Πανεπιστήμιο Πειραιώς

Τμήμα Ψηφιακών Συστημάτων

Μεταπτυχιακό στην Ασφάλεια Ψηφιακών Συστημάτων

"Τεχνολογίες Ασφάλειας και Εμπιστοσύνης στα RFID συστήματα"

Διπλωματική εργασία υποβληθείσα για την απόκτηση του πτυχίου

υπό Καραπάνο Δημήτριο

Επιβλέπων καθηγητής: Δρ. Κωνσταντίνος Λαμπρινουδάκης

Πειραιάς, Μαίος 2011

Περιεχόμενα

Περιεχόμενα

Περιεχόμενα

Περιεχόμενα

Λίστα Σχημάτων

Λίστα Εικόνων

1. Εισαγωγή

Η «Αναγνώριση/Ταυτοποίηση μέσω Ραδιοσυχνοτήτων» (Radio Frequency IDentification – RFID) έχει αποκτήσει μεγάλη δημοτικότητα τα τελευταία χρόνια. Όπως θα δούμε και στη συνέχεια, τα RFID συστήματα προορίζονται να διαδραματίσουν σημαντικό ρόλο στην καθημερινότητα μας, ως αντικαταστάτες των ραβδόμορφων κωδικών (barcodes) σε καταναλωτικά αγαθά, από αυτοκίνητα και ηλεκτρονικές συσκευές μέχρι φάρμακα, ρούχα και ξυραφάκια. Αλλά οι εφαρμογές τους δεν περιορίζονται στην ταυτοποίηση αντικειμένων, αφού έχει προταθεί και υλοποιηθεί μία σειρά κρίσιμων εφαρμογών, όπως ηλεκτρονικά διαβατήρια και ταυτότητες, κυκλώματα ελέγχου πρόσβασης σε εγκαταστάσεις και συσκευές φύλαξης ιατρικού φακέλου ασθενούς. Όλες οι εκτιμήσεις των αναλυτών προβλέπουν ραγδαία αύξηση των εφαρμογών των συστημάτων RFID, καθώς και των οικονομικών μεγεθών που σχετίζονται με αυτά, μέσα στα επόμενα χρόνια. Ενδεικτικά, η έρευνα της IDTechEx για το 2008-2018 [1] δείχνει ότι, για το 2008, η συνολική αγορά, για συστήματα και υπηρεσίες RFID, ανέρχεται σε 5,29 δισεκατομμύρια USD, ενώ το 2007 η αγορά άξιζε 4,93 δισεκατομμύρια USD και προβλέπεται να ξεπεράσει τα 25 δισεκατομμύρια USD το 2018. Στη ίδια έρευνα αναφέρεται ότι μόνο το 2008 πωλήθηκαν συνολικά 2,16 δισεκατομμύρια RFID ετικέτες, δηλαδή διπλάσιος αριθμός σε σχέση με το 2006, ενώ ο Οργανισμός για την Οικονομική Συνεργασία και Ανάπτυξη (ΟΟΣΑ) σε έρευνά του αναφέρεται στα πλεονεκτήματα που θα έχει η τεχνολογία στις επιχειρήσεις και τα άτομα [2].
Η πληθώρα των εφαρμογών και χρήσεων που έχουν προταθεί ή αναμένεται να προταθούν στο εγγύς μέλλον, καθιστούν επιτακτική την ανάγκη για διερεύνηση και κατανόηση των επιπτώσεων που θα επιφέρει η χρήση της τεχνολογίας RFID στην καθημερινότητά μας.
Στην εργασία αυτή θα παρουσιάσουμε απειλές που προκύπτουν κατά της ιδιωτικότητας από τη χρήση των συστημάτων RFID. Επίσης στο δεύτερο κεφάλαιο θα παρουσιάσουμε ένα πρωτόκολλο που παρέχει αυθεντικοποίηση σε ετικέτες RFID και βασίζεται σε πράκτορες λογισμικού. Στο τρίτο κεφάλαιο θα γίνει μια ανάλυση της πλατφόρμας JADE που θα χρησιμοποιήσουμε στην υλοποιησή μας και στο τέταρτο κεφάλαιο θα γίνει ανάλυση της υλοποίησης μας και η χρησιμοποίηση της θα γίνει σε ένα σενάριο. Επίσης όλο ο πηγαίος κώδικας παρέχεται στο παράρτημα.

1.1. Η τεχνολογία RFID

Η «Αναγνώριση/Ταυτοποίηση μέσω Ραδιοσυχνοτήτων» (Radio Frequency IDentification – RFID) είναι ένας γενικός και μάλλον εμπορευματοποιημένος όρος, που αναφέρεται σε μία τεχνολογία, η οποία έχει πολλές υλοποιήσεις και ακόμη περισσότερες εφαρμογές. Στην πραγματικότητα, η RFID τεχνολογία δεν περιορίζεται τις ραδιοσυχνότητες για την επικοινωνία – μπορεί να χρησιμοποιήσει και την ηλεκτρομαγνητική επαγωγή – ενώ η αναγνώριση είναι μία μόνο από τις δυνατές εφαρμογές της [3].
Η τεχνολογία RFID πρωτοεμφανίστηκε τη δεκαετία του 1940 και αναφερόταν στην αναγνώριση φιλικών και εχθρικών πολεμικών αεροσκαφών. Το 1960 γίνονται οι πρώτες απόπειρες εμπορικής χρήσης της τεχνολογίας σε εφαρμογές αποτροπής κλοπής για ηλεκτρονικά αγαθά μεγάλης αξίας. Η αναβίωση της, όμως, τα τελευταία

χρόνια οφείλεται στη ραγδαία ανάπτυξη της τεχνολογίας των ημιαγωγών, που επέτρεψε τη δημιουργία πιο πολύπλοκων, μικρών, αξιόπιστων και φθηνών κυκλωμάτων, καθώς και στην εμπορική επιτυχία της εφαρμογής των RFID κυρίως σε συστήματα logistics.

Υπό την ομπρέλα του όρου RFID μπορούμε να τοποθετήσουμε πολλές διαφορετικές τεχνολογίες, από τις τυπικές ετικέτες RFID που υλοποιούν το πρότυπο του Ηλεκτρονικού Κώδικα Προϊόντος της EPCglobal, έως τις ετικέτες που έχουν πιστοποιηθεί για εμφύτευση σε ζωντανούς οργανισμούς (πρότυπα ISO 11784, 11785. 14223) και τις ανεπαφικές κάρτες (contactless cards) που ορίζονται στο πρότυπο ISO 14443 και χρησιμοποιούνται στα ηλεκτρονικά διαβατήρια. Υπό αυτές τις προϋποθέσεις, ένας ευρύτερος ορισμός περιγράφει την RFID ως μία τεχνολογία που επιτρέπει τη συλλογή δεδομένων με ανεπαφικές ηλεκτρονικές ετικέτες και ασύρματους αναμεταδότες - αναγνώστες για ταυτοποίηση και άλλους σκοπούς [3].

Για λόγους απλοποίησης θα θεωρήσουμε ότι ένα τυπικό σύστημα RFID αποτελείται από τέσσερα συστατικά: ετικέτες (tags), αναγνώστες (readers), back-end υπολογιστικό σύστημα ξενιστή (host computer system) και το λογισμικό που περιλαμβάνει τις εφαρμογές του back-end, τα πρωτόκολλα επικοινωνίας κ.λπ.

Μία RFID ετικέτα είναι μία μικρή ραδιο-συσκευή, που επικολλάται πάνω ή εμφυτεύεται στο αντικείμενο που θέλουμε να ταυτοποιήσουμε. Η ετικέτα αποτελείται από ένα απλό κύκλωμα σιλικόνης, προσκολλημένο σε μία μικρή επίπεδη κεραία και τοποθετημένο σε ένα υπόστρωμα. Η όλη συσκευή μπορεί να περιβληθεί από ένα προστατευτικό περίβλημα, π.χ. πλαστικό ή γυαλί, ανάλογα με την εφαρμογή. Η ετικέτα, πέρα από τις συνήθως περιορισμένες επεξεργαστικές ικανότητες που διαθέτει, φέρει και μνήμη, στην οποία μπορούν να αποθηκευτούν δεδομένα περιορισμένου μεγέθους, όπως ένας μοναδικός αναγνωριστικός κωδικός EPC που αποτελεί μοναδική ταυτότητα για κάθε ετικέτα/αντικείμενο. Για λόγους απλότητας, θα χρησιμοποιήσουμε τον όρο ετικέτα με την ευρεία έννοιά του, ώστε να περιλαμβάνει το σύνολο των υλοποιήσεων, από τις πλέον απλές, μόνο για ανάγνωση, ετικέτες, μέχρι και τις ανεπαφικές έξυπνες κάρτες.

Ο RFID αναγνώστης είναι μία συσκευή, συνήθως φορητή, που αποστέλλει και λαμβάνει δεδομένα προς και από τις ετικέτες, με ασύρματο τρόπο μέσω των κεραιών του. Τα δεδομένα που φτάνουν στον αναγνώστη, συνήθως ο μοναδικός αναγνωριστικός κωδικός EPC που αποστέλλει η ετικέτα, προωθούνται στη συνέχεια στο back-end υπολογιστικό σύστημα, προκειμένου αυτό να τα επεξεργαστεί και να παράγει χρήσιμα δεδομένα, όπως να αναζητήσει σε μία βάση δεδομένων τις πληροφορίες εκείνες που είναι συσχετισμένες με την ταυτότητα. Δεν είναι απαραίτητο ο αναγνώστης και το back-end σύστημα να είναι διαχωρισμένα.

Σχήμα 1. *Τυπικό σύστημα RFID που αποτελείται από το back-end υπολογιστικό σύστημα, αναγνώστες και ετικέτες*

Οι περιορισμοί που επιβάλλουν η μικρή χωρητικότητα της μνήμης της ετικέτας και οι, συνήθως, περιορισμένες δυνατότητες επεξεργασίας, αποθήκευσης και διασύνδεσης των αναγνωστών, περιορίζουν ως ένα βαθμό το σύνολο των πιθανών εφαρμογών. Για παράδειγμα, μία τυπική ετικέτα δεν έχει αρκετά μεγάλη μνήμη για να φυλάξει το σύνολο των πληροφοριών που θα επιθυμούσαμε να αποθηκεύσουμε για ένα προϊόν – τουλάχιστον σε μορφή αναγνώσιμη από τον άνθρωπο. Για αυτό το λόγο χρησιμοποιούνται υπολογιστικά συστήματα τα οποία αναλαμβάνουν το φόρτο της φιλοξενίας των εφαρμογών. Τα back-end συστήματα αυτά, συνήθως φιλοξενούν μία βάση δεδομένων για τη διαχείριση των ιδιόκτητων ετικετών, απαλλάσσουν τον αναγνώστη από το φόρτο επεξεργασίας των δεδομένων (π.χ. εκτέλεση κρυπτογραφικών αλγορίθμων, ερωτημάτων, ειδικευμένου λογισμικού κ.λπ.) και μπορούν να διασυνδεθούν με άλλα συστήματα και το Διαδίκτυο ώστε να εξυπηρετήσουν τις πιο πολύπλοκες εφαρμογές του χρήστη, όπως διαχείριση έξυπνου σπιτιού με χρήση RFID ετικετών για τον έλεγχο πρόσβασης, προσκόμιση πληροφοριών προϊόντος από το online σύστημα του κατασκευαστή, διαχείριση αλυσίδας εφοδιασμού κ.λπ.
Βέβαια, ανάλογα με την εφαρμογή, η μορφή και οι τρόποι αλληλεπίδρασης διαφέρουν. Για παράδειγμα, μπορούμε να φανταστούμε έναν 'έξυπνο' φούρνο μικροκυμάτων, που μπορεί να ‘ρωτάει’ το συσκευασμένο φαγητό για οδηγίες μαγειρέματος. Εξαιτίας της περιορισμένης μνήμης της, η ετικέτα δε φέρει όλη την απαραίτητη πληροφορία, αλλά μπορεί να απαντήσει με ένα διαδικτυακό σύνδεσμο – για την ακρίβεια με μία αλληλουχία bits που θα πρέπει να αντιστοιχηθεί σε ένα σύνδεσμο – προς μία ιστοσελίδα που θα περιέχει κατανοητές, για το φούρνο, οδηγίες μαγειρέματος.
Στη συνέχεια θα εξετάσουμε με μεγαλύτερη λεπτομέρεια τα τεχνικά χαρακτηριστικά των RFID ετικετών και αναγνωστών και τους τρόπους της μεταξύ τους επικοινωνίας [3].

1.1.1. RFID Ετικέτες

Η ευρύτητα των εφαρμογών που έχουν προταθεί για τα RFID συστήματα, έχει οδηγήσει στην ανάπτυξη ετικετών με διαφορετικά χαρακτηριστικά και δυνατότητες. Αυτό μας επιτρέπει να ακολουθήσουμε διάφορες ομαδοποιήσεις ανάλογα με τα χαρακτηριστικά που ερευνούμε, όπως για παράδειγμα το μέγεθός τους, τις επεξεργαστικές τους δυνατότητες, τη δυνατότητα εγγραφής στη μνήμη τους ή το φυσικό μέσο επικοινωνίας που χρησιμοποιούν (π.χ. συχνότητα ραδιοκύματος ή ηλεκτρομαγνητική επαγωγή).
Ένας συνήθης διαχωρισμός είναι αυτός μεταξύ ενεργών (active) και παθητικών (passive) ετικετών.
Οι ενεργές ετικέτες τροφοδοτούνται από μπαταρία ή άλλη αυτόνομη πηγή ενέργειας. Μπορούν να εκπέμψουν ένα σήμα προς τον αναγνώστη σε μεγάλες αποστάσεις, π.χ. μερικές εκατοντάδες μέτρα και συνήθως χρησιμοποιούνται για την ιχνηλάτηση (tracking) αγαθών υψηλής αξίας, όπως οχήματα και εμπορευματοκιβώτια. Συνήθως οι ενεργές ετικέτες λειτουργούν στις συχνότητες UHF και μικροκυμάτων.
Οι παθητικές ετικέτες δεν περιέχουν κάποια πηγή ενέργειας, αλλά μπορούν να αξιοποιούν τα ραδιοκύματα που εκπέμπει ο αναγνώστης. Ο αναγνώστης εκπέμπει σε μία ραδιοσυχνότητα χαμηλής ενέργειας, και η ετικέτα συλλαμβάνει τα ραδιοκύματα αυτά με την κεραία της και τα μετατρέπει στην ενέργεια που χρειάζεται για τη λειτουργία της. Λόγω του τρόπου πρόσληψης της ενέργειας, οι παθητικές ετικέτες υφίστανται περιορισμούς στην απόσταση εκπομπής, τυπικά μέχρι 3 μέτρα και στο

μέγεθος της μνήμης που μπορούν να έχουν. Όμως, το χαμηλό τους κόστος, μικρότερο του ενός ευρώ και η μεγάλη διάρκεια ζωής τους, τις κάνει ελκυστικές για μεγάλο εύρος εφαρμογών, όπως αυτές στις οποίες χρησιμοποιούνται σήμερα ραβδόμορφοι κώδικες. Οι παθητικές ετικέτες μπορούν και λειτουργούν, πέραν των UHF και μικροκυματικών συχνοτήτων και στις LF και HF συχνότητες.
Οι ενεργές ετικέτες προσφέρουν ορισμένα πλεονεκτήματα έναντι των παθητικών. Ήδη αναφέραμε ότι το σήμα που μπορούν να εκπέμψουν είναι πιο ισχυρό και άρα μπορούν να επικοινωνήσουν με τον αναγνώστη από μεγαλύτερες αποστάσεις. Επιπλέον, επειδή ο αναγνώστης δεν είναι υποχρεωμένος να τροφοδοτήσει την ετικέτα με το σήμα του, μπορεί να χρησιμοποιήσει ένα σήμα πολύ μικρότερης ισχύος. Ακόμη, το γεγονός ότι είναι ενεργειακά αυτόνομες, επιτρέπει στις ενεργές ετικέτες, να εκκινούν μία επικοινωνία, γεγονός που είναι απαραίτητο σε πολλές εφαρμογές, όπως για παράδειγμα οι εφαρμογές που ζητούν από μία ετικέτα-αισθητήρα (sensor) να ενημερώνει των αναγνώστη όταν μία περιβαλλοντική ένδειξη φτάσει κάποια τιμή κατωφλίου.
Χάρη στην εξέλιξη της τεχνολογίας και των διαδικασιών κατασκευής ηλεκτρονικών κυκλωμάτων πολλοί κατασκευαστές παράγουν συνεχώς νέα μοντέλα ετικετών με διαφορετικές ιδιότητες. Στόχος είναι φθηνότερες, μικρότερες και ανθεκτικότερες ετικέτες, που θα φιλοξενούν κυκλώματα μεγαλύτερης πολυπλοκότητας και θα μπορούν να αναγνωστούν από μεγαλύτερη απόσταση. Ενδεικτικά αναφέρουμε ότι πρόσφατα η εταιρία Hitachi παρήγαγε ετικέτες RFID με μέγεθος μόλις 0.05 x 0.05 χιλιοστά. Το προηγούμενο επίτευγμα κατείχε και πάλι ετικέτα της Hitachi, η μ-Chip, με μέγεθος 0,4 x 0,4 χιλιοστά, ενώ η εταιρία Mojix το 2008 παρουσίασε το Mojix STAR System, ένα RFID σύστημα με εμβέλεια ανάγνωσης, σε εσωτερικό περιβάλλον, έως και 300 πόδια (~180 μέτρα). Επιπλέον, η εταιρία Fujitsu έχει κατασκευάσει RFID ετικέτες οι οποίες είναι επιτρεπτό να πλένονται και να σιδερώνονται. Είναι εύκολα αντιληπτό ότι οι RFID ετικέτες μπορούν να τοποθετηθούν πρακτικά παντού.

