είναι κρυπτογραφημένο με το μυστικό κλειδί K_{BV}, άρα ο κόμβος V πρέπει να είναι authenticated node, δηλαδή κόμβος του αρχικού δικτύου

Για την απόσταση d **μεταξύ τους ισχύει ότι** $d = 2r$ Σε αυτή την περίπτωση παρατηρούμε ότι οι κόμβοι A, V και B βρίσκονται στην ίδια ευθεία. Τότε όμως, είναι προφανές ότι ο κόμβος A θα δέχεται τα μηνύματα των B και V στην ίδια ζώνη, δηλαδή $zone(A, B) = zone(A, V)$. Άρα η συνθήκη 4.1 δεν θα ισχύει και επομένως ο κόμβος V δεν είναι verifier, που είναι άτοπο.
Τελικά, βλέπουμε ότι οι A και B δεν μπορεί να έχουν αποδεχτεί ο ένας τον άλλο ως γείτονες μέσω του verified neighbor discovery αλγορίθμου. □

Αδυναμίες του Verified Neighbor Discovery αλγορίθμου

Όπως είδαμε στην προηγούμενη πρόταση, ο αλγόριθμος verified neighbor discovery αποτρέπει τις wormhole επιθέσεις όταν οι κόμβοι απέχουν τουλάχιστον απόσταση $2r$. Τι γίνεται όμως στην περίπτωση που για την απόσταση d μεταξύ των κόμβων ισχύει ότι $r < d < 2r$;



Σχήμα 4.11: Worawannotai attack

Στο παραπάνω σχήμα βλέπουμε την τετριμμένη μορφή της wormhole επίθεσης

(που παρουσιάσαμε αρχικά στο σχήμα 3.2 της ενότητας 3.2), η οποία ονομάζεται Worawannotai attack. Σε αυτή την επίθεση, όπως έχουμε πει προηγουμένως, τα δύο άκρα της wormhole ταυτίζονται, δηλαδή ο εχθρός ελέγχει μόνο ένα κόμβο X και ουσιαστικά αναμεταδίδει όλα τα μηνύματα που λαμβάνει από μια από τις ζώνες της κεραίας του σε όλες τις υπόλοιπες ζώνες της.

Στο σχήμα 4.11 λοιπόν βλέπουμε ότι ενώ οι κόμβοι A και B είναι (ελάχιστα) εκτός εμβέλειας, επειδή ο εχθρικός κόμβος X προωθεί τα μηνύματα του ενός στον άλλο, οι A και B μπορούν τελικά να επικοινωνήσουν μεταξύ τους. Στη συνέχεια περιγράφουμε τι θα ακριβώς θα συμβεί σε αυτή την περίπτωση σε κάθε βήμα του αλγορίθμου VERIFIED NEIGHBOR DISCOVERY.

1ο Βήμα: Αρχικά ο κόμβος A θα στείλει ένα HELLO μήνυμα μαζί με την ταυτότητά του ID_A στη ζώνη 1. Κατόπιν, ο εχθρικός κόμβος X (της τετριμμένης wormhole X–X) θα συλλάβει το μήνυμα στη ζώνη 4 της κεραίας του και θα το αναμεταδώσει σε όλες τις άλλες κατευθύνσεις, δηλαδή στις ζώνες 2,3,4,5, και 6, οπότε ο κόμβος B θα ακούσει το μήνυμα του A (HELLO$|ID_A$) στην ζώνη 4 της κεραίας του.

2ο Βήμα: Ο κόμβος B θα στείλει το μήνυμα $ID_B|E_{K_{AB}}(ID_A|R_1|\,4)$ στην ζώνη 4 της κεραίας του. Έπειτα ο κόμβος A θα ακούσει το μήνυμα του B στην ζώνη 1 της κεραίας του μέσω του κόμβου X (όπως περιγράψαμε στο 1ο βήμα), όπως φαίνεται και στο σχήμα 4.11.

3ο Βήμα: Ο κόμβος A αποκρυπτογραφεί το κομμάτι $E_{K_{AB}}(ID_A|R|\,4)$ του μηνύματος και βλέπει τα στοιχεία $ID_A|R|\,4$. Διαπιστώνει αμέσως μετά ότι το μήνυμα περιέχει την ταυτότητά του ID_A και ότι $zone(B,A) = 4$. Έπειτα κάνει τον έλεγχο $zone(A,B) = \hat{}zone(B,A)$ και βλέπει ότι ισχύει η ισότητα γιατί $zone(A,B) = 1$ και $\hat{}zone(B,A) = 1$, οπότε στέλνει το αποκρυπτογραφημένο nonce R_1 στον B.

4ο Βήμα: Ο κόμβος B τώρα θα αναζητήσει κάποιο verifier κόμβο και έτσι θα στείλει το μήνυμα INQUIRY$|ID_B|ID_A|4$ στις ζώνες 2,3,5 και 6 της κεραίας του.

5ο Βήμα: Ο κόμβος V θα λάβει το μήνυμα INQUIRY$|ID_B|ID_A|4$ του B στη ζώνη 6 της κεραίας του. Με βάση τη ζώνη που έλαβε το μήνυμα και το περιεχόμενο του μηνύματος, ο V σε αυτή τη φάση γνωρίζει ότι $zone(V,B) = 6$ και $zone(B,A) = 4$, και ισοδύναμα $zone(A,B) = \hat{}zone(B,A) = 4$. Στη συνέχεια εκτελεί τον αλγόριθμο DIRECTED NEIGHBOR DISCOVERY για να δει αν μπορεί να επικοινωνήσει με τον κόμβο A.

DIRECTED NEIGHBOR DISCOVERY ΓΙΑ ΚΟΜΒΟΥΣ V ΚΑΙ A

1. Ο V στέλνει το μήνυμα HELLO$|ID_V$ σε όλες τις ζώνες της κεραίας του.
2. Ο A ακούει το μήνυμα HELLO$|ID_V$ του V και στέλνει στη ζώνη 2 της κεραίας του (όπου δέχτηκε το μήνυμα του V) το ακόλουθο μήνυμα $ID_A|E_{K_{AV}}(ID_A|R_2|\,2)$

3. Ο V θα δεχτεί το μήνυμα $ID_A|E_{K_{AV}}(ID_V|R_2|\,2)$ του Α στη ζώνη 5 της κεραίας του. Αφού το αποκρυπτογραφήσει θα δει ότι περιέχει την ταυτότητά του ID_V και ότι $zone(V,A) = 5 = \hat{}2 = \hat{}zone(A,V)$, επομένως μπορεί να επικοινωνήσει με τον Α και θα του στείλει το αποκρυπτογραφημένο nonce R_2.

Αφού λοιπόν ο V διαπίστωσε ότι μπορεί να επικοινωνήσει με τον κόμβο Α στη ζώνη $zone(V,A) = 5$ και άρα $zone(A,V) = 2$, ο V ελέγχει ότι

$$zone(A,B) = 4 \neq 2 = zone(A,V)$$

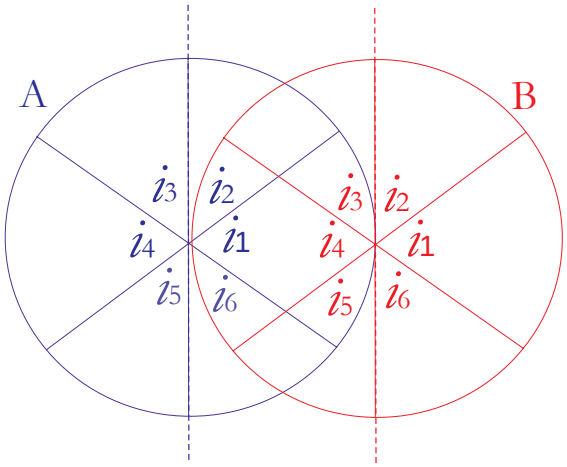
και ότι

$$zone(A,B) = 4 \neq 6 = zone(V,B)$$

Άρα ο V είναι verifier για τους Α και Β οπότε στέλνει στη ζώνη 6 της κεραίας του το μήνυμα $ID_V|E_{K_{VB}}(ID_A|6)$ στον Β.