Εικόνα 1. *Μία τυπική RFID ετικέτα, που χρησιμοποιείται για την αποτροπή κλοπών σε πολυκαταστήματα*

1.1.2. Προτυποποίηση

Προκειμένου να επιτευχθεί η διαλειτουργικότητα μεταξύ των RFID συσκευών διαφορετικών κατασκευαστών, έχουν συσταθεί επίσημοι φορείς προτυποποίησης. Στην Ευρώπη τα τεχνικά χαρακτηριστικά των συσκευών, όπως οι χρησιμοποιούμενες ραδιοσυχνότητες και η ισχύς τους, ρυθμίζονται από το Ευρωπαϊκό Ινστιτούτο Τηλεπικοινωνιακών Προτύπων ETSI (European Telecommunications Standards Institute), ενώ τα πρωτόκολλα επικοινωνίας προτείνονται από διάφορους φορείς και τους ίδιους τους κατασκευαστές. Οι δύο πιο σημαντικοί οργανισμοί προτυποποίησης

είναι ο Διεθνής Οργανισμός Προτυποποίησης ISO (International Organization for Standardization) και ο EPCglobal. Στο Σχήμα 2 φαίνονται ορισμένα βασικά πρότυπα που έχουν εκδοθεί.
Οι διάφοροι οργανισμοί και κατασκευαστές συνεργάζονται μεταξύ τους προκειμένου να επιτύχουν την απαιτούμενη διαλειτουργικότητα που θα επιτρέψει στα RFID να γίνουν μία πανταχού παρούσα τεχνολογία. Για παράδειγμα, τον Ιούλιο του 2006, ο Διεθνής Οργανισμός Προτυποποίησης ISO ενέκρινε το πρότυπο EPCGen 2 Class 1 UHF, της EPCGlobal, δημοσιεύοντάς το ως τροπολογία στο πρότυπο 18000-6, με την ονομασία ISO 18000-6C, για τις RFID συσκευές που λειτουργούν στις υψηλές συχνότητες 860-960 MHz. Τα υπόλοιπα έξι μέρη του προτύπου αφορούν γενικές οδηγίες για ασύρματες συσκευές (ISO 18000-1) και τεχνικές προδιαγραφές για τις υπόλοιπες συχνότητες (ISO 18000-2 έως ISO 18000-7).
προτείνονται από διάφορους φορείς και τους ίδιους τους κατασκευαστές. Οι δύο πιο σημαντικοί οργανισμοί προτυποποίησης είναι ο Διεθνής Οργανισμός Προτυποποίησης ISO (International Organization for Standardization) και ο EPCglobal. Στο Σχήμα 2 φαίνονται ορισμένα βασικά πρότυπα που έχουν εκδοθεί.
Οι διάφοροι οργανισμοί και κατασκευαστές συνεργάζονται μεταξύ τους προκειμένου να επιτύχουν την απαιτούμενη διαλειτουργικότητα που θα επιτρέψει στα RFID να γίνουν μία πανταχού παρούσα τεχνολογία. Για παράδειγμα, τον Ιούλιο του 2006, ο Διεθνής Οργανισμός Προτυποποίησης ISO ενέκρινε το πρότυπο EPCGen 2 Class 1 UHF, της EPCGlobal, δημοσιεύοντάς το ως τροπολογία στο πρότυπο 18000-6, με την ονομασία ISO 18000-6C, για τις RFID συσκευές που λειτουργούν στις υψηλές συχνότητες 860-960 MHz. Τα υπόλοιπα έξι μέρη του προτύπου αφορούν γενικές οδηγίες για ασύρματες συσκευές (ISO 18000-1) και τεχνικές προδιαγραφές για τις υπόλοιπες συχνότητες (ISO 18000-2 έως ISO 18000-7).

Σχήμα 2. *Κυριότερα RFID πρότυπα (Πηγή: OECD 2008)*

Η προσπάθεια προτυποποίησης δε σταματά στις τεχνικές φυσικές προδιαγραφές του RFID, αλλά επεκτείνεται και σε άλλους τομείς, όπως η κωδικοποίηση των δεδομένων (ISO 15962 Radio Frequency IDentification for item management –Data protocol: data encoding rules and logical memory functions), τα πρωτόκολλα επικοινωνίας (ISO 15961 Radio frequency identification (RFID) for item management – Data protocol: application interface) κ.λπ. Ένα βασικό πρότυπο, το οποίο, όμως, δε σχετίζεται άμεσα με τα τεχνικά/φυσικά χαρακτηριστικά της RFID ετικέτας, αλλά με

τα δεδομένα που καλείται να διαχειριστεί, είναι ο Ηλεκτρονικός Κωδικός Προϊόντος (Electronic Product Code – EPC). Ο EPC είναι μία οικογένεια διατάξεων κωδικοποίησης που δημιουργήθηκαν ως αντικαταστάτες του ραβδόμορφου κώδικα. Οι διατάξεις αυτές σχεδιάστηκαν ώστε να διασφαλίζουν τη μοναδικότητα κάθε ετικέτας συμβατής με το EPC. Συνεπώς κάθε EPC ετικέτα μπορεί να ταυτοποιεί μοναδικά κάθε αντικείμενο που παράγεται και όχι απλώς την κλάση του προϊόντος, όπως συμβαίνει με το ραβδόμορφο κώδικα. Το πρότυπο EPC βρίσκεται στην τρίτη έκδοσή του και ορίζει τις ακόλουθες διατάξεις:

- General Identifier (GID) GID-96
- Μια σειριακή έκδοση του GS1 Global Trade Item Number (GTIN) SGTIN-96 SGTIN-198
- GS1 Serial Shipping Container Code (SSCC) SSCC-96
- GS1 Global Location Number (GLN), SGLN-96 SGLN-195
- GS1 Global Returnable Asset Identifier (GRAI) GRAI-96 GRAI-170
- GS1 Global Individual Asset Identifier (GIAI) GIAI-96 GIAI-202 and
- DOD Construct DoD-96

Άλλες διατάξεις ονοματοθεσίας και κωδικοποίησης προτείνουν τη χρήση του συστήματος αρίθμησης του IPv6 για την αναγνώριση των αντικειμένων [4][5]. Η χρήση του IPv6 προτάθηκε αρχικά για τα συστήματα του στρατού των ΗΠΑ, σε περίπτωση που τα πρότυπα του EPCglobal κρίνονταν ανεπαρκή για τις στρατιωτικές εφαρμογές.

1.1.3. Αλγόριθμοι Επικοινωνίας

Συνήθως οι απλοί RFID αναγνώστες μπορούν να επικοινωνήσουν μόνο με μία ετικέτα κάθε στιγμή. Εάν περισσότερες της μιας ετικέτας απαντήσουν στο ερώτημα, ο αναγνώστης δε θα μπορέσει να διαβάσει καμία ετικέτα λόγω της σύγκρουσης. Για να λυθεί αυτό το πρόβλημα, έχει προταθεί μία σειρά από τεχνικές απομόνωσης (singulation) [6], οι οποίες επιτρέπουν στον αναγνώστη να επικοινωνήσει με όλες τις ετικέτες. Δύο είναι οι βασικές τεχνικές που χρησιμοποιούνται: Ο αλγόριθμος διάσχισης δυαδικού δένδρου, οποίος συναντάται κυρίες στις UHF ετικέτες και μία παραλλαγή του ALOHA αλγορίθμου, η οποία χρησιμοποιείται κυρίως στις HF ετικέτες.
Να σημειωθεί ότι, η σχεδίαση ενός αλγορίθμου απομόνωσης, θα πρέπει να λαμβάνει ιδιαίτερη μέριμνα για τις απαιτήσεις ασφάλειας και ιδιωτικότητας, καθώς, όπως θα δούμε στο παρόν κεφάλαιο, η ισχύς του σήματος του αναγνώστη τον καθιστά ιδανικό στόχο για υποκλοπείς. Θα πρέπει, λοιπόν, να περιορίζεται ή καλύτερα να αποφεύγεται η μετάδοση, εκ μέρους του αναγνώστη, σημαντικών πληροφοριών, όπως είναι τα μοναδικά αναγνωριστικά των ετικετών και τα μυστικά συνθηματικά χωρίς κρυπτογράφηση.

Αλγόριθμος ALOHA

Στα πιθανοτικά πρωτόκολλα αποφυγής σύγκρουσης (anti-collision) που βασίζονται στο Aloha, κάθε ετικέτα καθυστερεί κατά ένα τυχαίο χρονικό διάστημα, σε σχέση με το σήμα του αναγνώστη, προτού απαντήσει σε ένα ερώτημα. Εάν συμβεί μία σύγκρουση, ο αναγνώστης ενημερώνει τις ετικέτες που βρίσκονται εντός της εμβέλειάς του και τις αναγκάζει να καθυστερήσουν για ένα ακόμη τυχαίο χρονικό διάστημα προτού απαντήσουν. Αυτοί οι αλγόριθμοι λειτουργούν χωρίς να χρειάζεται

ο αναγνώστης να εκπέμψει καμία σημαντική πληροφορία (π.χ. μοναδικά αναγνωριστικά ετικετών), καθώς το μόνο που μεταδίδεται από πλευράς αναγνώστη είναι εντολές. Υπάρχουν, όμως, τροποποιήσεις, που επιτυγχάνουν καλύτερη αξιοποίηση του καναλιού, ορίζοντας ότι ο αναγνώστης πρέπει να είναι σε θέση να σιωπήσει κάθε ετικέτα που έχει ήδη αναγνωριστεί. Ένας τέτοιος μηχανισμός επιλογής, όμως, μπορεί να οδηγήσει σε διαρροή πληροφορίας αφού η ισχύς του σήματος του αναγνώστη, όπως είπαμε, είναι τέτοια που θα μπορούσε να επιτρέψει σε έναν υποκλοπέα να καταγράψει τα (μοναδικά) αναγνωριστικά των ετικετών από απόσταση.

Αλγόριθμος Διάσχισης Δυαδικού Δένδρου (Binary Tree-Walking)

Αντίθετα από τους πιθανοτικούς αλγορίθμους, η Διάσχιση Δυαδικών Δένδρων είναι ένας νομοτελειακός αλγόριθμος που επιτρέπει την αναγνώριση μεμονωμένων ετικετών χρησιμοποιώντας μία διαδικασία bit προς bit ερωτημάτων, που θυμίζει αναζήτηση κατά βάθος ενός δυαδικού δένδρου. Οι τεχνικές λεπτομέρειες της διαδικασίας δε θα μας απασχολήσουν στο παρόν κεφάλαιο και θα περιοριστούμε στις ενδεχόμενες επιπτώσεις του στην ασφάλεια και την ιδιωτικότητα. Ο κλασσικός αλγόριθμος απαιτεί από τον αναγνώστη να μεταδώσει το πρόθεμα (τα αρχικά bits) του μοναδικού αναγνωριστικού της ετικέτας που θέλει να διαβάσει. Το πλήθος των bits που πρέπει να μεταδώσει εξαρτάται από το πλήθος των συγκρούσεων που θα εντοπίσει ο αναγνώστης. Εάν, για παράδειγμα, υπάρχουν δύο ετικέτες που διαφέρουν μόνο στα τρία τελευταία bits του αναγνωριστικού τους, τότε για να επιλέξει ο αναγνώστης τη μία από τις δυο θα πρέπει να μεταδώσει όλα τα αρχικά bits μέχρι και το προτελευταίο. Αυτή η πληροφορία, όμως, μπορεί να αξιοποιηθεί από έναν κακόβουλο ωτακουστή.

1.1.4. Εφαρμογές της τεχνολογίας RFID

Η κύρια χρήση των RFID ετικετών είναι να αντικαταστήσουν τους ραβδόμορφους κώδικες (barcodes). Τα πλεονεκτήματά τους, έναντι των ραβδόμορφων κωδικών, περιλαμβάνουν την ανάγνωση χωρίς να χρειάζεται οπτική επαφή, την ανάγνωση πολλαπλών ετικετών/αντικειμένων, την υψηλή ακρίβεια ανάγνωσης και το επίπεδο αυτοματοποίησης και φύλαξη περισσότερης πληροφορίας. Το κυριότερο πλεονέκτημά τους, όμως, είναι ότι μπορούν να προσφέρουν ένα μοναδικό αναγνωριστικό, όχι ανά κατασκευαστή ή σειρά προϊόντος, αλλά για κάθε αντικείμενο. Κάθε μεμονωμένο αντικείμενο που φέρει ετικέτα, έχει μία μοναδική ταυτότητα EPC, η οποία μπορεί να χρησιμοποιηθεί για να το ανιχνεύσουμε σε όλη τη διάρκεια της ζωής του. Οι πιθανές χρήσεις αυτής της δυνατότητας είναι πολλές. Για παράδειγμα, μπορεί να χρησιμοποιηθεί στον έλεγχο ιδιοκτησίας (καταγραφή και φύλαξη των ταυτοτήτων κάθε ιδιόκτητου αντικειμένου), για την παροχή υπηρεσιών μετά την πώληση (after sales services), ή ακόμη και για την αυτοματοποίηση της ανακύκλωσης.

Όπως, όμως, έχουμε ήδη αναφέρει, οι εφαρμογές της τεχνολογίας RFID δεν περιορίζονται στην αναγνώριση αγαθών, αλλά επεκτείνουν με πολύ ευφάνταστο τρόπο αυτή τη βασική δυνατότητα. Ξεκινώντας το 1939 στα συστήματα αναγνώρισης φιλικών/εχθρικών πολεμικών αεροσκαφών και ύστερα στην διαχείριση αποθέματος και της αλυσίδας εφοδιασμού [7], το πλήθος των προτεινόμενων εφαρμογών συνεχώς διευρύνεται. Ενδεικτικά, τα RFID συστήματα μπορούν να χρησιμοποιηθούν στην αυτοματοποίηση των πληρωμών, στον έλεγχο φυσικής πρόσβασης σε αυτοκίνητα, σε έξυπνες οικίες και γραφεία, αλλά και για την αντιμετώπιση της πλαστογραφίας και

την αποτροπή κλοπών από καταστήματα, καθώς και για τη διαχείριση αποσκευών από αεροπορικές εταιρίες. Γενικότερα η ενσωμάτωση των RFID ετικετών σε προϊόντα, συσκευασμένα φαγητά με ενδείξεις καταλληλότητας (ημερομηνίες λήξης, προειδοποιήσεις για αλλεργίες κ.λπ.), έγγραφα με συνόψεις του εγγράφου, διαβατήρια και ταυτότητες ([8]), έξυπνα εισιτήρια μέσων μαζική μεταφοράς, συσκευές αυτόματης πληρωμής διοδίων, ακόμη και χρήματα [9], συνεχώς διευρύνεται.
Μία άλλη δημοφιλής εφαρμογή είναι η εμφύτευση ετικετών σε κατοικίδια/αγροτικά ζώα που φέρουν πληροφορίες για το ιατρικό ιστορικό τους και τον ιδιοκτήτη τους. Η Verichip Corp. και άλλες εταιρίες έχουν κατασκευάσει RFID εμφυτεύματα με πολύ μικρό μέγεθος για χρήση σε ανθρώπους. Το RFID της Verichip έχει εγκριθεί από τη U.S. Food and Drug Administration και ήδη αξιοποιείται σε εμπορικές και φαρμακευτικές εφαρμογές. Η είδηση ότι το γραφείο του Γενικού Εισαγγελέα του Μεξικό χρησιμοποιεί RFID εμφυτεύματα για τον έλεγχο της πρόσβασης στο υπολογιστικό κέντρο δίωξης εγκλήματος είχε προκαλέσει διενέξεις γύρω από τα ηθικά ζητήματα που εγείρονται, ενώ και σε σχολείο στην Osaka της Ιαπωνίας και ένα άλλο στο Doncaster του Ηνωμένου Βασιλείου, έχουν ραφτεί στις σχολικές στολές ετικέτες RFID που χρησιμοποιούνται για τον έλεγχο της θέσης των μαθητών, ενώ το σχολείο στο Doncaster έχει δηλώσει ότι στο μέλλον σκοπεύει να προσθέσει προσωπικά δεδομένα στις ετικέτες, τα οποία θα επιτρέπουν στο εκπαιδευτικό προσωπικό να διαβάζει, ασύρματα, τα στοιχεία και την πρόοδο του μαθητή με τη χρήση ενός αναγνώστη. Στο ίδιο πλαίσιο, οι κατασκευαστές ευελπιστούν ότι, στο κοντινό μέλλον, η τεχνολογία RFID θα χρησιμοποιείται και για την αναγνώριση των πελατών στα καταστήματα με στόχο την καλύτερη εξυπηρέτησή τους, με την παροχή προσωποποιημένων υπηρεσιών και διαφημίσεων.