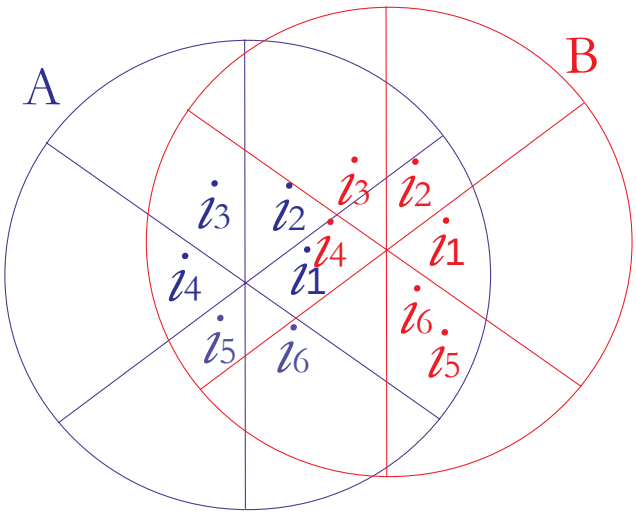
6ο Βήμα: Ο κόμβος Β αφού λάβει το μήνυμα του V αποδέχεται τον Α ως γείτονά του ($A \in neigbors(B,4)$) και στέλνει στη ζώνη 4 της κεραίας του το μήνυμα $ID_B|E_{K_{AB}}(ID_A|\text{ACCEPT})$. Αυτό το μήνυμα το λαμβάνει ο κόμβος Χ στη ζώνη 1 και το αναμεταδίδει στις ζώνες 2, 3, 4, 5 και 6, επομένως το λαμβάνει και ο κόμβος Α στη ζώνη 1 της κεραίας του, και αφού το αποκρυπτογραφήσει αποδέχεται τον V ως γείτονά του ($B \in neigbors(A,1)$).

Βλέπουμε λοιπόν ότι ο αλγόριθμος εκτελείται κανονικά, μιας και οι κόμβοι βρίσκουν ένα verifier, και οι Α και Β αποδέχονται ο ένας τον άλλο ως γείτονα, ενώ στην ουσία είναι εκτός εμβέλειας. Παρατηρούμε ότι ισχύει $zone(A,V) = 2$, $zone(A,B) = 1$ και $zone(V,B) = 6$, δηλαδή ότι η ζώνη $zone(A,V)$ είναι γειτονική και με την $zone(A,B)$ και με την $zone(V,B)$. Αυτό δεν είναι τυχαίο, γιατί η περιοχή που μπορεί να βρίσκεται ένας τέτοιος "ψεύτικος" verifier είναι μεταξύ των δύο κατακορύφων αξόνων που διέρχονται από τις θέσεις των δύο κόμβων, όπως φαίνεται στο παρακάτω σχήμα:



Σχήμα 4.12: Περιοχή που μπορεί να βρίσκονται "ψεύτικοι" verifiers

Συγκεκριμένα, αν $\{i_1, i_2, i_3, i_4, i_5, i_6\}$ είναι μια τυχαία διάταξη της κεραίας, τότε για την περιοχή που είναι η τομή των επιφανειών των ζωνών i_2 του A και i_3 του B, έχουμε ότι αν βρίσκεται κάποιος verifier V εκεί θα ισχύει $zone(A, V) = i_2$ και $zone(B, V) = i_3$. Ακόμα έχουμε ότι $zone(A, B) = i_1$ και ακόμα $zone(V, B) = \hat{}zone(B, V) = \hat{}i_3 = i_6$ και επομένως η ζώνη $zone(A, V)$ είναι γειτονική και με την $zone(A, B)$ και με την $zone(V, B)$. Όμοια, για την περιοχή που είναι η τομή των επιφανειών των ζωνών i_6 του A και i_5 του B, έχουμε ότι αν βρίσκεται κάποιος verifier V εκεί θα ισχύει $zone(A, V) = i_6$, $zone(V, B) = \hat{}i_5 = i_2$ και $zone(A, B) = i_1$, δηλαδή πάλι η ζώνη $zone(A, V)$ είναι γειτονική και με την $zone(A, B)$ και με την $zone(V, B)$.



Σχήμα 4.13: Όλες οι περιοχές των verifiers

Ας θεωρήσουμε στο παραπάνω σχήμα 4.13 την περιοχή για την οποία ισχύει ότι αν υπάρχει κάποιος verifier V εκεί θα ισχύει ότι $zone(A, V) = i_3 = zone(B, V)$. Είναι προφανές ότι σε αυτή την περιοχή δεν μπορεί να υπάρξει κάποιος verifier ο οποίος να χρησιμεύσει σε Worawannotai attack και επιπλέον $zone(A, B) = i_1$ και $zone(V, B) = \hat{}i_3 = i_6$, δηλαδή η ζώνη $zone(A, V)$ δεν είναι γειτονική και με την $zone(A, B)$ και με την $zone(V, B)$. Αυτό ισχύει και για τις άλλες τρεις τέτοιες περιοχές, όπου δηλαδή $zone(A, V) = zone(B, V) = i_2$, $zone(A, V) = zone(B, V) = i_5$ και $zone(A, V) = zone(B, V) = i_6$. Άρα, αν θέλουμε να ανακαλύψουμε verifiers οι οποίοι να μην μπορούν να χρησιμοποιηθούν από τον εχθρό σε κάποια Worawannotai attack, θα πρέπει η ζώνη $zone(A, V)$ να μην είναι γειτονική ταυτόχρονα και με την $zone(A, B)$ και με την $zone(V, B)$.

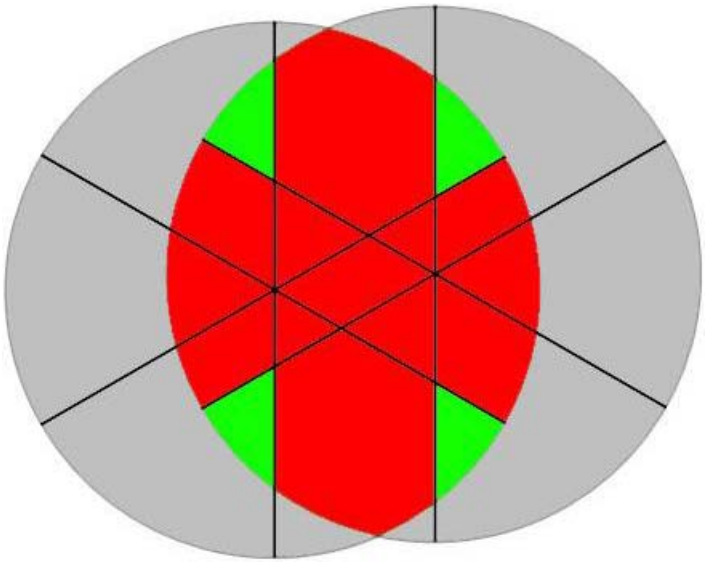
4.5.3 Αυστηρή Ανακάλυψη Γειτόνων (Strict Neighbor Discovery)

Λαμβάνοντας υπ'όψη τις παραπάνω παρατηρήσεις, προσθέτουμε μια νέα συνθήκη για το πότε θα θεωρούμε ότι ένας κόμβος V είναι verifier για τους κόμβους A και B (για την αποφυγή "ψεύτικων" verifiers σε ορισμένες περιπτώσεις όπως του σχήματος 4.11). Παρακάτω αναφέρουμε και τις τρεις συνθήκες που πρέπει να

ικανοποιούνται:

1. $zone(A, B) \neq zone(A, V)$.
2. $zone(A, B) \neq zone(V, B)$.
3. Η $zone(A, V)$ δεν μπορεί να είναι γειτονική και με την $zone(A, B)$ και με την $zone(V, B)$.

Στο επόμενο σχήμα 4.14 με γκρι χρώμα απεικονίζουμε τις περιοχές που αν υπάρχει πιθανός verifier κόμβος V δεν θα μπορεί να επικοινωνήσει και με τους δύο κόμβους A και B.



Σχήμα 4.14: Επιτρεπτές περιοχές για Verifiers (STRICT NEIGHBOR DISCOVERY)

Με κόκκινο χρώμα απεικονίζουμε τις περιοχές που αν υπάρχει πιθανός verifier κόμβος V, τότε θα ισχύει είτε $zone(A, B) = zone(A, V)$, είτε $zone(A, B) = zone(V, B)$ ή αλλιώς η $zone(A, V)$ θα είναι γειτονική και με την $zone(A, B)$ και με την $zone(V, B)$. Τέλος, με πράσινο χρώμα απεικονίζουμε τις περιοχές που αν ανήκει ένας κόμβος V, τότε είναι verifier κόμβος σύμφωνα με τον STRICT NEIGHBOR DISCOVERY αλγόριθμο.

Στην επόμενη σελίδα θα περιγράψουμε τον τελευταίο αλγόριθμο για την ανίχνευση wormholes, ο οποίος χρησιμοποιεί και αυτός verifier κόμβους για να διαπιστώσει αν οι κόμβοι επικοινωνούν μέσω κάποιας wormhole, οι οποίοι όμως ικανοποιούν επιπλέον και την τρίτη συνθήκη που περιγράψαμε πριν.

Αλγόριθμος Strict Neighbor Discovery

1. $A \rightarrow Region$ HELLO $|\, ID_A$

   Όπως και στον αλγόριθμο verified neighbor discovery, ο κόμβος A (Announcer) κάνει broadcast ένα HELLO μήνυμα το οποίο περιέχει την ταυτότητά του ID_A.

2. $N \rightarrow A$ $ID_N|E_{K_{NA}}(ID_A|R|zone(N,A))$

   Όπως και στον αλγόριθμο verified neighbor discovery, όλοι οι κόμβοι που λαμβάνουν το HELLO μήνυμα του A στέλνουν ένα μήνυμα–απάντηση στην ζώνη της κεραίας τους όπου έλαβαν το HELLO μήνυμα του A. Το μήνυμα που στέλνουν περιέχει αρχικά την ταυτότητα ID_N του κόμβου N και το υπόλοιπο είναι κρυπτογραφημένο με το κοινό κλειδί K_{NA} που γνωρίζουν οι κόμβοι N και A. Το περιεχόμενο του κρυπτογραφημένου μηνύματος είναι η ταυτότητα ID_A του announcer, ένα τυχαίο αριθμό R που ονομάζουμε nonce και την ζώνη όπου ο κόμβος N έλαβε το μήνυμα του A.

3. $A \rightarrow N$ R

   Όπως και στον αλγόριθμο verified neighbor discovery, ο announcer αποκρυπτογραφεί το μήνυμα και ελέγχει ότι περιέχει την ταυτότητά του ID_A. Έπειτα ελέγχει ότι η ζώνη που το έλαβε είναι η αντίθετη από αυτή που του έστειλε το μήνυμα ο πιθανός γείτονας N, δηλαδή ελέγχει ότι ισχύει ότι $zone(A,N) = \hat{}zone(N,A)$. Αν ναι, τότε στέλνει το αποκρυπτογραφημένο nonce R στον κόμβο N στη $zone(A,N)$, αλλιώς αγνοεί το μήνυμα που έλαβε.

   Μετά το τέλος αυτού του βήματος, οι κόμβοι A και N γνωρίζουν ο ένας σε ποια κατεύθυνση βρίσκεται ο άλλος, αλλά δεν έχουν εξακριβώσει ότι επικοινωνούν χωρίς οι επικοινωνίες τους να προωθούνται μέσω κάποιας wormhole.

4. $N \rightarrow Region - \{zone(N,A), \hat{}zone(N,A)\}$ INQUIRY$|ID_N|ID_A|zone(N,A)$

   Εδώ, όλοι οι πιθανοί γείτονες του announcer του βήματος 3 κάνουν broadcast σε όλες τις ζώνες εκτός από τις $zone(N,A)$, $\hat{}zone(N,A)$ ένα μήνυμα αναζήτησης (INQUIRY), για να βρεθεί κάποιος verifier. Έτσι, αν για παράδειγμα ο κόμβος N έλαβε το HELLO μήνυμα του A στη ζώνη 1, τότε θα στείλει το INQUIRY στις ζώνες 2, 3, 5 και 6. Όπως βλέπουμε, το μήνυμα περιέχει τη ζώνη $zone(N,A)$, έτσι ώστε οι πιθανοί verifier κόμβοι που θα λάβουν το μήνυμα να μπορούν να ελέγξουν αν ικανοποιούν τις δύο συνθήκες που περιγράψαμε πιο πάνω, οπότε θα είναι όντως verifiers για τους A και N.

5. $V \rightarrow N$ $ID_V|E_{K_{VN}}(ID_A|zone(V,N))$

   Εδώ, όσοι κόμβοι λάβανε το μήνυμα INQUIRY αρχικά εκτελούν τον directed neighbor discovery αλγόριθμο ώστε να δουν αν μπορούν να επικοινωνήσουν με τον κόμβο ο οποίος έχει το ID_A που έχουν λάβει. Έπειτα, ελέγχουν αν ικανοποιούν τις τρεις συνθήκες που περιγράψαμε στην αρχή της ενότητας, και αν ναι τότε είναι verifiers για τους κόμβους A και N, οπότε απαντούν

στον κόμβο N με ένα κρυπτογραφημένο μήνυμα που περιέχει την ταυτότητά τους, την ταυτότητα του κόμβου A που αρχικά έστειλε το HELLO μήνυμα και την ζώνη $zone(V, N)$ όπου λάβανε το μήνυμα του N.

Σε αυτό το σημείο, για να εκτελεστεί το 6ο βήμα του αλγορίθμου πρέπει ο κόμβος N να λάβει **τουλάχιστον ένα** μήνυμα από κάποιον verifier κόμβο. Αν αυτό συμβεί, τότε:

6. $N \rightarrow A \quad ID_N | E_{K_{AN}}(ID_A | \text{ACCEPT})$

   Ο κόμβος N αποδέχεται τον A ως γείτονα του, δηλαδή προσθέτει τον A στο σύνολο $\text{neighbors}(N, zone(N, A))$ και στέλνει στον A ένα κρυπτογραφημένο μήνυμα που περιέχει τις ταυτότητες τους και τη λέξη ACCEPT, οπότε και ο A δέχεται τον N ως γείτονά του στη ζώνη $zone(A, N)$.

Θα αποδείξουμε στη συνέχεια μια πρόταση η οποία θα μας χρησιμεύσει αργότερα στο να δείξουμε ότι ο αλγόριθμος strict neighbor discovery δεν επιτρέπει να δημιουργηθούν "ψεύτικοι" γείτονες, και άρα μας προστατεύει από wormholes.