1.2. Απειλές και Επιθέσεις κατά της Ιδιωτικότητας

Η ευρεία διαθεσιμότητα προσωπικών δεδομένων, με χαμηλό κόστος, που διατείνονται ότι θα προσφέρουν οι ετικέτες RFID, διευκολύνει τη νόμιμη, ημι-νόμιμη ή ακόμη και παράνομη επεξεργασία τους. Η αποκάλυψη πληροφοριών, όπως ιατρικά δεδομένα και προσωπικές προτιμήσεις, μπορεί να οδηγήσει σε διακρίσεις από ασφαλιστικές εταιρίες [10], στο χώρο εργασίας κ.λπ. αποκάλυψη προσωπικών πληροφοριών σε παρόχους υπηρεσιών και μεταπράτες, για τη βελτίωση της ποιότητας των παρεχόμενων υπηρεσιών/αγαθών, μπορεί να έχει δυσάρεστες επιπτώσεις, όπως η επιλεκτική διαφοροποίηση τιμών, η αποστολή ανεπιθύμητων προσωποποιημένων διαφημίσεων και η απώλεια της ανωνυμίας στις καθημερινές αγορές.
Δεδομένου ότι η κύρια χρήση της τεχνολογίας RFID είναι η χαμηλού κόστους αυτοματοποιημένη αναγνώριση, τα περισσότερα πρωτόκολλα επικοινωνίας δεν ενσωματώνουν μηχανισμούς προστασίας των δεδομένων από τη μη εξουσιοδοτημένη αποκάλυψη. Οι ετικέτες χαμηλού κόστους επικοινωνούν με τους αναγνώστες πάνω από ένα απροστάτευτο κανάλι, συχνά χωρίς να απαιτούν αυθεντικοποίηση του αναγνώστη[11]. Υπό αυτές τις συνθήκες, είναι πιθανή η παρακολούθηση/αναγνώριση ενός ατόμου από τα γυαλιά που φοράει, τα κλειδιά του ή το πορτοφόλι του, διαφανώς, χωρίς να ζητείται η συγκατάθεσή του.
Αλλά και οι πιο εξελιγμένες τεχνολογικά ετικέτες, όπως οι ανεπαφικές κάρτες που υλοποιούν το πρότυπο ISO 14443, και διαθέτουν έλεγχο πρόσβασης για την ανάγνωση του προστατευόμενου μέρους της μνήμη τους (αυτή τη στιγμή το ISO

14443 χρησιμοποιείται στα ηλεκτρονικά διαβατήρια δεύτερης γενεάς στη Μαλαισία και αλλού [8]) λόγω των πρωτοκόλλων αποφυγής σύγκρουσης που χρησιμοποιούν, διαρρέουν το μοναδικό αναγνωριστικό τους. Έτσι, αν το μοναδικό αναγνωριστικό ενός διαβατηρίου συσχετιστεί με τον ιδιοκτήτη του, κάθε επόμενη φορά που ο ιδιοκτήτης θα χρησιμοποιεί το διαβατήριό του θα μπορεί να ταυτοποιηθεί από μη εξουσιοδοτημένους χρήστες, χωρίς να απαιτείται πρόσβαση στα προστατευμένα δεδομένα της μνήμης.
Ακολούθως θα περιγράψουμε ορισμένες σημαντικές απειλές κατά της ιδιωτικότητας, που μπορεί να επιφέρει η είσοδος των RFID και των εφαρμογών τους στην καθημερινότητά μας. Ακόμη, θα αναφέρουμε τις κυριότερες επιθέσεις από τις οποίες κινδυνεύουν τα συστήματα RFID.

1.2.1. Απειλές στα συστήματα RFID

Πλήθος Πληροφοριών
Η χρήση RFID τεχνολογίας σε έγγραφα που φέρουν πληροφορίες για ένα άτομο, όπως διαβατήρια, ταυτότητες, κάρτες δημόσιας ασφάλισης, επαγγελματικές ταυτότητες, άδειες οδηγήσεως, έξυπνα εισιτήρια κ.λπ. αποτελεί μέγιστη απειλή για την ιδιωτικότητα και ιδιαίτερα την ανωνυμία του ατόμου, σε περίπτωση που δεν υλοποιηθούν τα κατάλληλα αντίμετρα, τα οποία θα δίνουν στο χρήστη ικανοποιητικό έλεγχο της πρόσβασης στα δεδομένα αυτά [12][13]. Μία υλοποίηση που θα επιτρέπει την ελεύθερη ασύρματη πρόσβαση στα δεδομένα, π.χ. ενός διαβατηρίου, θα έδινε τη δυνατότητα σε οποιονδήποτε διαθέτει έναν κατάλληλο RFID αναγνώστη και βρίσκεται σε κατάλληλη απόσταση, να αποκτήσει πλήθος προσωπικών πληροφοριών παραβιάζοντας κάθε έννοια ανωνυμίας και ιδιωτικότητας, καθώς τα δεδομένα αυτά μπορούν να χρησιμοποιηθούν στο πλαίσιο πληθώρας κακόβουλων σκοπών, όπως η κλοπή της ταυτότητας ενός ατόμου.
Αλλά ακόμη και αν χρησιμοποιηθεί κάποια προστατευτική τεχνολογία, η οποία θα περιορίζει την πρόσβαση από μη εξουσιοδοτημένους χρήστες, όπως προαναφέραμε με το ISO 14443, η χρήση RFID ετικετών δεν παύει να εγείρει ανησυχίες. Για παράδειγμα, είναι δύσκολο για τον κάτοχο μιας RFID ταυτότητας να έχει πλήρη έλεγχο στο είδος και στο πλήθος των πληροφοριών που μεταδίδονται, αφού δεν έχει το φυσικό έλεγχο του αναγνώστη, αλλά μόνον της ετικέτας, η οποία όμως δεν προβλέπεται να διαθέτει διεπαφή χρήστη. Το σενάριο του ελέγχου από τον κάτοχο ποια δεδομένα του διαβατηρίου του διαβάζονται από τον ελεγκτή σε ένα αεροδρόμιο, είναι αντιοικονομικό και εξάλλου έρχεται σε αντίφαση με το ισχύον σύστημα των χάρτινων διαβατηρίων.
Πρέπει να τονιστεί ότι, από μόνη της, η ηλεκτρονική μορφή των δεδομένων αποτελεί σημαντική απειλή για την ιδιωτικότητα του ατόμου καθώς διευκολύνει σε μέγιστο βαθμό την επεξεργασία των δεδομένων αυτών. Η αντιγραφή και φύλαξη, επ' αόριστον, των πληροφοριών μίας ταυτότητας θα μπορεί να γίνει σε κλάσματα δευτερολέπτου χωρίς να γίνεται αντιληπτή από τον κάτοχό της. Ακόμη και αν δοθούν τεχνολογικές και νομικές εγγυήσεις, ότι μόνον εξουσιοδοτημένα άτομα θα έχουν πρόσβαση στα δεδομένα αυτά – π.χ. όργανα της τάξεως, υπηρεσίες ασφάλειας κ.λπ. – η ευκολία απόκτησης των δεδομένων αποτελεί από μόνη της απειλή για την ιδιωτικότητα. Και αν ορισμένες χώρες διαθέτουν δημοκρατική παράδοση και λειτουργούντες θεσμούς, υπάρχουν λιγότερο φιλελεύθερα καθεστώτα τα οποία θα μπορούσαν να εκμεταλλευτούν μία τέτοια δυνατότητα. Ένα τέτοιο σενάριο δεν είναι καθόλου φανταστικό, όπως έδειξε η περίπτωση της Μαλαισίας, στην οποία "τον Ιανουάριο του 1999 το γραφείο του Πρωθυπουργού ανακοινώνει ότι τα

μουσουλμανικά ζευγάρια θα εφοδιάζονται στο εξής με μικροτσίπ για να αποδεικνύουν το καθεστώς του γάμου τους, ώστε η ισλαμική αστυνομία, έχοντας στη διάθεσή της ηλεκτρονικά όργανα, να επαληθεύει αν δύο άτομα αντίθετου φύλλου που εντοπίζονται μαζί είναι παντρεμένα ή πρέπει να συλληφθούν για έγκλημα 'κάλους' ή αλλιώς για παράνομη 'κοντινή γειτνίαση'..."

Διαφάνεια Επικοινωνίας

Πολλές σύγχρονες υλοποιήσεις επιτρέπουν την απρόσκοπτη ανάγνωση RFID ετικετών. Ο κάτοχος μιας ετικέτας, εξ ορισμού, έχει πολύ μικρό έλεγχο στην ετικέτα, η οποία αρέσκεται στη 'φλυαρία' με κάθε αναγνώστη που της απευθύνει ένα ερώτημα. Αυτή η αδυναμία έλεγχου των πληροφοριών που διαρρέουν από τις ετικέτες, παραβιάζει την ιδιωτικότητα και την ανωνυμία του κατόχου τους. Ο φορέας μιας ή περισσότερων ετικετών εκτίθεται στα αδιάκριτα 'ερωτήματα' των γειτονικών αναγνωστών και το συνάθροισμα της διαρρέουσας πληροφορίας δύσκολα γίνεται αντιληπτό.

Ακόμη και στη περίπτωση που η ετικέτα υλοποιεί διαδικασίες ελέγχου πρόσβασης ή/και αυθεντικοποίησης του RFID αναγνώστη, παραμένει ο κίνδυνος από την ασύρματη μετάδοση των δεδομένων. Εκτός από την περίπτωση που υλοποιηθούν μηχανισμοί κρυπτογράφησης ή συγκάλυψης της κίνησης – μηχανισμοί, όμως, που είναι οικονομικά ασύμφοροι για τις περισσότερες εφαρμογές και προσθέτουν προβλήματα διαχείρισης κλειδιών και πιστοποιητικών – είναι πολύ εύκολο για έναν μη εξουσιοδοτημένο αναγνώστη να κρυφακούσει την επικοινωνία[3].

Αποκάλυψη Δράσης

Αυτή η απειλή αναφέρεται στη δυνατότητα να συμπεράνουμε τη συμπεριφορά ενός ατόμου, από την απλή παρακολούθηση των ενεργειών μιας ομάδας ετικετών. Στο πρόσφατο παρελθόν, ένας μεγάλος μεταπράτης είχε κατηγορηθεί από ενώσεις. καταναλωτών ότι εφάρμοζε πειραματικά, χωρίς να το έχει κάνει γνωστό, φωτογραφικό σύστημα 'έξυπνων ραφιών'. Ένα τέτοιο σύστημα βασίζεται στην υπόθεση ότι η ξαφνική εξαφάνιση ετικετών, που αντιστοιχούν σε αγαθά μεγάλης αξίας, είναι μία σοβαρή ένδειξη ότι ένα άτομο, πελάτης ή υπάλληλος, σκοπεύει να τα κλέψει. Όταν λοιπόν η επικοινωνία αναγνώστη-ετικέτας δεν είναι δυνατή, πραγματοποιείται αυτόματη φωτογράφηση των ατόμων που βρίσκονται κοντά στο σημείο όπου βρίσκονταν οι ετικέτες μέχρι να εξαφανιστούν. Οι ενώσεις καταναλωτών θεώρησαν ότι ο αυτοματοποιημένος στιγματισμός ατόμων ως δυνητικών παραβατών παραβίαζε βασικά δικαιώματα, καθώς η εξαφάνιση μιας ετικέτας θα μπορούσε να έχει πληθώρα ερμηνειών, όπως για παράδειγμα μία ατυχή πτώση ή ένα άτομο που παρεμβάλλεται μεταξύ της ετικέτας και του αναγνώστη.

Συσχέτιση

Όταν ένας πελάτης αγοράζει ένα προϊόν που φέρει μία EPC ετικέτα, η ταυτότητα του πελάτη μπορεί να συσχετιστεί με το μοναδικό σειριακό ηλεκτρονικό κωδικό του προϊόντος. Η διαφορά με τη δυνατότητα συσχέτισης που προσφέρουν οι κάρτες ανταμοιβής αγορών (loyalty/reward cards), είναι ότι, εξαιτίας του EPC, ένας αγοραστής θα μπορεί πλέον να συσχετίζεται με ένα συγκεκριμένο αντικείμενο, π.χ. ένα συγκεκριμένο μπουκάλι γάλα, παρά με ένα τύπο προϊόντος. Επίσης, αντίθετα από τις κάρτες ανταμοιβής αγορών όπου η χρήση ή όχι είναι στη διακριτική ευχέρεια του πελάτη, η τεχνολογία RFID επιτρέπει την δίχως έγκριση και ειδοποίηση καταγραφή/συσχέτιση, καθώς ο πελάτης δεν μπορεί να αποφύγει την ανάγνωση της

ετικέτας στο ταμείο. Αν, επιπλέον, συνδυαστεί αυτή η δυνατότητα με τη χρήση πιστωτικών καρτών, τότε η συσχέτιση γίνεται πιο ισχυρή, καθώς η ταυτότητα του αγοραστή γίνεται πλέον γνωστή (η αγορά με μετρητά είναι μία ευρέως γνωστή πρακτική διατήρησης της ανωνυμίας στις αγορές). Από την άλλη πλευρά, η αδυναμία επιβεβαίωσης μιας συσχέτισης, π.χ. στην αγορά ενός δώρου ο πραγματικός ιδιοκτήτης είναι ο αποδέκτης του δώρου και όχι ο αγοραστής, ή σε περίπτωση δανεισμού, μπορεί να επιφέρει πολλά προβλήματα με τη σειρά της, χωρίς να μειώνει αποτελεσματικά – λόγω αμφιβολιών – την απειλή για την ιδιωτικότητα.

Ιχνηλάτηση
Αυτή η απειλή είναι μία ακόμη παρενέργεια της δυνατότητας συσχέτισης. Καθώς ένα άτομο συλλέγει αντικείμενα που φέρουν ετικέτες, δημιουργείται μία λίστα συσχετίσεων μεταξύ αυτών και της ταυτότητάς του. Όταν κάποια στιγμή το άτομο πετάξει κάποιο από αυτά τα αντικείμενα, το αποκαλούμενο ηλεκτρονικό ψίχουλο (electronic breadcrumb), ο συσχετισμός συνεχίζει να υφίσταται. Αυτή η ιδιότητα θα μπορούσε να χρησιμοποιηθεί κακόβουλα, π.χ. με την τοποθέτηση ενός αντικειμένου που έχει συσχετιστεί με ένα συγκεκριμένο πρόσωπο, τον αρχικό ιδιοκτήτη, στο χώρο ενός εγκλήματος, προκειμένου να δυσκολευτεί το έργο των αρχών. Το γεγονός και μόνο ότι μία συσχέτιση του παρελθόντος συνεχίζει να υφίσταται και στο μέλλον παρά τη θέληση μας, αποτελεί κατάφωρη παραβίαση της ιδιωτικότητας.

Αποκάλυψη Τοποθεσίας
Ο διαμοιρασμός κρυφών αναγνωστών σε συγκεκριμένες τοποθεσίες, δημιουργεί περισσότερες απειλές κατά τις ιδιωτικότητας. Ένα άτομο που φέρει αντικείμενα/ετικέτες με μοναδικούς κωδικούς μπορεί να παρακολουθηθεί μέσω αυτών και η τοποθεσία του να αποκαλυφθεί. Ακόμη και αν η συσχέτιση των ετικετών με το άτομο δεν είναι γνωστή, είναι δυνατό να παρακολουθήσουμε/καταγράψουμε το μονοπάτι που διέσχισε μία ετικέτα και συνδυάζοντας πληροφορίες από τρίτα συστήματα, όπως κλειστά κυκλώματα παρακολούθησης, να αποδώσουμε το μονοπάτι/ετικέτα σε ένα συγκεκριμένο άτομο.

Αποκάλυψη Προτιμήσεων
Όπως ήδη αναφέραμε, μία EPC ετικέτα ενός προϊόντος μπορεί να φέρει πληροφορία για τον κατασκευαστή του, το είδος του, αλλά και να το αναγνωρίσει μοναδικά. Διαβάζοντας αυτή την πληροφορία, κάποιος μπορεί να συμπεράνει τις προτιμήσεις του αγοραστή (π.χ. τι προϊόντα αγόρασε ή φέρει) ή και αποκτήσει πρόσβαση σε ιατρικές πληροφορίες (π.χ. ετικέτες φαρμακευτικών ειδών). Το γεγονός ότι η ανάγνωση των ετικετών μπορεί να γίνει, όπως είδαμε, με τρόπο μη αντιληπτό από τον αγοραστή – αντίθετα από τις κάρτες ανταμοιβής αγορών όπου υπάρχει συγκατάθεση – και με πολύ μικρό κόστος, δυσχεραίνει το πρόβλημα.
Μπορούμε εύκολα να φανταστούμε ένα ληστή, ο οποίος επιλέγει το επόμενο θύμα σύμφωνα με τις πληροφορίες που συλλέγει από τις ετικέτες που εκείνο φέρει, π.χ. το περιεχόμενο της τσάντας του, τα ρούχα του, ρολόι, ηλεκτρονικές συσκευές κ.λπ. Εξαιτίας των ετικετών RFID, η γνωστή προτροπή των αρχών να μην επιδεικνύουμε δημόσια αντικείμενα μεγάλης αξίας – προς αποφυγή δυσάρεστων συνεπειών – δε θα προσφέρει πλέον καμία προστασία.