Πρόταση 4.5. *Όταν οι κόμβοι A και B απέχουν απόσταση μεγαλύτερη από r, τότε η περιοχή των πιθανών verifier κόμβων όταν χρησιμοποιούμε τον strict neighbor discovery αλγόριθμο είναι μηδέν.*

Απόδειξη. Έστω ότι οι κόμβοι A και B απέχουν απόσταση πάνω από r και θεωρούμε τον άξονα που διχοτομεί την γωνία του κυκλικού τομέα της ζώνης $zone(A, B)$ της κεραίας του A. Με x_K θα συμβολίζουμε την x συντεταγμένη του κόμβου K, με d_{KL} την απόσταση των κόμβων K και L, ενώ με θ_{KL} θα συμβολίζουμε την γωνία που σχηματίζεται από τον οριζόντιο άξονα που διέρχεται από το σημείο K και το ευθύγραμμο τμήμα KL. Παρατηρούμε ότι για τις περιοχές στις οποίες μπορούν να υπάρχουν verifiers V σύμφωνα με τον strict neighbor discovery αλγόριθμο ισχύει ότι $zone(A, V) = zone(B, V)$. Ακόμα, με βάση το συμβολισμό μας, θα ισχύουν ότι:

$$x_V = x_A + d_{AV} \cos\theta_{AV} \tag{4.3}$$

$$x_V = x_B + d_{BV} \cos\theta_{BV} \tag{4.4}$$

$$x_B = x_A + d_{AB} \cos\theta_{AB} \tag{4.5}$$

Ας υποθέσουμε τώρα ότι $zone(A, B) = 1$ και $zone(B, A) = 4$ και ότι υπάρχει ένας verifier όπως ορίζει ο αλγόριθμος strict neighbor discovery. Στο σχήμα 4.15 της σελίδας 57 βλέπουμε την περίπτωση όπου $zone(A, V) = zone(B, V) = 2$.

Διακρίνουμε τις εξής περιπτώσεις:

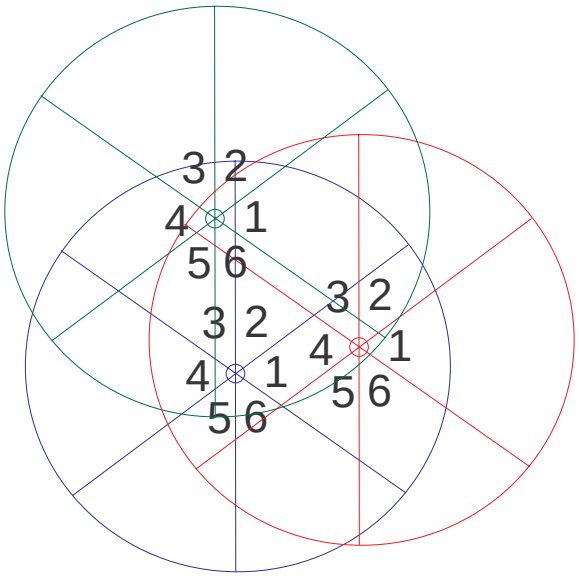
- $zone(A, V) = zone(B, V) = 3$ ή $zone(A, V) = zone(B, V) = 5$.

  Σε αυτή την περίπτωση, αν $zone(A, V) = zone(B, V) = 3$ θα έχουμε

  $$\frac{\pi}{2} \leq \theta_{AV} \leq \frac{2\pi}{3} \quad \text{και} \quad \frac{\pi}{2} \leq \theta_{BV} \leq \frac{2\pi}{3}$$

  ενώ αν ισχύει ότι $zone(A, V) = zone(B, V) = 5$ θα έχουμε

  $$\frac{7\pi}{6} \leq \theta_{AV} \leq \frac{3\pi}{2} \quad \text{και} \quad \frac{7\pi}{6} \leq \theta_{BV} \leq \frac{3\pi}{2}$$



Σχήμα 4.15: Strict verifier με $zone(A,V) = zone(B,V) = 2$

Άρα, για τις τιμές $\cos\theta_{AV}$ και $\cos\theta_{BV}$ θα ισχύει ότι

$$\cos\theta_{AV} \in \left[-\frac{1}{2}, 0\right] \quad \text{και} \quad \cos\theta_{BV} \in \left[-\frac{1}{2}, 0\right]$$

Θεωρώντας τώρα το maximum του $\cos\theta_{AV}$ στη σχέση 4.3 έχουμε ότι

$$x_V \leq x_A + d_{AV} \max\cos\theta_{AV} = x_A \;\Rightarrow\; x_V \leq x_A$$

Αντικαθιστώντας την σχέση 4.5 στην σχέση 4.4 παίρνουμε την σχέση

$$x_V = x_A + d_{AB}\cos\theta_{AB} + d_{AV}\cos\theta_{BV} \tag{4.6}$$

Επειδή $zone(A,B) = 1$ έχουμε ότι

$$-\frac{\pi}{6} \leq \theta_{AB} \leq \frac{\pi}{6} \;\Rightarrow\; \cos\theta_{AB} \in \left[\frac{\sqrt{3}}{2}, 1\right]$$

Θεωρώντας τώρα το minimum των συνημιτόνων στη σχέση 4.6 έχουμε ότι

$$x_V \geq x_A + d_{AB}\min\cos\theta_{AB} + d_{BV}\min\cos\theta_{BV} = x_A + \frac{d_{AB}\sqrt{3} - d_{BV}}{2}$$

Από την υπόθεση μας έχουμε ότι $d_{AB} > r$ και ακόμα $d_{BV} \leq r$ (αλλιώς οι κόμβοι B και V δεν θα μπορούσαν να επικοινωνήσουν), οπότε και ισχύει ότι $d_{AB}\sqrt{3} - d_{BV} > 0$, επομένως έχουμε $x_V > x_A$ από την προηγούμενη σχέση, το οποίο όμως είναι άτοπο επειδή είδαμε πριν ότι $x_V \leq x_A$. Συνεπώς, οι περιοχές των πιθανών verifiers που βρίσκονται αριστερά από τον κατακόρυφο άξονα που διέρχεται από τη θέση του κόμβου A είναι κενές.

- $zone(A,V) = zone(B,V) = 2$ ή $zone(A,V) = zone(B,V) = 6$.

  Σε αυτή την περίπτωση, αν $zone(A,V) = zone(B,V) = 2$ θα έχουμε

  $$\frac{\pi}{6} \leq \theta_{AV} \leq \frac{\pi}{2} \quad \text{και} \quad \frac{\pi}{6} \leq \theta_{BV} \leq \frac{\pi}{2}$$

  ενώ αν ισχύει ότι $zone(A,V) = zone(B,V) = 6$ θα έχουμε

  $$\frac{3\pi}{2} \leq \theta_{AV} \leq -\frac{\pi}{6} \quad \text{και} \quad \frac{3\pi}{2} \leq \theta_{BV} \leq -\frac{\pi}{6}$$

  Άρα, για τις τιμές $\cos\theta_{AV}$ και $\cos\theta_{BV}$ θα ισχύει ότι

  $$\cos\theta_{AV} \in \left[0, \frac{\sqrt{3}}{2}\right] \quad \text{και} \quad \cos\theta_{BV} \in \left[0, \frac{\sqrt{3}}{2}\right]$$

  Θεωρώντας το minimum του $\cos\theta_{BV}$ στη σχέση 4.4 έχουμε ότι

  $$x_V \geq x_B + d_{BV} \min\cos\theta_{BV} = x_B + 0 \cdot d_{BV} \;\Rightarrow\; x_V \geq x_B$$

  Αφαιρώντας κατά μέλη τις σχέσεις 4.3 και 4.5 έχουμε ότι

  $$x_V - x_B = d_{AV}\cos\theta_{AV} - d_{AB}\cos\theta_{AB} \;\Leftrightarrow$$

  $$x_B = x_V + d_{AB}\cos\theta_{AB} - d_{AV}\cos\theta_{AV} \tag{4.7}$$

  Επειδή $zone(A,B) = 1$ έχουμε ότι

  $$-\frac{\pi}{6} \leq \theta_{AB} \leq \frac{\pi}{6} \;\Rightarrow\; \cos\theta_{AB} \in \left[\frac{\sqrt{3}}{2}, 1\right]$$

  Θεωρώντας τώρα το minimum των συνημιτόνων στη σχέση 4.7 έχουμε ότι

  $$x_B \geq x_V + d_{AB}\min\cos\theta_{AB} - d_{AV}\min\cos\theta_{AV} = x_V + \frac{d_{AB}\sqrt{3}}{2}$$

Άρα, έχουμε $x_B > x_V$ από την προηγούμενη σχέση, το οποίο όμως είναι άτοπο επειδή είδαμε πριν ότι $x_V \leq x_B$. Συνεπώς, οι περιοχές των πιθανών verifiers που βρίσκονται δεξιά από τον κατακόρυφο άξονα που διέρχεται από τη θέση του κόμβου B είναι κενές.

Δείξαμε δηλαδή ότι αν $zone(A,B) = 1$, οι περιοχές των πιθανών verifiers σύμφωνα με τον αλγόριθμο strict neighbor dicovery είναι κενές. Η περίπτωση $zone(A,B) = 4$ είναι ακριβώς όμοια καθώς αντιστοιχεί σε εναλλαγή των θέσεων κόμβων A και B. Επιπρόσθετα, με την ίδια λογική, οι περιπτώσεις $zone(A,B) = 5$ και $zone(A,B) = 6$ είναι όμοιες με τις περιπτώσεις $zone(A,B) = 2$ και $zone(A,B) = 3$ αντίστοιχα.

Τέλος, μπορούμε να παρατηρήσουμε ότι οι αποδείξεις για τις περιπτώσεις $zone(A,B) = 2$ και $zone(A,B) = 3$ είναι όμοιες με την περίπτωση $zone(A,B) = 1$, αν θεωρήσουμε ως άξονα $x'x$ του ορθοκανονικού συστήματος την ευθεία που

διχοτομεί την γωνία την ζωνης $zone(A, B)$, ως γωνία θ_{AV} την γωνία που ορίζεται από τον προηγούμενο άξονα και το ευθύγραμμο τμήμα AV, και ως γωνία θ_{BV} την γωνία που ορίζεται από την ευθεία που διχοτομεί την γωνία της ζώνης $zone(B, A)$ και το ευθύγραμμο τμήμα BV. Σε κάθε περίπτωση λοιπόν, αν οι κόμβοι A και B απέχουν απόσταση μεγαλύτερη από r, τότε η περιοχή των πιθανών verifier κόμβων όταν χρησιμοποιούμε τον αλγόριθμο strict verifier είναι μηδέν. □

Μπορούμε τώρα να δείξουμε ότι ο αλγόριθμος ανακάλυψης γειτόνων strict neighbor discovery ανιχνεύει την ύπαρξη μιας wormhole και δεν αφήνει τους κόμβους να επικοινωνήσουν μέσω αυτής.

Πρόταση 4.6. *Ο αλγόριθμος ανακάλυψης γειτόνων* STRICT NEIGHBOR DISCOVERY *δεν επιτρέπει να σχηματιστούν "ψεύτικοι" γείτονες, όταν ο εχθρός χρησιμοποιεί μόνο μια wormhole.*

Απόδειξη. Διακρίνουμε τις εξής 2 περιπτώσεις:

1η Περίπτωση: Για την απόσταση d μεταξύ των κόμβων ισχύει ότι $0 \leq d \leq r$

Τότε είναι προφανές ότι οι κόμβοι είναι ο ένας μέσα στην εμβέλεια του άλλου και άρα μπορούν σίγουρα να επικοινωνήσουν μεταξύ τους ανεξάρτητα από την ύπαρξη wormhole.

2η Περίπτωση: Για την απόσταση d μεταξύ των κόμβων ισχύει ότι $d > r$

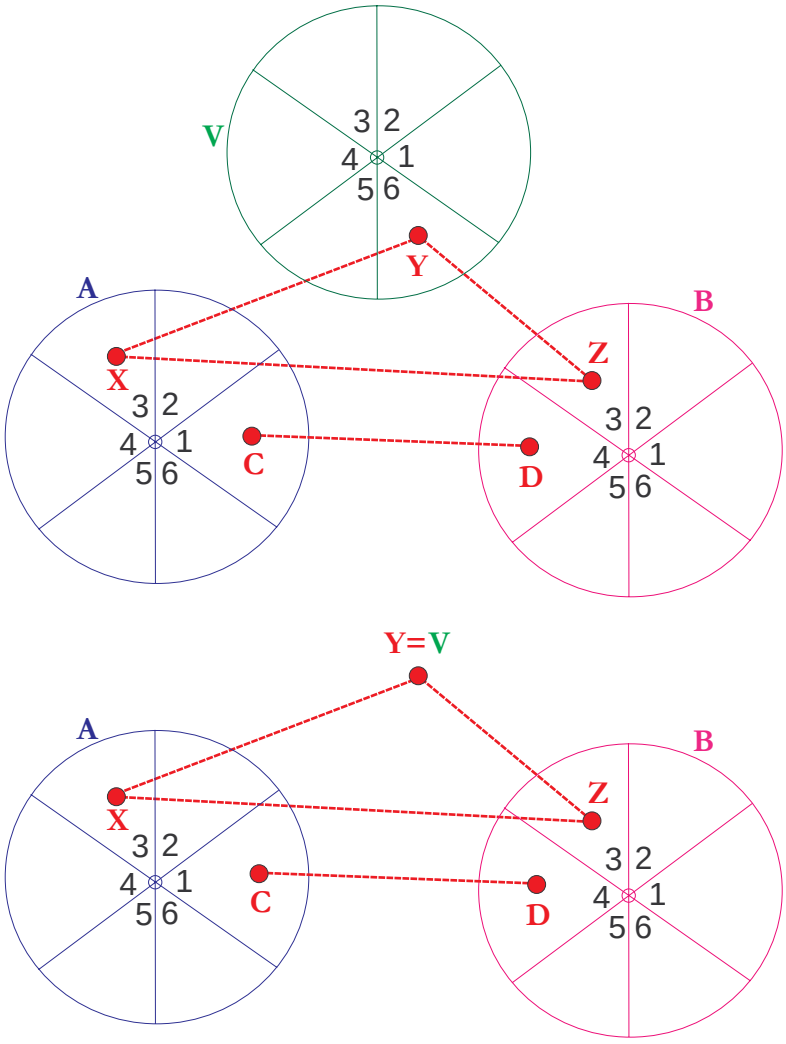
Από την πρόταση 4.5 έχουμε ότι η περιοχή των πιθανών verifier κόμβων σύμφωνα με τον αλγόριθμο strict neighbor discovery είναι μηδέν, επομένως οι κόμβοι δεν θα μπορέσουν να βρουν κανένα verifier κόμβο όπως απαιτεί ο αλγόριθμος και άρα δεν θα μπορέσουν να τον εκτέλεσουν, με αποτέλεσμα να μην αποδεχτεί ο ένας τον άλλο ως γείτονα.