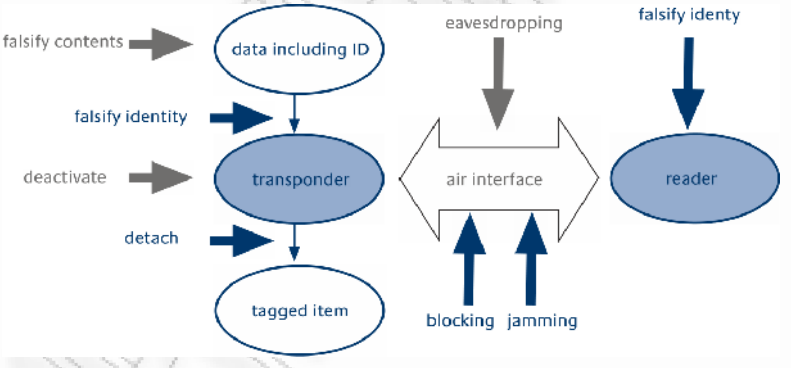
Αστερισμοί Ετικετών
Ακόμη και αν η ταυτότητα ενός ατόμου δεν έχει συσχετιστεί με τις ετικέτες που εκείνο φέρει (π.χ. οι ετικέτες των ρούχων του, των αγορών του κ.λπ.), το σύνολο των

ετικετών σχηματίζει ένα μοναδικό αστερισμό (constellation). Η μοναδικότητα αυτή μπορεί να χρησιμοποιηθεί για να παρακολουθήσουμε κάποιον και να καταγράψουμε τις κινήσεις του με μεγάλη ακρίβεια.

Αποκάλυψη Συναλλαγών
Όταν αντικείμενα, που φέρουν ετικέτες, μεταφέρονται από ένα σύνολο/αστερισμό ετικετών σε ένα άλλο, είναι εύκολο να υποθέσουμε ότι συνέβη μία ανταλλαγή μεταξύ των ατόμων που συσχετίζονται με τους αστερισμούς αυτούς.

1.2.2. Επιθέσεις

Το Γερμανικό Ομοσπονδιακό Γραφείο για την Ασφάλεια των Πληροφοριών BSI (Bundesamt für Sicherheit in der Informationstechnik) σε έκθεσή του για την ασφάλεια των συστημάτων RFID του 2005 [14] αναγνώρισε οχτώ βασικές επιθέσεις κατά της ασφάλειας συστημάτων RFID (Σχήμα 3). Παραδόξως, οι τέσσερις από αυτές, η απενεργοποίηση της ετικέτας (deactivate), η αποκόλληση της ετικέτας (detach), η παρεμπόδιση της επικοινωνίας (blocking) και η παρεμβολή του σήματος (jamming), μπορούν να αποτελέσουν τη βάση για τη δημιουργία αντιμέτρων προστασίας της ιδιωτικότητας και της ανωνυμίας.Τις υπόλοιπες τέσσερις επιθέσεις, οι επιθέσεις υποκλοπής (eavesdropping) και πλαστογράφησης της ταυτότητας του αναγνώστη (falsify identity) απειλούν άμεσα την ιδιωτικότητα και την ανωνυμία των χρηστών, ενώ οι επιθέσεις πλαστογράφησης των περιεχομένων (falsify contents) και πλαστογράφησης της ταυτότητας της ετικέτας μπορούν να οδηγήσουν σε έμμεση απώλεια της ιδιωτικότητας και της ανωνυμίας.



Σχήμα 3. *Βασικές επιθέσεις σε RFID συστήματα*
πηγή: BSI 2005

Παραποίηση περιεχομένων ετικέτας
Η παραποίηση είναι δυνατή όταν μία μη εξουσιοδοτημένη οντότητα αποκτά δικαιώματα εγγραφής σε μία ετικέτα. Συνήθως αυτού του είδους οι επιθέσεις είναι στοχευμένες. Μετά το πέρας της επίθεσης, οι αναγνωριστικές πληροφορίες της ετικέτας, όπως η ταυτότητά της και ότι άλλες πληροφορίες ασφάλειας μπορεί να υπάρχουν (π.χ. κλειδιά), παραμένουν ως έχουν, έτσι ώστε ο νόμιμος αναγνώστης να μπορέσει να αυθεντικοποίηση/αναγνωρίσει χωρίς πρόβλημα την ετικέτα. Αυτή η επίθεση έχει νόημα μόνο στην περίπτωση που η ετικέτα φέρει επιπλέον πληροφορία, πέραν του αναγνωριστικού της και των πληροφοριών ασφάλειας.

Πλαστογράφηση ταυτότητας ετικέτας

Ο επιτιθέμενος αποκτά την ταυτότητα και κάθε άλλη πληροφορία ασφάλειας της ετικέτας και τις χρησιμοποιεί προκειμένου να αυθεντικοποιηθεί στον αναγνώστη. Συνήθως ο επιτιθέμενος είτε διαθέτει κάποια συσκευή προσομοίωσης ετικετών, είτε μπορεί να παράγει μία νέα ετικέτα κλώνο της αρχικής (cloning).

Απενεργοποίηση ετικέτας
Αυτού του τύπου οι επιθέσεις καθιστούν μία ετικέτα άχρηστη μέσω της μη εξουσιοδοτημένης χρήσης εντολών διαγραφής (delete command) ή οριστικού τερματισμού (kill command) ή της φυσικής καταστροφής της. Ανάλογα με τον τρόπο απενεργοποίησης, ο αναγνώστης είτε αδυνατεί να εντοπίσει την ταυτότητα της ετικέτας είτε δεν μπορεί να εντοπίσει την ίδια την ύπαρξη της ετικέτας.

Αποκόλληση ετικέτας
Ανάλογα με την εφαρμογή, μία ετικέτα μπορεί να αποκολληθεί φυσικά από το αντικείμενο που ταυτοποιούσε και στη συνέχεια να επικολληθεί και να συσχετισθεί με ένα νέο αντικείμενο. Συνεπώς όλα τα συστήματα RFID που βασίζονται μόνο στις πληροφορίες που μεταδίδει η ετικέτα είναι ευάλωτα σε αυτή την επίθεση.
Στην πραγματικότητα, σε ένα σύστημα RFID ταυτοποιούμε την ετικέτα και όχι το αντικείμενο που τη φέρει και αυτό αποτελεί βασικό ελάττωμα του συστήματος. Έτσι, όπως η κατοχή ενός κλειδιού δε διασφαλίζει πάντοτε τη νόμιμη ιδιοκτησία, έτσι και μία ετικέτα δε διασφαλίζει την ύπαρξη του αντικείμενου, με το οποίο έχει συσχετιστεί.

Υποκλοπή ραδιοσημάτων
Η επικοινωνία ανάμεσα στην ετικέτα και τον αναγνώστη, μέσω της διεπαφής αέρα, μπορεί να υποκλαπεί από αναγνώστες οι οποίοι παρακολουθούν τα ραδιοσήματα που ανταλλάσσονται. Είναι μία από τις πιο στοιχειώδεις απειλές για τα συστήματα RFID.

Παρεμπόδιση επικοινωνίας
Η παρεμπόδιση επιτυγχάνεται με την κατάλληλη ρύθμιση της 'blocker tag', σε σχέση με το πρωτόκολλο αποφυγής σύγκρουσης που χρησιμοποιείται για την εύρυθμη επικοινωνία του αναγνώστη με τις ετικέτες.

Παρεμβολή σήματος
Η παρακώλυση του σήματος είναι πολύ εύκολη, λόγω της χαμηλής ισχύος – για λόγους προστασίας της δημόσιας υγείας – των RFID συσκευών. Ένας κακόβουλος χρήστης, εφοδιασμένος με μία απλή συσκευή παραγωγής θορύβου σε κατάλληλες συχνότητες, μπορεί να προκαλέσει το χάος σε ένα σύστημα RFID.

Πλαστογράφηση ταυτότητας αναγνώστη
Σε ένα ασφαλές σύστημα RFID πρέπει να υπάρχει αμοιβαία αυθεντικοποίηση (mutual authentication) μεταξύ του αναγνώστη και της ετικέτας, έτσι ώστε, εάν ένας επιτιθέμενος θέλει να επικοινωνήσει με μία ετικέτα θα πρέπει να υποδυθεί το νόμιμο αναγνώστη. Δυστυχώς, σε πολλά συστήματα, οι ετικέτες δεν αυθεντικοποιούν τον RFID αναγνώστη, καθώς για πολλές εφαρμογές, αυτή η διαδικασία είναι πολύ πολύπλοκη και ακριβή, αφού συνήθως απαιτεί ακριβές ετικέτες, με αυξημένες δυνατότητες επεξεργασίας (π.χ. υλοποίηση κυκλωμάτων εκτέλεσης βασικών κρυπτογραφικών συναρτήσεων) και διαχείριση κλειδιών.

1.3. Μέτρα προστασίας της Ιδιωτικότητας

Στην παρούσα ενότητα παρουσιάζουμε τεχνικές, οι οποίες έχουν προταθεί για την προστασία της ιδιωτικότητας σε εφαρμογές των ετικετών RFID. Στόχος όλων αυτών των μεθόδων είναι η αποτροπή αναγνώρισης μιας ετικέτας από μη εξουσιοδοτημένο αναγνώστη. Οι λύσεις που θα παρουσιάσουμε μπορούν να ταξινομηθούν σε τρεις κατηγορίες: τα φυσικά αντίμετρα, τα μη-κρυπτογραφικά αντίμετρα και τα κρυπτογραφικά αντίμετρα.

Η προστασία της ιδιωτικότητας με τη χρήση κρυπτογραφικών αλγορίθμων έχει, εν γένει, μελετηθεί εκτενώς τα τελευταία χρόνια και πολλές αξιόπιστες και ασφαλείς λύσεις έχουν προταθεί. Η υλοποίηση, όμως, όλων σχεδόν αυτών των κρυπτογραφικών πρωτοκόλλων θεωρείται πολύ δαπανηρή για τις ετικέτες RFID. Βασική επιδίωξη των κατασκευαστών RFID συστημάτων, είναι η παραγωγή ετικετών πολύ χαμηλού κόστους, ώστε να μπορούν να τοποθετούνται σε κάθε προϊόν χωρίς να επηρεάζεται σημαντικά η αξία του. Η προσπάθεια, όμως, αυτή διατήρησης του κόστους κατασκευής μιας ετικέτας στο ελάχιστο δυνατό επίπεδο, επιβάλλει πολλούς περιορισμούς σε ότι αφορά: στο χώρο που καταλαμβάνουν τα κυκλώματα του κρυπτογραφικού αλγορίθμου, στην ενέργεια που απαιτείται για τη λειτουργία των κυκλωμάτων αυτών και στο ποσοστό της συνολικής μνήμης της ετικέτας που ο αλγόριθμος χρειάζεται. Οι περιορισμοί αυτοί καθιστούν αδύνατη τη χρήση των γνωστών και καλά μελετημένων κρυπτογραφικών πρωτοκόλλων, ειδικά αυτών, τα οποία βασίζονται σε αλγορίθμους ασύμμετρης κρυπτογραφίας οι οποίοι θεωρούνται πολύ δαπανηροί στην υλοποίηση. Για την αποφυγή τους κόστους που συνεπάγεται η χρήση κρυπτογραφικών μεθόδων, έχουν προταθεί διάφορες λύσεις.

Στην κατηγορία των φυσικών αντιμέτρων ανήκουν οι λύσεις, οι οποίες σκοπό έχουν να προστατέψουν την ετικέτα χωρίς να την επιβαρύνουν, κυρίως διαχειριζόμενες εξωτερικά την ασύρματη επικοινωνία ανάμεσα στην ετικέτα και τον αναγνώστη, εκμεταλλευόμενες τις φυσικές ιδιότητες των σημάτων που εκπέμπονται. Οι μέθοδοι αυτές λύνουν μερικώς το πρόβλημα, όπως θα δούμε, ενώ πολλές φορές, επειδή βασίζονται σε μεθόδους άρνησης εξυπηρέτησης, μπορεί να επηρεάσουν τη λειτουργία και ετικετών, οι οποίες ανήκουν σε άλλο χρήστη.

Στην κατηγορία των μη-κρυπτογραφικών αντιμέτρων ανήκουν οι τεχνικές εκείνες, οι οποίες δεν κάνουν χρήση κρυπτογραφικών αλγορίθμων, αλλά μεταβάλλουν τον τρόπο λειτουργίας μιας ετικέτας. Σχεδόν όλες αυτές οι τεχνικές βασίζονται στην ύπαρξη ενός μυστικού συνθηματικού γνωστού μόνο στην ετικέτα και στον αναγνώστη. Η χρήση τέτοιων συνθηματικών παρουσιάζει, όμως, όλα τα προβλήματα εκείνα που χαρακτηρίζουν τα συστήματα διαχείρισης μυστικών κλειδιών.

Ο σχεδιασμός κρυπτογραφικών πρωτοκόλλων τα οποία είναι ασφαλή, αλλά ταυτόχρονα συμμορφώνονται με τους περιορισμούς που επιβάλλει το χαμηλό κόστος κατασκευής μιας ετικέτας, αποτελεί μία από τις πιο ενεργές ερευνητικές περιοχές. Οι περισσότερες λύσεις που έχουν προταθεί στηρίζονται, όπως είναι φυσικό, στους λιγότερο δαπανηρούς αλγορίθμους συμμετρικού κλειδιού, όπως οι συναρτήσεις σύνοψης και οι γεννήτριες ψευδοτυχαίων αριθμών. Η έρευνα για νέα πρωτόκολλα αναμένεται να εντατικοποιηθεί τα επόμενα χρόνια.

1.3.1. 'Φυσικά' αντίμετρα

Δυο αντίμετρα που ανήκουν στην κατηγορία των φυσικών αντιμέτρων αποτελούν ο κλωβός Faraday (Faraday Cage) και η μέθοδος των ενεργών παρεμβολών (active jamming). Και οι δυο αυτές λύσεις παρουσιάζουν τα ίδια μειονεκτήματα: δεν μπορούν να εφαρμοστούν σε μεγάλη κλίμακα και επιβαρύνουν το χρήστη για την εφαρμογή τους.

Κλωβός Faraday (Faraday Cage)
Η μέθοδος αυτή βασίζεται στο γεγονός ότι μεταλλικά αντικείμενα, αλλά και το νερό, εξασθενούν την ισχύ των ηλεκτρομαγνητικών κυμάτων που εκπέμπονται από μία ετικέτα RFID. Η τοποθέτηση, λοιπόν, της ετικέτας σε μία θήκη κατασκευασμένη από μεταλλικές ίνες, μπορεί να την προστατέψει από μη εξουσιοδοτημένη ανάγνωση. Η αποτελεσματικότητα της θήκης εξαρτάται από τη συχνότητα που χρησιμοποιείται. Για παράδειγμα, ενώ μία θήκη μπορεί να εμποδίζει τη διάδοση σημάτων από UHF ετικέτες, οι HF ετικέτες να λειτουργούν κανονικά στην ίδια θήκη.
Η προστασία που μπορεί να προσφέρει η χρήση του κλωβού είναι περιορισμένη. Η μέθοδος αυτή είναι πρακτικά αδύνατο να εφαρμοστεί σε μεγάλο αριθμό αντικειμένων που φέρουν ετικέτα RFID. Επίσης, αντικείμενα τα οποία έχουν πολύ μεγάλο μέγεθος για να χωρέσουν σε μία θήκη επενδυμένη με μεταλλικό πλέγμα παραμένουν απροστάτευτα. Επιπλέον, η χρήση μιας προστατευτικής θήκης (π.χ. μιας κατάλληλα επενδυμένης τσέπης παλτού), μπορεί να προστατέψει την ιδιωτικότητα του κατόχου ενός αντικειμένου που φέρει RFID ετικέτα, αλλά δεν προσφέρει καμία προστασία από τη στιγμή που ο κάτοχος του αντικειμένου αποφασίσει να χρησιμοποιήσει το αντικείμενο αυτό και επομένως το αφαιρέσει από την προστατευτική του θήκη.

Μέθοδος των Ενεργών Παρεμβολών (Active Jamming)
Μια λύση εναλλακτική του κλωβού Faraday αποτελεί η χρήση συσκευής εκπομπής ραδιοσημάτων θορύβου, τα οποία παρεμποδίζουν τη λειτουργία όλων των αναγνωστών οι οποίοι βρίσκονται κοντά στην ετικέτα. Η χρησιμοποίηση, όμως, μιας τέτοιας συσκευής θα μπορούσε να προκαλέσει προβλήματα και σε νόμιμα συστήματα, αλλά και στην αναγνώριση ετικετών άλλων χρηστών. Πιθανότατα η χρήση της θα απαγορευόταν δια νόμου, τουλάχιστον για εκπομπή σημάτων μεγάλης ισχύος.
Αξίζει να σημειωθεί ότι η χρήση μιας συσκευής παρεμβολών μπορεί να εγείρει από μόνη της νέες απειλές ενάντια στην ιδιωτικότητα του χρήστη της και ίσως μάλιστα μεγαλύτερες από αυτές της ετικέτας RIFD που θέλει να προστατεύσει. Σε περίπτωση που η συσκευή δεν είναι σωστά σχεδιασμένη, το σήμα παρεμβολής το οποίο εκπέμπει, μπορεί να χρησιμοποιηθεί για την αναγνώριση του χρήστη της συσκευής και μάλιστα από απόσταση πολύ μεγαλύτερη από την απαιτούμενη για την ετικέτα RFID, δεδομένου ότι η ισχύς του σήματος της συσκευής είναι πολύ μεγαλύτερη.

1.3.2. Μη κρυπτογραφικά αντίμετρα

Απενεργοποίηση ετικέτας (Tag Deactivation): οι εντολές 'τερματισμός' και 'ύπνωση'
Οι προδιαγραφές του ECPglobal προσφέρουν δυο εντολές, τις οποίες μπορεί να εκτελέσει η ετικέτα RIFD, για την προστασία του καταναλωτή. Τις εντολές 'τερματισμός' (kill command) και 'ύπνωση' (sleep command).
Η χρήση της εντολής 'τερματισμός' αποτελεί δραστική λύση και προσφέρει μόνιμη προστασία της ιδιωτικότητας. Πιο συγκεκριμένα, η εντολή αυτή καθιστά την ετικέτα μόνιμα ανενεργή, ενώ δεν υπάρχει εντολή επανενεργοποίησής της. Για τις EPC Class 1 Gen 1 ετικέτες χρησιμοποιείται ένα συνθηματικό των 8-bits, το οποίο ορίζεται κατά

την παραγωγή της ετικέτας. Το συνθηματικό προστατεύει την ετικέτα από μη εξουσιοδοτημένη απενεργοποίηση. Για τις ετικέτες EPC Class 1 Gen 2, το συνθηματικό έχει μήκος 32 bits, ενώ και για τις δυο γενιές ετικετών, συνεχόμενες αποτυχημένες εισαγωγές συνθηματικού οδηγεί σε κλείδωμα της ετικέτας. Έτσι, η εντολή μπορεί να εκτελείται στο κατάστημα κατά την αγορά του προϊόντος που φέρει την ετικέτα. Αξίζει να σημειωθεί ότι, η υλοποίηση της εντολής σε λογισμικό, συνήθως περιορίζεται στην απενεργοποίηση μόνο της ασύρματης διασύνδεσης της ετικέτας. Έχουν προταθεί επιθέσεις, οι οποίες επανενεργοποιούν την ετικέτα με άμεση επαφή με αυτή, χωρίς τη χρήση της ασύρματης διασύνδεσης.
Μία πιο ήπια αντιμετώπιση του προβλήματος της ιδιωτικότητας προσφέρει η εντολή 'ύπνωση', η οποία θέτει σε κατάσταση αναστολής λειτουργίας την ετικέτα. Πιο συγκεκριμένα, η ετικέτα δεν αποκρίνεται σε εντολές του αναγνώστη και διατηρεί αυτή την κατάσταση μέχρι να λάβει εντολή 'αφύπνιση' (wake command). Και οι δύο εντολές, 'ύπνωση' και 'αφύπνιση', προστατεύονται με τη χρήση συνθηματικού, όμοια με την εντολή 'τερματισμός'. Φυσικά, το συνθηματικό είναι μοναδικό για κάθε ετικέτα RFID και όπως κάθε χρήση μυστικού κλειδιού κληροδοτεί στο σύστημα όλες τις δυσκολίες που έχει η διαχείριση τέτοιων κλειδιών.
Και οι δύο εντολές, 'ύπνωση' και 'τερματισμός', αποτελούν εύκολα υλοποιήσιμες λύσεις και είναι συμβατές με τη σχετική νομοθεσία η οποία επιβάλλει την παροχή δυνατότητας απενεργοποίησης της ετικέτας σε όποιον δεν επιθυμεί τη χρήση της. Οι λύσεις αυτές, όμως, έχουν πολλά και σημαντικά μειονεκτήματα. Αρχικά, ενώ προσφέρουν ασφάλεια στον τελικό χρήστη ενός προϊόντος, δεν εξασφαλίζουν την ιδιωτικότητα στη γραμμή παραγωγής και διάθεσης του προϊόντος. Από την άλλη πλευρά, η υλοποίηση αυτών των εντολών ως προαιρετικών υπηρεσιών, τις οποίες μπορεί να χρησιμοποιήσει κατά βούληση ο χρήστης, επιφέρει μεγάλη επιβάρυνση για τα μικρά καταστήματα λόγου της ανάγκης χρήσης ακριβού εξοπλισμού διαχείρισης των εντολών στα σημεία πώλησης. Επιπλέον, η απενεργοποίηση των ετικετών στερεί από τους χρήστες όλες εκείνες τις υπηρεσίες, οι οποίες εκμεταλλεύονται την ύπαρξη ετικετών RFID σε ένα προϊόν, όπως η προστασία από κλοπή, η χρήση των πληροφοριών της ετικέτας σε συνδυασμό με 'έξυπνες συσκευές' (π.χ. φούρνοι μικροκυμάτων, πλυντήριο ρούχων κ.λπ.), η ανακύκλωση προϊόντων, η επιστροφή ενός προϊόντος στο κατάστημα αγοράς του χωρίς τη χρήση απόδειξης αγοράς κ.λπ. Αξίζει να τονίσουμε για άλλη μία φορά, ότι η χρήση συνθηματικών για την προστασία των εντολών συνεπάγεται όλα τα προβλήματα που έχουν τα συστήματα διαχείρισης μυστικών κλειδιών.

Η μέθοδος 'Blocker tag'
Η μέθοδος αυτή προστασίας της ιδιωτικότητας είναι μία τεχνική άρνησης εξυπηρέτησης (denial-of-service) η οποία βασίζεται στο πρωτόκολλο απομόνωσης του αλγορίθμου Διάσχισης Δυαδικού Δένδρου που χρησιμοποιείται από τους RFID αναγνώστες για την επικοινωνία τους με τις ετικέτες. Οι εμπνευστές της μεθόδου ισχυρίζονται ότι μπορεί εξίσου να εφαρμοστεί όταν χρησιμοποιείται το πρωτόκολλο αποτροπής συγκρούσεων ALOHA.
Η μέθοδος 'blocker tag', για να επιτύχει την προστασία της ιδιωτικότητας, εκμεταλλεύεται το γεγονός ότι, κατά τη διάσχιση του δένδρου, οι ετικέτες οι οποίες έχουν το ίδιο πρόθεμα αναγνωριστικού βρίσκονται στο ίδιο υποδέντρο/ζώνη και ανιχνεύονται από τον αναγνώστη σειριακά. Μία ετικέτα 'blocker' είναι απλώς μία ετικέτα, η οποία μπορεί να υποδυθεί μία άλλη ετικέτα. Συγκεκριμένα, όταν ο αναγνώστης στέλνει αίτηση στις ετικέτες σε ένα υποδέντρο ενός κόμβου Β ζητώντας το επόμενο bit , η ετικέτα 'blocker' εκπέμπει ταυτόχρονα και το '0' και το '1' bit.

Αυτή η τεχνητή σύγκρουση που δημιουργείται υποχρεώνει τον αναγνώστη να επιστρέψει σε όλους του κόμβους. Με δεδομένο ότι ο αναγνώστης έχει αρκετή υπολογιστική ισχύ, μπορεί να εξάγει όλους τους πιθανούς σειριακούς αριθμούς από το δέντρο.
Υπάρχουν δυο παραλλαγές του σχήματος αυτού. Στη μία παραλλαγή η ετικέτα υποδύεται όλες τις πιθανές ετικέτες, ενώ στη δεύτερη μόνο τις ετικέτες που ανήκουν σε ένα συγκεκριμένο υποδέντρο/ζώνη.
Επειδή ο αλγόριθμος διάσχισης δένδρου ανιχνεύει τα αναγνωριστικά IDs των ετικετών σειριακά, η ετικέτα 'blocker', η οποία προστατεύει ένα συγκεκριμένο υποδέντρο, μπορεί να εμποδίσει έναν αναγνώστη από την απόκτηση πρόσβασης σε κάποιους κόμβους, ακόμη και αν αυτοί δεν ανήκουν στο υποδέντρο που προστατεύει. Για την επίλυση αυτού του προβλήματος προτείνεται η χρήση ετικετών, οι οποίες ενημερώνουν τον αναγνώστη ότι ένα υποδέντρο είναι κάτω από την προστασία τους, ώστε ο αναγνώστης να αγνοήσει αυτό το μέρος του δέντρου και να συνεχίσει την αναζήτησή του.
Επιπρόσθετα, υπάρχουν περιπτώσεις όπου ένας χρήστης επιθυμεί να προστατέψει μερικές, μόνον, από τις RFID ετικέτες του και όχι το σύνολο τους. Για το λόγο αυτό προτάθηκε ο ορισμός 'ζωνών ιδιωτικότητας' ('privacy zones'), στις οποίες ο χρήστης μπορεί να προσθαφαιρεί ετικέτες αλλάζοντας το σειριακό τους αριθμό (π.χ. αναστρέφοντας συγκεκριμένα bits του σειριακού αριθμού). Μία τέτοια ζώνη μπορεί να οριστεί με τον καθορισμό ενός 'bit ιδιωτικότητας' ('privacy bit'), όπου, ανάλογα με την τιμή του, ορίζεται αν μία ετικέτα είναι μέσα ή έξω από μία ζώνη. Η ιδέα μπορεί να επεκταθεί στον ορισμό πολλαπλών ανεξαρτήτων ζωνών, για την παροχή πιο σύνθετων πολιτικών ιδιωτικότητας. Αξίζει να σημειωθεί ότι η χρήση της μεθόδου αυτής είναι προαιρετική, δηλαδή οι χρήστες επιφορτίζονται με τη λήψη απόφασης αν θέλουν να προστατέψουν την ιδιωτικότητα τους, κουβαλώντας μαζί τους μία ετικέτα 'blocker'.
Η μέθοδος αυτή παρουσιάζει δύο βασικά μειονεκτήματα. Από τη μία η σωστή λειτουργία της επαφίεται στο χρήστη, ο οποίος συνήθως δεν είναι σε θέση να τη εφαρμόσει σωστά, αποκτώντας με τον τρόπο αυτό μόνο μία ψευδαίσθηση ασφάλειας και προστασίας. Από την άλλη, υπάρχει η πιθανότητα παρεμβολής στην επικοινωνία ετικετών οι οποίες ανήκουν σε άλλους χρήστες, εμποδίζοντας τη σωστή λειτουργία τους.

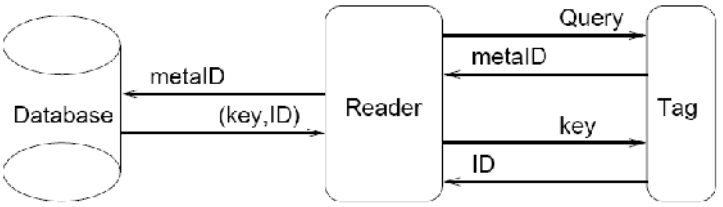
Ετικέτες με δυνατότητα ανάλυσης της ενέργειας της κεραίας
Στη μέθοδο αυτή, η ετικέτα προσπαθεί να εντοπίσει αν ο αναγνώστης είναι νόμιμος, μετρώντας την ποιότητα του σήματος που λαμβάνει. Το αντίμετρο αυτό βασίζεται στο γεγονός ότι ο επιτιθέμενος, χρησιμοποιεί συνήθως αναγνώστη ο οποίος βρίσκεται σε αρκετή απόσταση από την ετικέτα RFID. Δεδομένου ότι η ποιότητα του σήματος ελαττώνεται με την απόσταση, η ετικέτα μπορεί να μετρά το λόγο σήματος προς θόρυβο, αυξάνει όσο πιο κοντά είναι ο αναγνώστης στην ετικέτα και ανάλογα με την τιμή του να καθορίζει την αντίδραση της. Η μέθοδος αυτή, από μόνη της, δεν προσφέρει ασφάλεια, αλλά είναι πολύ χρήσιμη όταν εφαρμόζεται σα συμπλήρωμα άλλων αντιμέτρων.

Αντιμετώπιση Υποκλοπής Σημάτων του RFID αναγνώστη
Δεδομένου ότι ο πομπός του αναγνώστη είναι πιο ισχυρός από τον πομπό της ετικέτας RIFD, έχει την ικανότητα να εκπέμπει σήματα σε μεγαλύτερες αποστάσεις. Είναι, λοιπόν, πιο εύκολο για τον επιτιθέμενο να υποκλέπτει αυτά τα σήματα, παρά τα σήματα που εκπέμπει μία ετικέτα. Η ιδιαιτερότητα αυτή του συστήματος

αναγνώστη/ετικέτας, σε συνδυασμό με το γεγονός ότι όταν ένα σύστημα χρησιμοποιεί το βασικό πρωτόκολλο απομόνωσης του αλγορίθμου διάσχισης δένδρου, ο επιτιθέμενος μπορεί να συνάγει το αναγνωριστικό ID της ετικέτας, απλώς υποκλέπτοντας τα σήματα που αποστέλλει ο αναγνώστης, καθιστά το σύστημα τρωτό σε επιθέσεις παραβίασης της ιδιωτικότητας.
Για την αντιμετώπιση αυτού του προβλήματος έχει προταθεί η χρήση ετικετών με δυναμική μνήμη, οι οποίες είναι σε θέση να αποθηκεύουν την τρέχουσα θέση του bit που ζητήθηκε από τον αναγνώστη. Έτσι, σε αντίθεση με το πρωτόκολλο διάσχισης δένδρου, οι αναγνώστες δε χρειάζεται να αποστέλλουν όλο το πρόθεμα (prefix) στις ετικέτες, αλλά μία απλή εντολή 'απόστειλε το επόμενο bit' η οποία είναι αρκετή για να διατρέξει ο αναγνώστης όλο το δυαδικό δέντρο. Πιο συγκεκριμένα, όπως και με τη βασική έκδοση του αλγόριθμου διάσχισης δένδρου, όσο δυο ή περισσότερες ετικέτες μοιράζονται το ίδιο πρόθεμα, δεν υπάρχει σύγκρουση και ο αναγνώστης συνεχίζει την αναζήτηση στο μονοπάτι. Σε περίπτωση σύγκρουσης, δηλαδή όταν το αναγνωριστικό δυο ετικετών διαφέρει στη θέση i, ο αναγνώστης με την εντολή "select" επιλέγει υποδέντρο. Αντί όμως να στείλει στις ετικέτες όλο το κοινό πρόθεμα μαζί με το bit της θέσης i, υπολογίζει το XOR του τελευταίου κοινού bit (στη θέση $i-1$) με το επιλεγμένο τελευταίο (ανάλογα με το ποιο υποδέντρο επιλέγει) και στέλνει το αποτέλεσμα. Όλες οι ετικέτες υπολογίζουν την τιμή XOR του bit που λάβανε με την τιμή του δικού τους bit στη θέση $i-1$, και συγκρίνουν την τιμή που προκύπτει με την τιμή του δικού τους bit στη θέση i. Αν ταιριάζει, η ετικέτα έχει επιλεγεί και αποκρίνεται με το επόμενο bit, δηλαδή αυτό της θέσης $i+1$. Αν ο επιτιθέμενος μπορεί να υποκλέψει μόνο τα μηνύματα του αναγνώστη, πλέον δεν είναι σε θέση, όπως συνέβαινε με το βασικό αλγόριθμο διάσχισης δένδρου, να παρατηρήσει τα bits των δίχως σύγκρουση προθεμάτων (collision-free prefixes), αφού ο αναγνώστης στέλνει μόνο την εντολή 'στείλε επόμενο bit'. Σε περίπτωση σύγκρουσης, δεν μπορεί να υπολογίσει κανένα από τα bits που στέλνονται, αφού η τιμή του bit στη θέση $i-1$ (το οποίο είναι άγνωστο στον επιτιθέμενο) εξασφαλίζει μέσω της πράξης XOR την εμπιστευτικότητα του επιλεγμένου από τον αναγνώστη subtree-bit.
Πρέπει όπως να τονιστεί ότι η προτεινόμενη λύση κάνει χρήση δυναμικής μνήμης, η οποία αυξάνει σημαντικά το κόστος της ετικέτας. Επίσης, δεν προσφέρει καμία προστασία από ενεργές επιθέσεις, όπου ο επιτιθέμενος μπορεί να μεταβάλλει τα μηνύματα που ανταλλάσσονται μεταξύ αναγνώστη και ετικέτας. Απλώς διορθώνει, εν μέρει, το πρόβλημα της υποκλοπής των μηνυμάτων του αναγνώστη.

1.3.3. Κρυπτογραφικά αντίμετρα

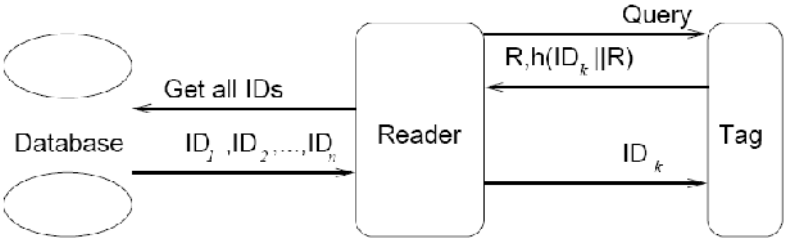
Σε αυτή την κατηγορία εντάσσονται τα αντίμετρα τα οποία βασίζονται στη χρήση κρυπτογραφικών αλγορίθμων [15]. Επιδίωξη των κατασκευαστών ετικετών RFID είναι να διατηρήσουν το κόστος τους όσο πιο χαμηλό γίνεται, με αποτέλεσμα μόνο πρωτόκολλα, τα οποία βασίζονται σε απλές πράξεις και σε μερικούς συμμετρικούς αλγορίθμους όπως οι συναρτήσεις σύνοψης (hash functions), να είναι αποδεκτά.