Συνεπώς, βλέπουμε ότι σε κάθε περίπτωση ο εχθρός δεν μπορεί να επιτύχει μία wormhole attack. □

4.6 Συζήτηση σχετικά με τους παραπάνω αλγορίθμους

Όπως αποδείξαμε στην προηγούμενη ενότητα, ο STRICT NEIGHBOR DISCOVERY αλγόριθμος αποτρέπει τις wormhole επιθέσεις στο δίκτυο, όταν όμως ο εχθρός ελέγχει **μόνο μια wormhole (δηλαδή οι εχθρικοί κόμβοι που δημιουργούν wormholes είναι μόνο δύο)**. Ειδικότερα, ένας εχθρός ο οποίος ελέγχει πολλές wormholes θα μπορούσε να περικυκλώσει ένα κόμβο S με τα άκρα 6 wormholes που ελέγχει έτσι ώστε κάθε άκρο να καταλήγει σε μια ζώνη της κεραίας του. Έπειτα, συνδέοντας τα άλλα άκρα των wormholes μεταξύ τους (πιθανόν χρησιμοποιώντας και άλλους κόμβους του δικτύου), ο εχθρός θα μπορεί να στείλει ένα συγκεκριμένο μήνυμα σε οποιαδήποτε ζώνη της κεραίας του κόμβου S. Είναι προφανές όμως ότι για να μπορεί να ελέγξει τελείως τις επικοινωνίες ενός κόμβου με τον παραπάνω τρόπο, ο εχθρός θα χρειαστεί 6 wormholes για κάθε κόμβο που θέλει να ελέγξει, κάτι το οποίο είναι αρκετά απαιτητικό. Από αυτά όμως φαίνεται ότι **κανείς από τους τρεις αλγορίθμους που χρησιμοποιούν κατευθυνόμενες κεραίες δεν μας παρέχουν 100% ασφάλεια ενάντια σε Byzantine**

Overlay Wormhole επιθέσεις, μιας και ένας εχθρός ο οποίος έχει καταλάβει μερικούς κόμβους και έχει σχηματίσει ένα overlay δίκτυο είναι πολύ πιθανόν να έχει και από ένα κόμβο σε περισσότερες από μια ζώνες των κεραιών αρκετών κόμβων του αρχικού δικτύου, οπότε και να μπορεί να ελέγξει τις επικοινωνίες τους όπως περιγράψαμε πριν. Ακόμα, σε αυτή την περίπτωση ο εχθρός **θα μπορεί να χρησιμοποιήσει και κάποιον από τους κόμβους που έχει καταλάβει σαν verifier για κάποιο από τα wormholes του overlay δικτύου που έχει δημιουργήσει.** Δύο τέτοια παραδείγματα φαίνονται στο επόμενο σχήμα:



Σχήμα 4.16: Byzantine Overlay Wormhole επιθέσεις στον STRICT NEIGHBOR DISCOVERY αλγόριθμο

Και στα δύο σχήματα βλέπουμε ότι ο εχθρός έχει σχηματίσει το overlay δίκτυο με τους κόμβους X–Y–Z, **οι οποίοι είναι κόμβοι του αρχικού δικτύου** και ελέγχει την wormhole C–D. Έτσι, στο πρώτο σχήμα ο εχθρός απλά προωθεί τα μηνύματα της ζώνης 1 του A στη ζώνη 4 του B και τα μηνύματα των ζωνών 3 των A και B στη ζώνη 6 του κόμβου V, και ο αλγόριθμος STRICT NEIGH-

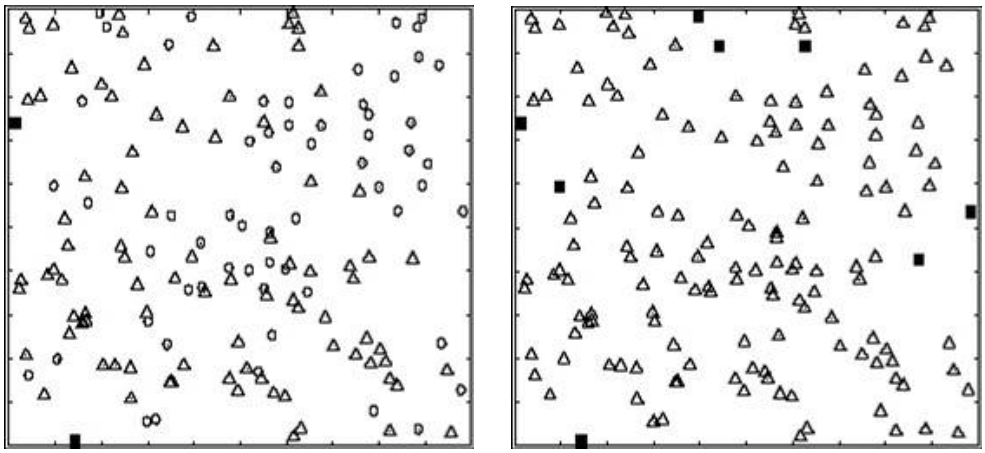
BOR DISCOVERY εκτελείται με επιτυχία. Επιπρόσθετα, επειδή οι κόμβοι X, Y, Z είναι κόμβοι του αρχικού δικτύου, υπάρχουν και έχουν μοιραστεί τα κλειδιά K_{AX}, K_{BX}, K_{VX} στον κόμβο X, τα κλειδιά K_{AY}, K_{BY}, K_{VY} στον κόμβο Y και τέλος τα κλειδιά K_{AZ}, K_{BZ}, K_{VZ} στον κόμβο Z, και άρα ο εχθρός μπορεί απλά να χρησιμοποιήσει τον κόμβο Y για να κατασκευάσει μηνύματα ενός strict verifier (ο οποίος υπάρχει στις ζώνες 3 των A και B), τα οποία μάλιστα θα παραδώσει και στις κατάλληλες ζώνες ώστε να εκτελεστεί με επιτυχία ο STRICT NEIGHBOR DISCOVERY αλγόριθμος, όπως φαίνεται στο δεύτερο σχήμα.

Ακόμα, ένας εχθρός θα μπορούσε να αποπροσανατολίσει την μαγνητική βελόνα της κεραίας ενός κόμβου, ώστε να αλλάξει η διάταξη των ζωνών της κεραίας και τελικά να καταφέρει να προωθήσει ένα μήνυμα στην κατάλληλη ζώνη. Ύστερα όμως θα έπρεπε να προσανατολίσει την κεραία του κόμβου όπως ήταν πριν, και γενικά να αλλάζει τη διάταξή της όποτε θέλει να στείλει κάποιο μήνυμα, κάτι το οποίο απαιτεί πολύ καλό συγχρονισμό με την προώθηση των μηνυμάτων και έτσι κάνει αυτού του είδους την επίθεση τρομερά δύσκολη.

Οι αλγόριθμοι οι οποίοι χρησιμοποιούν κατευθυνόμενες κεραίες που παρουσιάστηκαν έχουν το πλεονέκτημα ότι χρησιμοποιούν πολύ μικρό αριθμό μηνυμάτων για να σιγουρέψουν ότι μια σύνδεση προς ένα γείτονα δεν διέρχεται από κάποια wormhole, γιατί οπως φαίνεται και από την περιγραφή του STRICT NEIGHBOR DISCOVERY αλγορίθμου, ο μέγιστος αριθμός μηνυμάτων που θα ανταλλαχθεί μεταξύ των κόμβων είναι 3 μέχρι να ανακαλυφθεί η σύνδεση και το πολύ άλλα 6 για να επαληθευτεί η σύνδεση μέσω verifier. Το πρόβλημα που εμφανίζεται και με τον VERIFIED NEIGHBOR DISCOVERY αλλά και με τον STRICT NEIGHBOR DISCOVERY αλγόριθμο, είναι ότι μπορεί να μην υπάρχουν verifiers τους οποίους θα αναζητήσουν οι κόμβοι για την μεταξύ τους σύνδεση, αλλά η σύνδεση μεταξύ τους να είναι πραγματική (δηλαδή να μην διέρχεται μέσω wormhole). Έτσι, αυτές οι συνδέσεις, παρά το γεγονός ότι είναι έγκυρες, δεν θα γίνουν αποδεκτές από τους κόμβους με αποτέλεσμα να μειώνεται κάπως η απόδοση του ασυρμάτου δικτύου. Φυσικά, μπορούμε να δεχόμαστε κάθε σύνδεση η οποία αναγνωρίζεται μέχρι το 3ο βήμα του STRICT NEIGHBOR DISCOVERY αλγορίθμου (δηλαδή να χρησιμοποιούμε μόνο τον πρώτο αλγόριθμο DIRECTIONAL NEIGHBOR DISCOVERY), επειδή έτσι οι wormhole επιθέσεις μειώνονται κατά μέσο όρο στο $\frac{1}{6}$, αλλά όπως είπαμε πιο πριν αυτό δεν σημαίνει ότι μειώνεται το ίδιο και η απόδοση της wormhole επίθεσης **(αφού ακόμα και μια wormhole επηρεάζει πάνω από το 5% όλων των διαδρομών)**.