Σχήμα 4. *Hash Locking: Ο αναγνώστης ξεκλειδώνει την ετικέτα*

Hash Lock

Η μέθοδος αυτή προστατεύει τα δεδομένα που είναι αποθηκευμένα στην ετικέτα περιορίζοντας την πρόσβαση σε αυτά ([15]). Αυτό επιτυγχάνεται ως ακολούθως: κάθε ετικέτα RFID είναι εφοδιασμένη με μία συνάρτηση σύνοψης $H(x)$. Αρχικά, ο RFID αναγνώστης επιλέγει ένα τυχαίο κλειδί k, για το οποίο υπολογίζει την τιμή της συνάρτησης $h = H(k)$. Η τιμή h είναι γνωστή και ως metaID. Στη συνέχεια, αποθηκεύει την τιμή metaID σε ειδική περιοχή της ετικέτας, αλλά και σε μία βάση δεδομένων που διατηρεί στο back-end σύστημα. Μετά την αποθήκευση της τιμής metaID, η ετικέτα αποκρίνεται μόνο σε αιτήσεις, οι οποίες ζητούν το αναγνωριστικό της metaID και δεν επιτρέπει καμία άλλη λειτουργία. Λέμε ότι η ετικέτα είναι 'κλειδωμένη'.



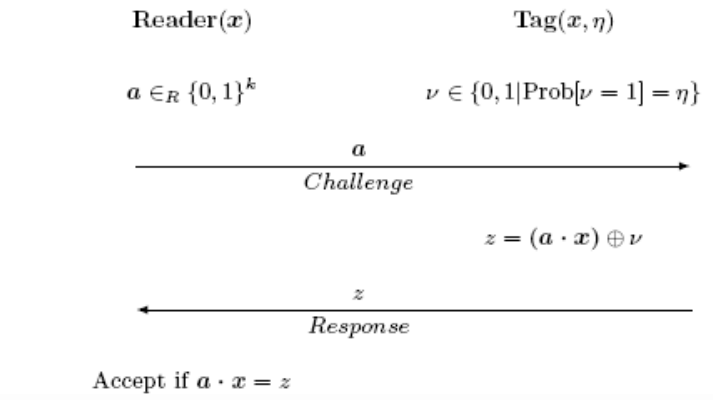
Σχήμα 5. *Randomized Hash Locking: Ο αναγνώστης ξεκλειδώνει μία ετικέτα αφού έχει βρει το αναγνωριστικό της είναι* ID_k**.**

Ο αναγνώστης που θέλει να χρησιμοποιήσει μία ετικέτα, ζητάει την τιμή metaID από την ετικέτα αυτή. Χρησιμοποιώντας την τιμή metaID, είναι σε θέση να βρει, από το ευρετήριο που διατηρεί στο back-end σύστημα, την τιμή του κλειδιού k το οποίο χρησιμοποιήθηκε για την παραγωγή του metaID. Αποστέλλει το k στην ετικέτα, η οποία επιβεβαιώνει ότι είναι το σωστό κλειδί υπολογίζοντας την τιμή της συνάρτησης $h = H(k)$ και επαληθεύοντας ότι ισούται με την αποθηκευμένη τιμή metaID. Όταν οι δυο τιμές ταιριάζουν, η ετικέτα διαγράφει την τιμή metaID και αποκρίνεται σε όλες τις αιτήσεις του αναγνώστη. Η ετικέτα είναι 'ξεκλείδωτη'.

Ενώ, με τη μέθοδο αυτή, η ετικέτα δεν επιτρέπει τη μη εξουσιοδοτημένη πρόσβαση στα δεδομένα που φέρει, η ίδια παραμένει απροστάτευτη σε επιθέσεις κατά της ιδιωτικότητας. Πιο συγκεκριμένα, η ετικέτα αποκρίνεται σε κάθε αίτημα αναγνώστη με την αποστολή της ίδιας τιμής αναγνωριστικού metaID, επιτρέποντας με τον τρόπο αυτό τον εντοπισμό της. Το μειονέκτημα αυτό μπορεί να διορθωθεί με τη χρήση "randomized hash-locks". Σε αυτή την παραλλαγή του πρωτοκόλλου υποθέτουμε ότι

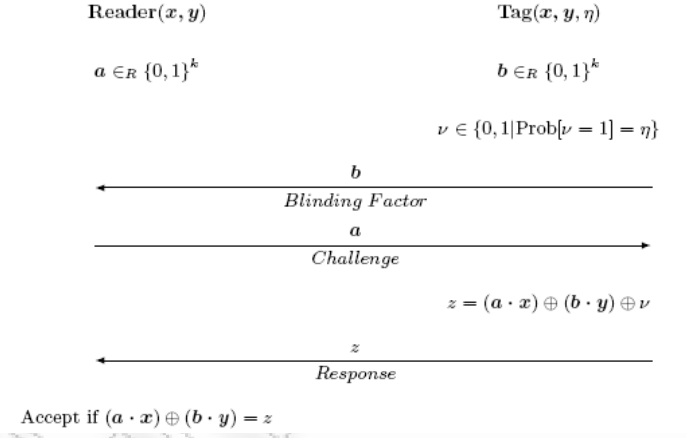
η ετικέτα διαθέτει και μία γεννήτρια ψευδοτυχαίων αριθμών. Στην περίπτωση αυτή, για το κλείδωμα της ετικέτας δεν απαιτείται κάποιο κλειδί, αλλά πραγματοποιείται με μία απλή εντολή κλειδώματος. Για το ξεκλείδωμα της ετικέτας, ο αναγνώστης στέλνει μία αίτηση στην οποία η ετικέτα αποκρίνεται με την αποστολή ενός τυχαίου αριθμού N και της εξόδου της συνάρτησης σύνοψης, με είσοδο την αλληλουχία του τυχαίου αριθμού N και του αναγνωριστικού της ID, το οποίο δεν αλλάζει στο χρόνο ζωής της ετικέτας. Δηλαδή στέλνει το μήνυμα ($N, H(N \| ID)$). Δεδομένου ότι ο νόμιμος αναγνώστης γνωρίζει όλα τα αναγνωριστικά ID_i των ετικετών που βρίσκονται στη δικαιοδοσία του, υπολογίζει για καθένα από αυτά την τιμή $h_i = H(N \| ID_i)$. Το αναγνωριστικό της ετικέτας είναι αυτό του οποίου η τιμή h_i είναι ίδια με την τιμή $H(N \| ID)$ που έλαβε ο αναγνώστης. Αφού έχει βρει τη σωστή τιμή ID, μπορεί να το χρησιμοποιήσει είτε για να ξεκλειδώσει την ετικέτα είτε για οποιαδήποτε άλλη λειτουργία, όπως για παράδειγμα, για την καταγραφή της παρουσίας ενός προϊόντος. αφήνοντας την ετικέτα κλειδωμένη.
Η ασφάλεια του πρωτοκόλλου βασίζεται στην ασφάλεια της συνάρτησης σύνοψης, η οποία θα πρέπει να φέρεται ως μονόδρομη συνάρτηση. Δηλαδή θα πρέπει να είναι αδύνατο για τον επιτιθέμενο να μπορεί να υπολογίσει την είσοδο, ή μέρος της εισόδου x της συνάρτησης, όταν η έξοδος της $h = H(x)$ του είναι γνωστή.
Φυσικά η επιβάρυνση για τον RIFD αναγνώστη είναι μεγάλη και αυξάνει με το πλήθος των ετικετών που βρίσκονται μέσα στη δικαιοδοσία του, καθώς θα πρέπει να υπολογίζει τη συνάρτηση $h_i = H(N \| ID_i)$ για καθεμία από αυτές κάθε φορά που θέλει να ξεκλειδώσει μία ετικέτα. Πρέπει να τονιστεί ότι σε πολλές εφαρμογές, το κόστος που συνεπάγεται η υλοποίηση μιας συνάρτησης σύνοψης σε μία ετικέτα, μπορεί να είναι αποτρεπτικό για τη χρησιμοποίηση της συνάρτησης. Είναι κατανοητό ότι πρέπει να αφήνουμε την ετικέτα ξεκλείδωτη για όσο το δυνατό μικρότερο χρονικό διάστημα.

Η οικογένεια πρωτοκόλλων HB
Η οικογένεια πρωτοκόλλων HB αποτελεί ίσως το πιο χαρακτηριστικό παράδειγμα χαμηλού κόστους κρυπτογραφικού πρωτοκόλλου αυθεντικοποίησης. Όλα τα πρωτόκολλα που ανήκουν σε αυτή την οικογένεια είναι παραλλαγές του HB πρωτοκόλλου, το οποίο είχε αρχικά προταθεί για την αυθεντικοποίηση ανθρώπου από μηχανή.



Σχήμα 6. *Ένας γύρος του πρωτοκόλλου HB.*

Λόγο των ομοιοτήτων ανάμεσα στον ανθρώπινο εγκέφαλο και την ετικέτα RFID όσον αφορά τις περιορισμένες υπολογιστικές δυνατότητες αλλά και την περιορισμένη διαθέσιμη μνήμη, η προσαρμογή πρωτοκόλλων που έχουν σχεδιαστεί για να εκτελούνται από έναν άνθρωπο αποτελεί λογική αφετηρία για τη σχεδίαση πρωτοκόλλων για RFID ετικέτες. Το HB πρωτόκολλο αποτελείται από N επαναλήψεις ενός βασικού βήματος ανταλλαγής δυο μηνυμάτων ανάμεσα στον αναγνώστη και την ετικέτα, για τους οποίους υποθέτουμε ότι μοιράζονται ένα κοινό κλειδί x. Ο αναγνώστης αρχικά στέλνει ένα τυχαίο μήνυμα a, στο οποίο η ετικέτα αποκρίνεται στέλνοντας το bit $u = a \cdot x + v$, όπου το $v = 0$ με πιθανότητα q και $a \cdot x$ είναι το εσωτερικό γινόμενο των δυαδικών διανυσμάτων a και x. Ο αναγνώστης υπολογίζει την τιμή $a \cdot x$ και αν $a \cdot x = u$ για περίπου qN από τις επαναλήψεις του βασικού βήματος, τότε δέχεται ότι η ετικέτα είναι αυθεντική. Δεδομένης της τυχαιότητας που εισάγει η μεταβλητή v, ένας επιτιθέμενος δεν είναι σε θέση από τις αποκρίσεις της ετικέτας να αποφανθεί αν ακούει την ίδια ετικέτα ή κάποια άλλη, ακόμη και για το ίδιο μήνυμα a, προστατεύοντας την ιδιωτικότητα.
Το πρωτόκολλο HB, στην αρχική του μορφή, δεν μπορεί να εφαρμοστεί για την αυθεντικοποίηση ετικέτας από αναγνώστη, γιατί παρουσιάζει αδυναμίες σε ενεργές επιθέσεις, δηλαδή σε επιθέσεις όπου ο επιτιθέμενος μπορεί να επιλέξει το μήνυμα a. Για την αντιμετώπιση ενεργών επιθέσεων έχει επιλεγεί το πρωτόκολλο HB+. Το HB+ είναι μία γενίκευση του HB. Το κύριο χαρακτηριστικό του πρωτοκόλλου αυτού είναι ότι συνοδεύεται από μαθηματική απόδειξη της ασφάλειας που προσφέρει.



Σχήμα 7. *Ένας γύρος του πρωτοκόλλου HB+.*

Συγκεκριμένα, είναι δυνατόν να δείξει κανείς ότι το HB+ είναι ασφαλές από ενεργές επιθέσεις, δεδομένης της υπολογιστικής δυσκολίας επίλυσης του LPN (learning parity with noise) προβλήματος. Όμως, το HB+ παρουσιάζει αδυναμία σε επιθέσεις man-in-the-middle, όπου ο επιτιθέμενος παρεμβαίνει σε μία εκτέλεση του πρωτοκόλλου από αυθεντικό αναγνώστη και ετικέτα, γεγονός που έχει οδηγήσει σε νέες βελτιωμένες παραλλαγές του.

1.3.4. Μελλοντικές κατευθύνσεις

Η ανάγκη εύρεσης ασφαλών λύσεων, ελάχιστου κόστους, για την προστασία της ιδιωτικότητας είναι επιτακτική. Οι μέχρι σήμερα προτεινόμενες μέθοδοι δύνανται μόνο μερικώς να αντιμετωπίσουν το πρόβλημα και όλες τους παρουσιάζουν αδυναμίες.

Ίσως η περισσότερα υποσχόμενη, αλλά και λογική ερευνητική κατεύθυνση, είναι η μελέτη και δημιουργία κρυπτογραφικών πρωτοκόλλων χαμηλού κόστους. Η κρυπτογραφική κοινότητα διαθέτει την απαραίτητα εμπειρία και αναμένεται ότι σύντομα να παρουσιαστούν λύσεις οι οποίες θα παρέχουν ικανοποιητική ασφάλεια, δεδομένων πάντα των περιορισμών που υπάρχουν. Η μελέτη και περαιτέρω ανάλυση πρωτοκόλλων, όπως τα πρωτόκολλα της οικογένειας HB, αποτελεί μία τέτοια επιλογή. Δηλαδή ο σχεδιασμός πρωτοκόλλων, τα οποία να στηρίζονται σε απλές πράξεις πολλαπλασιασμού και πρόσθεσης δυαδικών ψηφίων και τα οποία να βασίζουν την ασφάλεια τους σε γνωστά και καλά μελετημένα προβλήματα της θεωρίας κωδίκων διόρθωσης σφάλματος (error correcting codes).
Επιπλέον όμως των κρυπτογραφικών πρωτοκόλλων, υπάρχουν και μέθοδοι οι οποίες από μόνες τους, ή συνεπικουρούμενες από κρυπτογραφικούς αλγορίθμους μπορούν να αποτελέσουν λύσεις για την προστασία της ιδιωτικότητας. Ένα τέτοιο παράδειγμα είναι η χρήση του κινητού τηλεφώνου, δηλαδή μιας συσκευής την οποία οι περισσότεροι καταναλωτές διαθέτουν, ως εξυπηρετητή (server) για την επικοινωνία αναγνώστη ετικέτας. Οι ετικέτες, δηλαδή, εκμεταλλεύονται τις αυξημένες υπολογιστικές δυνατότητες που προσφέρει το κινητό τηλέφωνο για να υποστηρίξουν πιο εξελιγμένες μεθόδους κρυπτογράφησης.
Δυο παρατηρήσεις είναι απαραίτητες. Αρχικά πρέπει να τονιστεί ότι, ανεξάρτητα από τη μέθοδο προστασίας που θα επιλεγεί, ένα από τα μεγαλύτερα προβλήματα τα οποία καλούμαστε να αντιμετωπίσουμε, είναι αυτό της διαχείρισης των μυστικών κλειδιών και συνθηματικών που χρησιμοποιούνται. Επιπλέον, μέχρι τώρα θεωρούμε ότι το υπολογιστικό κόστος, στο οποίο καλείται να αντεπεξέλθει ο αναγνώστης, είναι αποδεκτό όσο μεγάλο και αν είναι. Στην πράξη, όμως, ειδικά στην περίπτωση που ο αναγνώστης είναι σε θέση να αναγνώσει πολλές ετικέτες, το θέμα των απαιτήσεων σε υπολογιστική ισχύ μπορεί να αποτελέσει τροχοπέδη στην εφαρμογή μιας μεθόδου.
Είναι, πάντως, γεγονός ότι οι τεχνολογικές λύσεις από μόνες τους δεν επαρκούν για την αντιμετώπιση του προβλήματος προστασίας της ιδιωτικότητας που προκύπτει από τη χρήση ετικετών RFID. Είναι απαραίτητο να συμπληρώνονται από ρυθμίσεις, κανονισμούς και νόμους οι οποίοι διασφαλίζουν τα δικαιώματα των καταναλωτών.

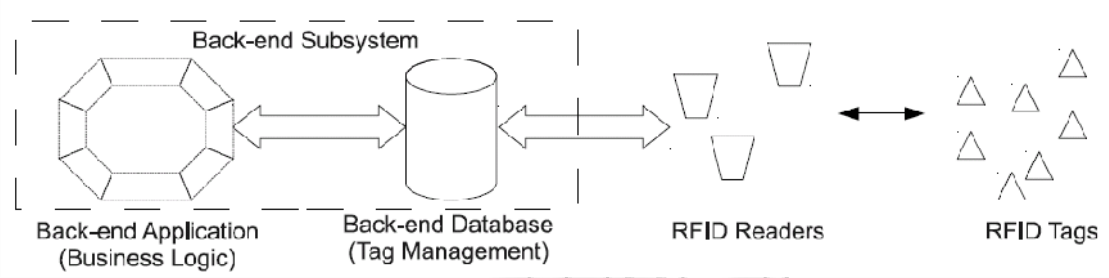
2. Ένα σύστημα διαχείρισης RFID ετικετών βασισμένο σε πράκτορες λογισμικού (software agents)

2.1. Εισαγωγή

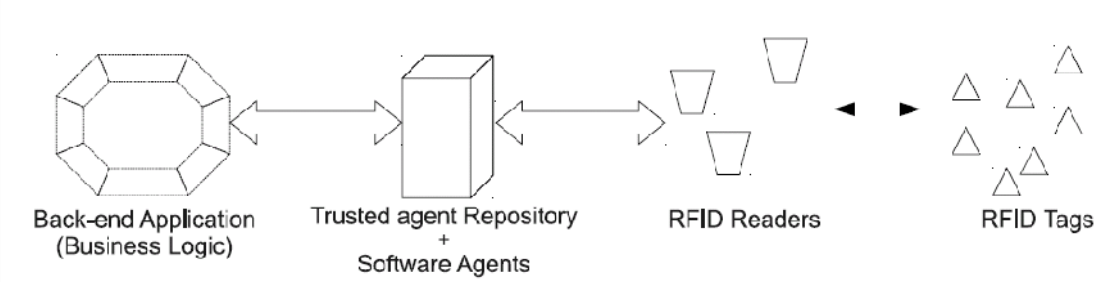
Η ανάγκη για πρωτόκολλα ασφάλειας σε RFID περιβάλλοντα δημιούργησε, όπως βλέπουμε στην βιβλιογραφία, αρκετές λύσεις. Οι περισσότερες λύσεις όμως δημιούργησαν ένα σημαντικό πρόβλημα αυτό, της διαλειτουργικότητας που προκαλείται από την πολυμορφία των κρυπτογραφικών πρωτοκόλλων. Σύμφωνα με τα προβλήματα αυτά ο Ρεκλείτης et al. προτείνει μια λύση με πράκτορες λογισμικού που απλοποιεί την ενσωμάτωση και την διαχείριση ετερογενών RFID ετικετών. Το νέο σύστημα μπορεί να υλοποιήσει διαφορετικά πρωτοκόλλα ασφαλείας αλλά και επικοινωνίας, μεταβιβάζοντας ένα κομμάτι από τις διαχειριστικές και επικοινωνιακές λειτουργίες από το back-end σύστημα και τον αναγνώστη RFID, σε μια λύση που θα αντιστοιχίζει κάθε πράκτορα λογισμικού, που θα βρίσκεται σε ένα repository, σε μία ετικέτα.

2.2. RFID πλατφόρμα βασισμένη σε πράκτορες λογισμικού

Στις περισσότερες εργασίες πάνω στα RFID συστήματα γίνεται ο διαχωρισμός σε 3 βασικά μέρη: τις RFID ετικέτες, του αναγνώστες των ετικετών και το back-end σύστημα.
Το back-end σύστημα είναι υπεύθυνο για την διαχείριση των πληροφοριών που σχετίζονται με την ετικέτα. Θεωρούμε το back-end σύστημα ως τον συνδυασμό της back-end βάσης δεδομένων που συνδέει τα αναγνωριστικά των ετικετών με πληροφορίες σχετικές με τα αντικείμενα που βρίσκεται η ετικέτα και ένα back-end υποσύστημα εφαρμογής, που εκτελεί λειτουργίες σχετικές με την επιχείρηση. Μπορούμε να υποθέσουμε ότι η επικοινωνία μεταξύ του back-end συστήματος και των αναγνωστών είναι ασφαλής, ενώ η επικοινωνία μεταξύ αναγνωστών και ετικετών δεν είναι .



Σχήμα 8. *RFID σύστημα*



Σχήμα 9. *RFID σύστημα με πράκτορες λογισμικού*

Το πρωτόκολλο που θα μας απασχολήσει φαίνεται σχήμα 9. Η back-end βάση δεδομένων και οι εγγραφές σχετικά με τις ετικέτες αντικαθιστούνται από ένα repository πρακτόρων λογισμικού και τους πράκτορες λογισμικού για τις ετικέτες.
Το repository των πρακτόρων λογισμικού είναι μια πλατφόρμα που παρέχει, στους διαμένοντες ή επισκεπτόμενους πράκτορες, τους απαραίτητους πόρους και υπηρεσίες. Από πλευράς ασφάλειας, διακρίνουμε τις αποθήκες των πρακτόρων σε έμπιστες και μη έμπιστες. Η έμπιστη είναι αυτή που συμφωνεί με την πολιτική ασφάλειας και ιδιωτικότητας (S&P policy) λειτουργεί σαν ασφαλές καταφύγιο για τους διαχειριζόμενους πράκτορες λογισμικού.
Ένας πράκτορας λογισμικού για τις ετικέτες είναι μια αυτόνομη, οντότητα λογισμικού, που διαχειρίζεται μια μόνο RFID ετικέτα. Κάθε πράκτορας αποθηκεύει όλες τις σχετικές με την ετικέτα πληροφορίες, περιλαμβανομένου και όλα τα δεδομένα που απαιτούνται για να αλληλεπιδράσει με το back-end σύστημα και να διαχειριστεί την RFID ετικέτα. Υποθέτουμε ότι οι πράκτορες λογισμικού που σχετίζονται με τις ετικέτες και βρίσκονται σε ένα έμπιστο repository για λόγους ασφάλεια δεν επιτρέπονται να ταξιδεύουν εκτός αυτού. Ωστόσο η κινητικότητα είναι ένα επιθυμητό χαρακτηριστικό, αφού θα υπάρχουν περιπτώσεις που θα θέλαμε να

στείλουμε έναν πράκτορα σε μία ξένη πλατφόρμα, π.χ. όταν ένα αντικείμενο που έχει πάνω μια RFID ετικέτα αλλάζει ιδιοκτήτη ή όταν θέλουμε να δώσουμε δικαίωμα ανάγνωσης σε έναν άλλο αναγνώστη που δεν ανήκει στον ιδιοκτήτη. Γι' αυτό επιτρέπεται η δημιουργία κλώνων των πρακτόρων λογισμικού οι οποίοι περιέχουν ένα υποσύνολο από της πληροφορίες του γνήσιου πράκτορα λογισμικού.

2.2.1. Βασικές λειτουργίες της ετικέτας

Παρακάτω θα περιγραφεί πώς εκτελούνται οι εργασίες που σχετίζονται με την ετικέτα από την πλατφόρμα των πρακτόρων λογισμικού.

Έναρξη της ετικέτας: Κατά την διάρκεια της έναρξης μιας ετικέτας (δηλ. της ανάθεσης ενός μοναδικού αναγνωριστικού (uID, unique identifier) και ενός μυστικού), ένας πράκτορας λογισμικού δημιουργείται που έχει όλες τις απαραίτητες πληροφορίες (διαπιστευτήρια, πολιτικές, πληροφορίες σχετικές με την ετικέτα, πρωτόκολλα επικοινωνίας κ.α.) για να επικοινωνήσει με ασφάλεια και με το back-end σύστημα καθώς και με την ετικέτα.

Πρόσβαση στα δεδομένα της ετικέτας: Οι πράκτορες επιβάλλουν μηχανισμούς ελέγχους πρόσβασης, για επιτρέπουν σε εξουσιοδοτημένες οντότητες να προσπελάσουν, προσθέσουν ή να τροποποιήσουν τα δεδομένα της ετικέτας, βασιζόμενοι στα δικαιώματα. Έτσι υπάρχει η δυνατότητα για έναν τακτικό και αυτόματο τρόπο για να αναβαθμιστούν/τροποποιηθούν πληροφορίες (π.χ. τοποθεσία, κατάσταση, κ.α.) σχετικό με το αντικείμενο που έχει την ετικέτα.

Αυθεντικοποίηση Ετικέτας/ Ανανέωση μυστικού: Ο πράκτορας λογισμικού έχει τη δυνατότητα να δίνει οδηγίες στον αναγνώστη για να χρησιμοποιήσει το σωστό πρωτόκολλο, για την επικοινωνία με την ετικέτα, και παρέχει τις απαιτούμενες πληροφορίες. Με γνώμονα την ασφάλεια, ο πράκτορας ξέρει ποιο πρωτόκολλο αυθεντικοποίησης υλοποιεί η ετικέτα και γνωρίζει τα μυστικά κλειδιά που χρησιμοποιούνται και έχει την δυνατότητα να ανανεώνει και να παράγει αυτά τα κλειδιά, αναγνωριστικά, ψευδώνυμα, κ.α., αναλόγως.

Μεταβίβαση ιδιοκτησίας της ετικέτας: Για υλοποιήσεις ετικετών που υποστηρίζουν ασφαλή μεταβίβαση ιδιοκτησίας της ετικέτας, το back-end σύστημα που βασίζεται στους πράκτορες διευκολύνει την διαδικασία και διαχείριση των πληροφοριών. Ο προηγούμενος ιδιοκτήτης είναι σε θέση να αποφασίσει την ποσότητα των πληροφοριών που θέλει να μεταβιβάσει στο νέο ιδιοκτήτη. Αυτό επιτυγχάνεται με την δημιουργία και προώθηση ενός , σωστά δημιουργημένου κλώνου πράκτορα λογισμικού ο οποίος υποστηρίζει την λειτουργία της ανανέωση του μυστικού. Για παράδειγμα, σε ένα κατάστημα λιανικής πώλησης ο ιδιοκτήτης θα αφαιρέσει οποιοδήποτε ευαίσθητο δεδομένων που έχει σχέση με το σύστημα εφοδιασμού του καταστήματος, επίσης θα πραγματοποιήσει την λειτουργία ανανέωσης του μυστικού (για να αλλάξει το μυστικό που περιέχεται μέσα στην ετικέτα σε μία προσωρινή τιμή). Ο πράκτορας κλώνος θα δημιουργηθεί με το προσωρινό μυστικό, μαζί με δεδομένα που απαιτούνται από το νόμο (π.χ. ημερομηνία λήξης, κατασκευαστής, κ.α.) και δεδομένα που διευκολύνουν τις υπηρεσίες μετά την πώληση του προϊόντος. Σαν τελευταίο βήμα, ο νέος ιδιοκτήτης αναλαμβάνει την ιδιοκτησία της ετικέτας ανανεώνοντας το μυστικό σε ένα ελεγχόμενο περιβάλλον (για να αποφύγει ωτακουστές στο κατάστημα).

Μεταβίβαση δικαιωμάτων ετικέτας: Αυτή η λειτουργία υποστηρίζεται με την δημιουργία πρακτόρων κλώνων. Ένας κατάλληλος κλώνος δημιουργείται και προωθείται στην αντίστοιχη οντότητα, επιτρέποντας αλληλεπίδραση με την αντίστοιχη ετικέτα, χωρίς πρόσβαση στο repository του ιδιοκτήτη. Για υλοποιήσεις

που δεν υποστηρίζουν την ανάκληση της μεταβίβασης των δικαιωμάτων, ο ιδιοκτήτης της ετικέτας πρέπει να χρησιμοποιεί την μεταβίβαση δικαιωμάτων μόνο για έμπιστες οντότητες που βρίσκονται στον έλεγχο του. Σε αντίθεση όταν είναι δυνατή και επιθυμητή η ανάκληση, ο ιδιοκτήτης της ετικέτας παρέχει έναν ειδικά δημιουργημένο πράκτορα κλώνο, σύμφωνα με τις οδηγίες του πρωτοκόλλου. Όταν η μεταβίβαση δικαιωμάτων λήξει ή ανακληθεί, ο κλώνος θα σταματήσει την λειτουργία του και η οντότητα που είχε λάβει τα δικαιώματα δεν θα είναι σε θέση να επικοινωνήσει με την ετικέτα πλέον. Το πρωτόκολλο που θα αναλυθεί παρακάτω παρέχει προσωρινή και ανακαλούμενη μεταβίβαση δικαιωμάτων.

2.2.2. Πλεονεκτήματα

Υποστήριξη ετερογενών ετικετών: Εφόσον η αλληλεπίδραση ετικέτας-αναγνώστη επιβλέπεται από πράκτορες λογισμικού, το back-end δεν χρειάζεται να γνωρίζει τις λεπτομέρειες της υλοποίησης.
Απλοποιημένη διαχείριση του κλειδιού: Όλες οι σχετικές πράξεις ανατίθενται στον πράκτορα λογισμικού, ελαχιστοποιώντας την πολυπλοκότητα της διαχείρισης ετερογενών ετικετών ή ετικετών που έχουν διαφορετικές ανάγκες από πλευράς ασφάλειας και ιδιωτικότητας.
Διευκόλυνση μεταβίβαση ιδιοκτησίας και δικαιωμάτων ετικέτας: Για υλοποιήσεις που υποστηρίζουν τόσο προηγμένες λειτουργίες, η πλατφόρμα παρέχει τους κατάλληλους κλώνους πράκτορες μαζί με τις κατάλληλες πολιτικές ιδιωτικότητας, το για να διαχειριστούν την αποκάλυψη πληροφοριών. Στην μεριά του παραλήπτη προφέρουν το πλεονέκτημα της μεταφοράς των πληροφοριών της υλοποίησης, ελαττώνοντας της ανησυχίες για την παρουσίαση και ενοποίηση μιας άγνωστης τεχνολογίας στην πλατφόρμα του.
Υποστήριξη πολύπλοκων business logic: Η προτεινόμενη υποδομή μπορεί να παρουσιαστεί για να χειριστεί τα δεδομένα των ετικετών, σύμφωνα με τις ανάγκες του οργανισμού, παρουσιάζοντας την αυτοματοποίηση στην συλλογή και διαχείριση των δεδομένων.

2.3. Μια νέα σουίτα από ¨ελαφριά¨ πρωτόκολλα διαχείρισης

Στο κεφάλαιο περιγράφεται μια σουίτα από ελαφριά (lightweight) πρωτόκολλα διαχείρισης ετικετών, που εκμεταλλεύονται την προτεινόμενη υποδομή με τους πράκτορες λογισμικού. Τα προτεινόμενα πρωτόκολλα υποστηρίζουν λειτουργίες για την ετικέτα όπως αυθεντικοποίηση, μεταβίβαση και ανάκληση δικαιωμάτων και μεταβίβαση ιδιοκτησίας, ικανοποιώντας ταυτόχρονα σημαντικές απαιτήσεις ασφάλειας και ιδιωτικότητας, όπως η εμπιστευτικότητα και η μη-ανιχνευσιμότητα.

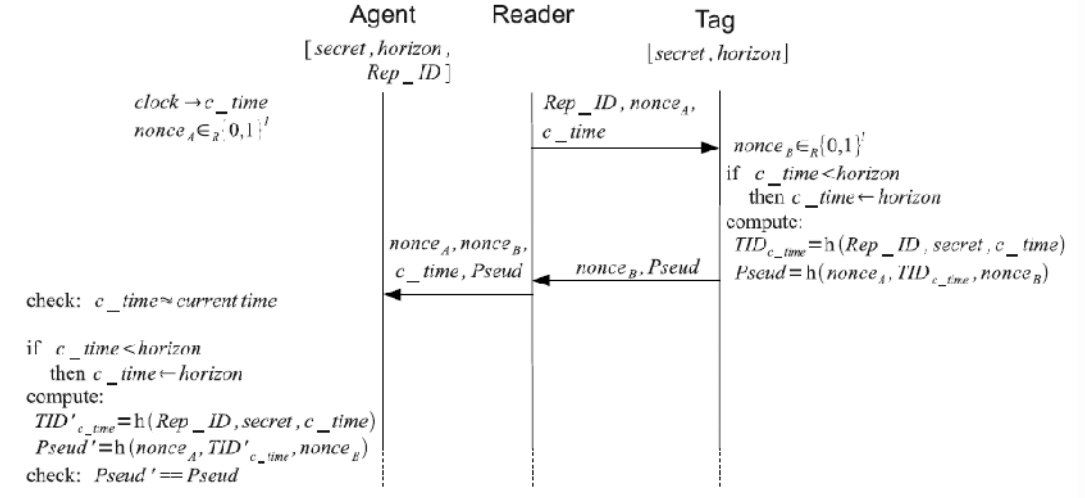
2.4. Βασικά πρωτόκολλα

Η προτεινόμενη σουίτα υποστηρίζει την μεταβίβαση δικαιωμάτων (με την χρήση προσωρινών ψευδωνύμων που βασίζονται στο χρόνο) και μεταβίβαση ιδιοκτησίας που εξασφαλίζει την ιδιωτικότητα (με ανανέωση του μυστικού), ενώ απαιτεί περιορισμένες υπολογιστικές δυνατότητες.
Η ετικέτα πρέπει να ενσωματώνει μια ασφαλή μονόδρομη συνάρτηση hash h(-) και μια ψευδοτυχαία γεννήτρια αριθμών. Επιπροσθέτως η ετικέτα χρειάζεται να

αποθηκεύει 3 τιμές, το μυστικό (secret), που μοιράζεται η ετικέτα με τον αντίστοιχο πράκτορα, ένα αναγνωριστικό ετικέτας που βασίζεται στον χρόνο (time-dependent tag identifier, TID) και μία χρονική τιμή (horizon), που καθορίζει ένα συγκεκριμένο σημείο στον χρόνο και είναι γνωστό. Ο χρόνος είναι μια σημαντική έννοια για την μεταβίβαση δικαιωμάτων της ετικέτας και η αναπαράσταση του συμμορφώνεται με το διεθνές πρότυπο ISO 8601.
Παρακάτω περιγράφεται η βασική σουίτα πρωτοκόλλων, δηλαδή το αίτημα για την ετικέτα, το αίτημα μεταβίβαση δικαιωμάτων της ετικέτας, ανανέωση του μυστικού και ανανέωση του χρονικού ορίζοντα. Το πρωτόκολλα για το αίτημα μεταβίβασης δικαιωμάτων της ετικέτας είναι σημαντικό γιατί εκμεταλλεύεται την πλατφόρμα που βασίζεται στους πράκτορες λογισμικού. Η μεταβίβαση των δικαιωμάτων είναι προσωρινή και θα ανακληθεί αυτόματα μετά από ένα χρονικό διάστημα.