Έτσι λοιπόν, όσο πιο πυκνό είναι το δίκτυο μας, τόσο μικρότερη είναι η πιθανότητα να μην υπάρχει τουλάχιστον ένας verifier κόμβος για να επαληθεύσει την κάθε σύνδεση μεταξύ γειτονικών κόμβων, άρα η πιθανότητα να είναι ένας κόμβος αποκομμένος από το δίκτυο είναι αντιστρόφως ανάλογη με την πυκνότητα του δικτύου. Αξίζει να παρατηρήσουμε όμως, ότι όσο πιο κοντά είναι οι κόμβοι μεταξύ τους, τόσο πιο πιθανό είναι να μπορούν να βρουν verifier, και ακόμα ότι οι κόμβοι που είναι πιο πιθανόν να είναι αποκομμένοι από το υπόλοιπο δίκτυο είναι αυτοί που βρίσκονται στα όρια του δικτύου, για τον απλό λόγο ότι έχουν πολύ λιγότερους γειτονικούς κόμβους από τους άλλους.

Παρακάτω δίνουμε ένα παράδειγμα ενός αραιού ασυρμάτου δικτύου, και δείχνουμε ποιοι κόμβοι είναι αποκομμένοι και ποιοί όχι όταν χρησιμοποιούμε τον VERIFIED NEIGHBOR DISCOVERY και τον STRICT NEIGHBOR DISCOVERY αλγόριθμο.



Σχήμα 4.17: Επίδραση των αλγορίθμων στις συνδέσεις μεταξύ των κόμβων

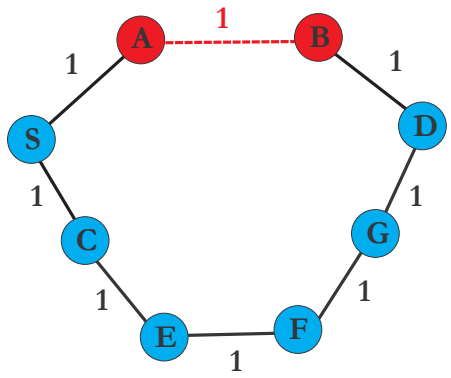
Στο αριστερό σχήμα χρησιμοποιούμε τον VERIFIED NEIGHBOR DISCOVERY και στο δεξί σχήμα τον STRICT NEIGHBOR DISCOVERY αλγόριθμο. Ακόμα, στο παραπάνω δίκτυο κάθε κόμβος έχει κατά μέσο όρο 3 γείτονες μέσα στην εμβέλεια της κεραίας του. Όταν λοιπόν χρησιμοποιούμε τον αλγόριθμο VERIFIED NEIGHBOR DISCOVERY το 14% όλων των συνδέσεων δεν γίνεται αποδεκτό από τους κόμβους, και μόνο το 1.3% των κόμβων είναι αποκομμένοι από το δίκτυο. Όταν χρησιμοποιούμε τον STRICT NEIGHBOR DISCOVERY αλγόριθμο, επειδή η περιοχή των verifiers μειώνεται σημαντικά, το 58% όλων των συνδέσεων δεν γίνεται αποδεκτό από τους κόμβους, και το 5.3% των κόμβων είναι αποκομμένοι από το δίκτυο. Στα παραπάνω σχήματα, οι κόμβοι που εμφανίζονται σαν κύκλοι έχουν αποδεχτεί όλες τις συνδέσεις προς τους γείτονές τους, οι κόμβοι που εμφανίζονται σαν τρίγωνα έχουν αποδεχτεί μερικές από τις συνδέσεις προς τους γείτονές τους, και οι κόμβοι που εμφανίζονται σαν μαύρα τετράγωνα είναι αποκομμένοι από το δίκτυο (δηλαδή δεν έχουν αποδεχτεί καμία σύνδεση με γείτονα λόγω έλλειψης verifier). Από την άλλη μεριά, σε ένα πιο συνηθισμένο δίκτυο (το οποίο είναι πιο πυκνό), όπου για παράδειγμα κάθε κόμβος έχει κατά μέσο όρο 10 γείτονες μέσα στην εμβέλεια της κεραίας του, τότε με τον verified neighbor discovery αλγόριθμο το 0.5% όλων των συνδέσεων δεν γίνεται αποδεκτό από τους κόμβους και κανένας κόμβος δεν είναι αποκομμένος από το δίκτυο, ενώ με τον strict neighbor discovery αλγόριθμο το 40% όλων των συνδέσεων δεν γίνεται αποδεκτό από τους κόμβους, και μόλις το 0.03% των κόμβων είναι αποκομμένοι από το δίκτυο. Γενικά, για δίκτυα τα οποία κάθε κόμβος έχει κατά μέσο όρο πάνω από 10 γείτονες μέσα στην εμβέλεια του, τότε σχεδόν όλες οι συνδέσεις προς τους γείτονες διατηρούνται αν οι κόμβοι απέχουν το πολύ $0.06r$ και χρησιμοποιούμε τον strict neighbor discovery αλγόριθμο (όπου r είναι η ακτίνα της κεραίας των κόμβων).

4.7 Επίλογος

Τέλος, αναφέρουμε παρακάτω τα πλεονεκτήματα που έχουν οι αλγόριθμοι ανίχνευσης γειτόνων που χρησιμοποιούν κατευθυνόμενες κεραίες σε σχέση με τις δύο άλλες μεθόδους αντιμετώπισης των wormholes:

- Η μέθοδος αντιμετώπισης με τα packet leashes απαιτεί ή να γνωρίζει ο κάθε κόμβος την θέση του ή όλοι οι κόμβοι να έχουν ακριβή συγχρονισμένα ρολόγια, κάτι το οποίο σημαίνει ότι θα πρέπει να υπάρχει μια υπηρεσία η οποία θα τους στέλνει πληροφορίες για την θέση τους (όπως για παράδειγμα το GPS) ή θα τους βοηθάει στο να έχουν τον απαιτούμενο συγχρονισμό της ώρας. Κατά συνέπεια, η υπηρεσία αυτή μπορεί να αποτελέσει στόχο επίθεσης, και αν η επίθεση επιτύχει, αυτό μπορεί να οδηγήσει μέχρι και σε αδυναμία επικοινωνίας ΟΛΩΝ των κόμβων μεταξύ τους (μιας και κάθε μήνυμα που θα σταλεί μεταξύ των κόμβων θα χρειάζεται κάποια πληροφορία που θα παρέχεται άμεσα ή έμμεσα από αυτή την υπηρεσία).

- Η μέθοδος αντιμετώπισης με την χρήση του πρωτοκόλλου ODSBR έχει το μειονέκτημα ότι χρησιμοποιεί μεγάλο αριθμό μηνυμάτων προκειμένου να αντιμετωπίσει τις επιθέσεις. Ας θεωρήσουμε το ασύρματο δίκτυο του παρακάτω σχήματος (όπου εμφανίζεται η wormhole X–Y και οι κόμβοι X και Y δεν αποτελούν μέρος του αρχικού δικτύου):



Σχήμα 4.18: Μειονέκτημα του ODSBR

Ας υποθέσουμε τώρα ότι οι κόμβοι S και D θέλουν να στείλουν ένα μήνυμα ο ένας στον άλλο. Αν χρησιμοποιήσουμε κάποιον από τους αλγορίθμους ανίχνευσης γειτόνων, **τότε αυτοί έχουν το πλεονέκτημα ότι θα χρειαστούν μόλις τρία μηνύματα για να αναγνωριστεί η ύπαρξη της wormhole A–B**, και αν οι κόμβοι δεν μετακινηθούν, το μήνυμα που θέλει να στείλει ο S στον D θα σταλεί μια φορά μέσω της μοναδικής διαδρομής $S \rightarrow C \rightarrow E \rightarrow F \rightarrow G \rightarrow D$ που θα ανακαλυφθεί, και όμοια το μήνυμα που θέλει να στείλει ο S στον D θα σταλεί μια φορά μέσω της μοναδικής διαδρομής $D \rightarrow G \rightarrow F \rightarrow E \rightarrow C \rightarrow S$, που θα ανακαλυφθεί, άρα για να φτάσουν τα μηνύματα στον προορισμό τους **θα χρειαστούν συνολικά τουλάχιστον 5 μηνύματα**.

Από την άλλη μεριά, αν χρησιμοποιήσουμε το πρωτόκολλο ODSBR, τότε, όπως είδαμε και στην ενότητα που περιγράψαμε την λειτουργία του ODSBR το μήνυμα που θα στείλει ο ο S στον D θα σταλεί 3 φορές για να φτάσει στον προορισμό του, μέχρι δηλαδή να αυξηθεί το κόστος της διαδρομής μέσω της wormhole αρκετά ώστε να προτιμηθεί η έγκυρη διαδρομή $S \rightarrow C \rightarrow E \rightarrow F \rightarrow G \rightarrow D$. Όμοια, το μήνυμα που θα στείλει ο ο D στον S θα σταλεί και αυτό 3 φορές για να φτάσει στον προορισμό του, δηλαδή για να φτάσουν τα μηνύματα στον προορισμό τους **θα χρειαστούν συνολικά τουλάχιστον 6 μηνύματα**.

Συνεπώς, όταν χρησιμοποιούμε το ODSBR, τότε κάθε φορά που η wormhole σύνδεση θα βρίσκεται σε μια διαδρομή, τότε το μήνυμα προς αποστολή θα στέλνεται πολλές φορές μέχρι το κόστος της προβληματική σύνδεσης να αυξηθεί αρκετά ώστε να προτιμηθεί μια έγκυρη διαδρομή. Δεδομένου ότι στις wormhole επιθέσεις τα άκρα της wormhole τοποθετούνται αρκετά μακριά (ώστε να επηρεάζουν περισσοτερες διαδρομές), αυτό μπορεί να σημαίνει ότι το πρωτόκολλο ODSBR θα χρειάζεται ένα μεγάλο αριθμό μηνυμάτων για να αντιμετωπίζει τις "απλές" wormhole επιθέσεις. Έτσι, μπορούμε να πούμε ότι στις περισσότερες περιπτώσεις οι αλγόριθμοι ανίχνευσης γειτόνων με τη χρήση κατευθυνόμενων κεραιών χρησιμοποιούν λιγότερους πόρους του δικτύου για να αντιμετωπίζουν τις wormholes σε σχέση με το πρωτόκολλο ODSBR, και άρα αποτελούν μια καλύτερη μέθοδο αντιμετώπισης τους.

Είναι φανερό λοιπόν ότι ανάλογα με το ασύρματο δίκτυο που έχουμε στην διάθεσή μας, μπορούμε να διαλέξουμε ποιά από τις τρεις μεθόδους αντιμετώπισης είναι η πιο κατάλληλη. Για παράδειγμα, αν το ασύρματο δίκτυο μας αποτελείται από κόμβους οι οποίοι έχουν περιορισμένη μπαταρία, τότε, επειδή η αποστολή μηνυμάτων καταναλώνει περισσότερη μπαταρία από την εκτέλεση υπολογισμών με τη χρήση του επεξεργαστή που διαθέτει ο κόμβος, η αντιμετώπιση με τις κατευθυνόμενες κεραίες είναι προτιμότερη από το να χρησιμοποιήσουμε το πρωτόκολλο ODSBR, αφού έτσι **θα εξασφαλίσουμε μεγαλύτερο χρόνο ζωής για τους κόμβους** και συνεπώς το ασύρματο δίκτυο θα μπορεί να χρησιμοποιηθεί για μεγαλύτερο χρονικό διάστημα.

Βιβλιογραφία

[1] L. Hu and D. Evans, *"Using directional antennas to prevent wormhole attacks"*, at the Eleventh Annual Network and Distributed System Security Symposium (NDSS 2004), 2004.

[2] Y. Hu, A. Perrig, and D. Johnson. *Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks*. INFOCOM 2003, April 2003.

[3] Yih-Chun Hu, Adrian Perrig and David B. Johnson. *Efficient Security Mechanisms for Routing Protocols*, in the proceedings of the Tenth Annual Network and Distributed System Security Symposium (NDSS 2003), 2003.

[4] G. Malkin. *RIP Version 2*. RFC 2453 (Standard). November 1998.

[5] A.J. Menezes, P.C. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996. Chapters availiable free at: `http://www.cacr.math.uwaterloo.ca/hac/`

[6] D. Johnson, D. Maltz, and J. Broch. *The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*. In Ad Hoc Networking, C. Perkins, Ed. Addison-Wesley, 2001.

[7] Charles Perkins and Pravin Bhagwat. *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*. Computer Communications Review, October 1994.

[8] Y. Hu, D. Johnson, and A. Perrig. *SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks*. IEEE Workshop on Mobile Computing Systems and Applications, June 2002.

[9] Y. Hu, A. Perrig and D. Johnson. *Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks*. ACM MobiCom 2002, September 2002.

[10] Panagiotis Papadimitratos and Zygmunt J. Haas. *Secure Routing for Mobile Ad hoc Networks*. In Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002), San Antonio, TX, January 27-31, 2002.

[11] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum et L. Viennot. *Optimized Link State Routing Protocol*. IEEE INMIC Pakistan 2001.

[12] Adrian Perrig, Yih-Chun Hu, and David B. Johnson. *Wormhole Protection in Wireless Ad Hoc Networks*. Technical Report TR01-384, Department of Computer Science, Rice University, December 2001.

[13] Refik Molva and Pietro Michiardi. *Security in Ad Hoc Networks*. `http://citeseer.ist.psu.edu/582085.html`

[14] Baruch Awerbuch, Reza Curtmola, David Holmer, Cristina Nita-Rotaru and Herbert Rubens. *Mitigating Byzantine Attacks in Ad Hoc Wireless Networks*. Technical Report, March 2004. `http://citeseer.ist.psu.edu/661767.html`

[15] C. Karlof and D. Wagner. *Secure Routing in Sensor Networks: Attacks and Countermeasures*. First IEEE International Workshop on Sensor Network Protocols and Applications, May, 2003.

[16] William A. Shay. *Understanding Communications and Networks* 3^e. Brooks/Cole–Thomson Learning, 2004.

[17] William Stallings. *Network Security Essentials: Applications and Standards*. Prentice Hall, New Jersey, 2000.

[18] Youlu Zheng, Shakil Akhtar. *Networks for Computer Scientists and Engineers*. Oxford University Press, New York, 2002.

[19] R. Choudhury and N. Vaidya. *Ad Hoc Routing Using Directional Antennas*. University of Illinois, Coordinated Science Laboratory, Technical Report, August 2002.

[20] Karygiannis, Tom, and Les Owens. *NIST Special Publication 800-48: Wireless Network Security 802.11, Bluetooth and Handheld Devices*. National Institute for Standars and Technology (NIST), November 2002. `http://csrc.nist.gov/publications/nistpubs/800-48/NIST_SP_800-48.pdf`

[21] S.M. Bellovin. *Security Problems in the TCP/IP Protocol Suite*. ACM Computer Communications Review, 19(2): 32-48, April 1989.