2.4.1. Αίτημα ετικέτας

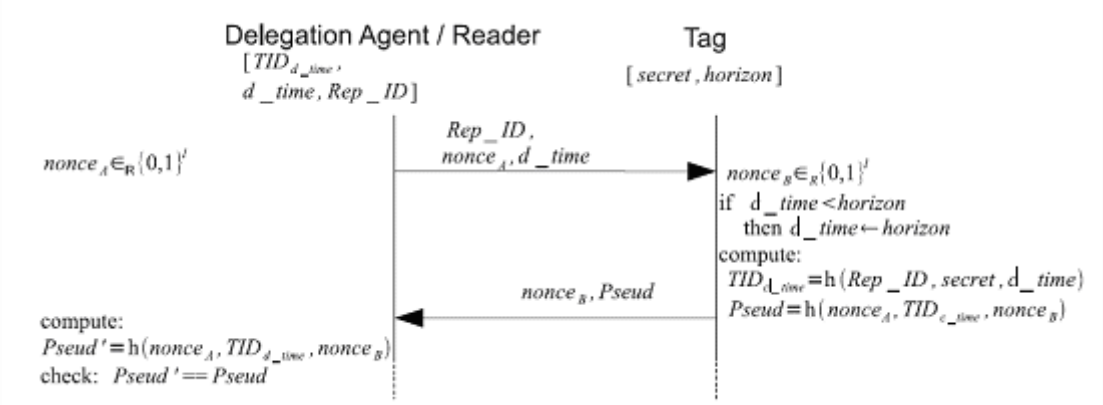
1. Ο αναγνώστης στέλνει στην ετικέτα: το αναγνωριστικό του repository του αναγνώστη (Rep_ID), ένα τυχαίο $nonce_A$, που προκύπτει από την ψευδοτυχαία γεννήτρια αριθμών και τον τρέχοντα χρόνο, c_time.
2. Αν ο c_time υποδεικνύει ένα σημείο στο χρόνο 'παλιότερο από' τον ορίζοντα (horizon), τότε η ετικέτα αντικαθιστά το c_time με τον ορίζοντα. Παράγει ένα δεύτερο $nonce_B$ και υπολογίζει ένα αναγνωριστικό βασισμένο στο χρόνο TID_{c_time} = h (Rep ID, secret, c time). Έπειτα υπολογίζει ένα ψευδώνυμο Pseud = h(nonceA, $TID_{c\_time,}$ nonceB), το οποίο αποστέλλεται στον αναγνώστη μαζί με το nonceB.
3. Ο αναγνώστης προωθεί τις ποσότητες που έλαβε, μαζί με το $nonce_A$ και το c_time στο repository.
4. Στο repository, η φρεσκάδα του ληφθέντα c_time ελέγχεται με το ρολόι και αν βρεθεί μια ασυμφωνία, πραγματοποιούνται οι κατάλληλες ενέργειες π.χ. η εκκίνηση ενός συναγερμού. Περαιτέρω, κάθε πράκτορας συγκρίνει το c_time με τον αποθηκευμένη τιμή του ορίζοντα και αν βρεθεί παλαιότερη την αντικαθιστά με τον ορίζοντα. Στη συνέχεια, κάθε πράκτορας υπολογίζει το δικό του TID ($TID'_{c\ time}$ = h(Rep ID, secret, c time)) και έπειτα υπολογίζει το ψευδώνυμο Pseud= h($nonce_A$, $TID'_{c\_time,}$ nonceB) και το συγκρίνει με το ψευδώνυμο που έχει λάβει από την ετικέτα.



Σχήμα 10. *Σχήμα για το αίτημα ετικέτας*

2.4.2. Αίτημα για μεταβίβαση δικαιωμάτων ετικέτας

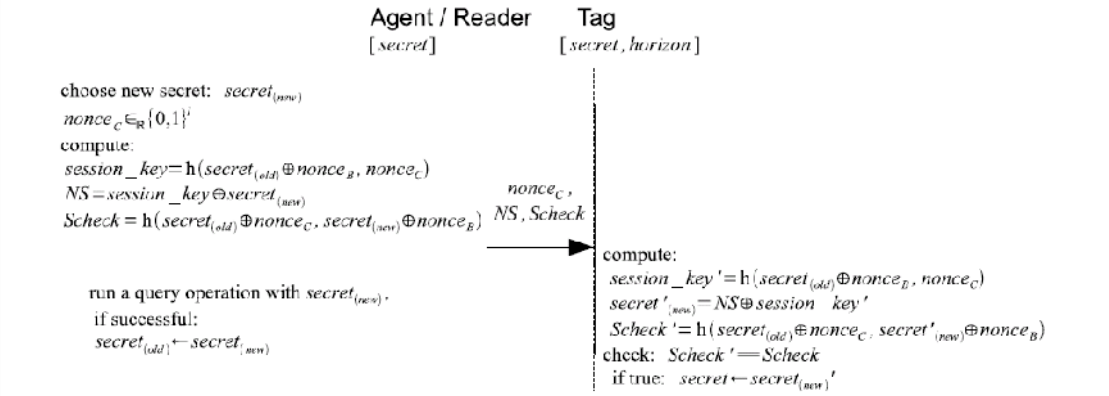
Το πρωτόκολλο το οποίο φαίνεται στην σχήμα 11, είναι ταυτόσημο με το πρωτόκολλο για το αίτημα ετικέτας που περιγράφηκε παραπάνω, με μία αξιοσημείωτη διαφορά, ότι χρησιμοποιείται ένας κλώνος πράκτορα λογισμικού. Ο ιδιοκτήτης της ετικέτας δημιουργεί ένα πράκτορα κλώνο για την μεταβίβαση των δικαιωμάτων, d_Agent, και το στέλνει στον προσωρινό χρήστη της ετικέτας. Ο πράκτορας κλώνος, δεν αποθηκεύει το μυστικό (secret), αλλά ένα αναγνωριστικό βασισμένο στον χρόνο T ID_{d_time} = h(Rep ID, secret, d_time) και την και την αντίστοιχη χρονική ποσότητα (d_time). Το αναγνωριστικό υπολογίζεται από τον αρχικό πράκτορα λογισμικού που διαχειρίζεται την ετικέτα, για να χρησιμοποιηθεί από ένα συγκεκριμένο Repository (Rep_ID), για μια προκαθορισμένη χρονική περίοδο. Ο d_Agent μπορεί να χρησιμοποιηθεί για να ανιχνεύει και να εντοπίσει την ετικέτα για όσο η τιμή του ορίζοντα, που είναι αποθηκευμένη στην ετικέτα, είναι προγενέστερο ή ίσο με το d_time. Επομένως, ο ιδιοκτήτης της ετικέτας μπορεί να ανακαλέσει την μεταβίβαση δικαιωμάτων, απλά ανανεώνοντας τον ορίζοντα σε μία μεταγενέστερη χρονική τιμή. Μόλις η ετικέτα ανανεωθεί με το πρωτόκολλο ενημέρωσης του ορίζοντα, ο κλώνος d_Agent γίνεται ξεπερασμένος.



Σχήμα 11. *Σχήμα για το αίτημα μεταβίβασης δικαιωμάτων ετικέτας*

2.4.3. Ενημέρωση μυστικού ετικέτας

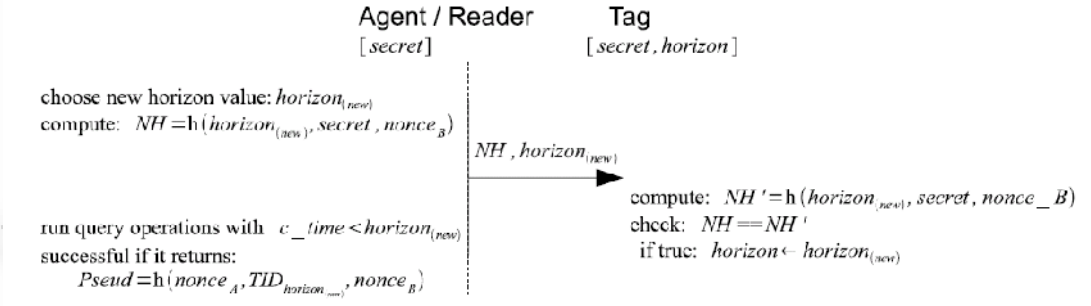
1. Εκτέλεση πρωτοκόλλου για αίτημα ετικέτας και αν γίνει επιτυχώς συνέχιση.
2. Ο πράκτορας λογισμικού επιλέγει ένα νέο μυστικό ($secret_{(new)}$), δημιουργεί ένα τρίτο $nonce_C$ και υπολογίζει ένα κλειδί συνόδου session_key = h($secret_{(old)} \oplus nonce_B$, $nonce_C$). Έπειτα πραγματοποιεί και στα δύο XOR (NS= session_key $\oplus$ $secret_{(new)}$) και υπολογίζει μια τιμή checksum Scheck= h($secret_{(old)} \oplus nonce_C$, $secret_{(new)} \oplus nonce_B$).Και στέλνει στον αναγνώστη τις τρεις τιμές: $nonce_C$, NS και Scheck.
3. Ο αναγνώστης προωθεί και τις τρεις ποσότητες στην ετικέτα.
4. Η ετικέτα υπολογίζει το κλειδί συνόδου session_key′ = h($secret_{(old)} \oplus nonce_B$, $nonce_C$), και εξάγει το νέο μυστικό ($secret'_{(new)}$ = NS $\oplus$ session key′) και υπολογίζει Scheck′ = h($secret_{(old)} \oplus nonce_C$, $secret'_{(new)} \oplus nonce_B$). Αν το checksum συμφωνεί τότε αντικαθιστά το παλιό μυστικό με το $secret'_{(new)}$.
5. Ο πράκτορας λογισμικού πραγματοποιεί το αίτημα ετικέτας χρησιμοποιώντας το secret_new, αν ολοκληρωθεί επιτυχώς τότε η ετικέτα ενημερώθηκε σωστά και το παλιό μυστικό του πράκτορα αντικαθίσταται από το νέο.



Σχήμα 12. *Σχήμα για την ενημέρωση μυστικού της ετικέτας*

2.4.4. Ενημέρωση χρονικού ορίζοντα

1. Εκτέλεση πρωτοκόλλου για αίτημα ετικέτας και αν γίνει επιτυχώς συνέχιση.
2. Ο πράκτορας λογισμικού επιλέγει την νέα τιμή του ορίζοντα, $horizon_{(new)}$ και υπολογίζει ένα checksum
$$NH = h(horizon_{(new)}, secret, nonce_B).$$
3. Ο αναγνώστης προωθεί και τις δύο ποσότητες στην ετικέτα.
4. Η ετικέτα υπολογίζει $NH' = h(horizon_{(new)}, secret, nonce_B)$ και ελέγχει αν είναι ίσο με το NH που έλαβε. Αν ναι αντικαθιστά το υπάρχον $horizon_{(old)}$ με το $horizon_{(new)}$.
5. Ο πράκτορας λογισμικού εκτελεί ένα μεταγενέστερο αίτημα ετικέτας, χρησιμοποιώντας μία τιμή c_time παλαιότερη από το $horizon_{(new)}$. Εάν η ετικέτα απαντήσει με ένα ψευδώνυμο που δημιουργήθηκε χρησιμοποιώντας την τιμή $TID_{horizon(new)}$, μπορούμε να υποθέσουμε ότι η τιμή του ορίζοντα ενημερώθηκε επιτυχώς.



Σχήμα 13. *Σχήμα για την ενημέρωση του χρονικού ορίζοντα*

2.5. Ανάλυση ασφαλείας

Το πιο σημαντικό πρωτόκολλο από άποψης ασφάλειας είναι το πρωτόκολλο μεταβίβασης δικαιωμάτων της ετικέτας καθώς αυτό εκμεταλλεύεται την νέα πλατφόρμα με τους πράκτορες λογισμικού.

Δεν υπάρχει αμφιβολία για την ασφάλεια του πράκτορα λογισμικού που χρησιμοποιείται για την μεταβίβαση των δικαιωμάτων. Μόλις φύγει από το έμπιστο repository, ο νέος ιδιοκτήτης μπορεί να το παραποιήσει, με όποιον τρόπο θέλει, π.χ. να κρατήσει μόνο τα δεδομένα και τα $TID_{d\_time,}$ και αφαιρώντας οτιδήποτε άλλο έχει σχέση με τον d_Agent. Συνεπώς, η εφαρμογή, στην λογική των κλώνων των πρακτόρων λογισμικού, ελέγχων για την φρεσκάδα ή οποιοσδήποτε άλλος έλεγχος, πέρα από αυτόν που επιβάλλεται από το TID, δεν θα αυξήσει την ασφάλεια. Η ασφάλεια του πρωτοκόλλου εξαρτάται μόνο από την ασφάλεια της μονόδρομης hash συνάρτησης, της τυχαιότητας της γεννήτριας και το μήκος (bits) των μυστικών.
Παρακάτω απαριθμούνται μερικές από τις ιδιότητες ασφάλειας και ιδιωτικότητας που τις ικανοποιούνται από το πρωτόκολλο:

1. **Ιδιωτικότητα πληροφοριών ετικέτας:** Ο κλώνος πράκτορας λογισμικού που χρησιμοποιείται για την μεταβίβαση δικαιωμάτων ετικέτας, περιέχει τις ελάχιστες δυνατές πληροφορίες που αφορούν την ετικέτα, δηλαδή όλες τις πληροφορίες που ο προσωρινός χρήστης χρειάζεται να ξέρει. Η πιο σημαντική πληροφορία που αφορά την ετικέτα είναι, φυσικά, το μυστικό κλειδί. Ωστόσο, από κατασκευής, μόνο οι τιμές TID_{d_time} και d_time αποθηκεύονται στον πράκτορα κλώνο. Έτσι ακόμα και αν κάποιος επιτιθέμενος, αποκτήσει τον κλώνο, το όφελος του θα είναι αυτές οι τιμές. Αφότου η μονόδρομη hash συνάρτηση που χρησιμοποιείται θεωρείται ασφαλής, ο επιτιθέμενος δεν μπορεί να μάθει επιπλέον πληροφορίες.
2. **Ιδιωτικότητα θέσης της ετικέτας:** Οι αποκρίσεις της ετικέτας είναι ανώνυμες, εφόσον η ετικέτα εκπέμπει, μόνο, δύο μηνύματα, $nonce_B$ and Pseud, που διαφοροποιούνται, με ένα τυχαίο τρόπο, κάθε φορά που φθάνει ένα νέο αίτημα. Έτσι, αυτά τα μηνύματα δεν μπορούν να συνδεθούν με κάποια ετικέτα.
3. **Επίθεση πλαστοπροσωπίας ετικέτας:** Αυτή η επίθεση είναι δυνατή μόνο αν η ετικέτα αντιγραφεί. Σε αντίθετη περίπτωση όπως αναφέρθηκε και πριν πρέπει να γίνει αντιστροφή της μονόδρομης hash, κάτι το οποίο είναι υπολογιστικά ανέφικτο.
4. **Ανθεκτικότητα σε επιθέσεις επανάληψης:** Το πρωτόκολλο είναι ένα τυπικό πρωτόκολλο αυθεντικοποίησης ερώτησης-απόκρισης που χρησιμοποιεί τυχαίους αριθμούς για να αντέξει τις επιθέσεις επανάληψης.
5. **Ανθεκτικότητα στον εντοπισμό θέσης μιας ετικέτας, χρησιμοποιώντας έναν ανακλημένο κλώνο πράκτορα που χρησιμοποιήθηκε για την μεταβίβαση δικαιωμάτων:** Αυτή η επίθεση θεωρείται επιτυχής όταν ο επιτιθέμενος είναι σε θέση να εντοπίσει μια ετικέτα που έχει μεταβιβάσει το δικαίωμα ανάγνωση της, ακόμα και αν αυτό το δικαίωμα έχει ανακληθεί. Υπάρχουν δύο κύριες ανησυχίες η πρώτη έχει να κάνει με την ακρίβεια του τρέχοντος χρόνου και η άλλη με την προσωρινή τιμή TID_{d_time}. Ο σχεδιασμός του πρωτοκόλλου έγινε με τέτοιο τρόπο έτσι ώστε η ακρίβεια του τρέχοντα χρόνου να μην είναι σημαντική. Με άλλα λόγια, δεν αναμένεται όλες οι οντότητες να είναι αληθής ως προς την ακρίβεια του ρολογιού, που θα δείχνει την ώρα και την ημερομηνία. Και θα δούμε παρακάτω γιατί μία ψευδής ώρα δεν επηρεάζει την ασφάλεια του πρωτοκόλλου.
   Υποθέτοντας ότι ο επιτιθέμενος έχει πρόσβαση σε έναν κλώνο πράκτορα λογισμικού που χρησιμοποιήθηκε για την μεταβίβαση δικαιωμάτων και έχει πλέον ανακληθεί. Αν αποφασίσει να στείλει οποιαδήποτε χρονική τιμή, που είναι παλαιότερη από τον ορίζοντα που είναι αποθηκευμένο στην ετικέτα, η ετικέτα θα απαντήσει με ένα τυχαίο nonce και ένα ψευδώνυμο που βασίζεται

στον ορίζοντα και στο nonce. Δεδομένης της ασφάλειας που παρέχει η μονόδρομη hash συνάρτηση, δεν θα είναι σε θέση να σε θέση να αντιστοιχήσει με αυτά με την ποσότητα TID_{d_time} του πράκτορα κλώνου ή να το ξεχωρίσει από μια τυχαία τιμή. Αν στείλει μια σύγχρονη χρονική τιμή (που ισούται ή είναι μεταγενέστερη από τον ορίζοντα), η απάντηση της ετικέτας θα φανεί και πάλι τυχαία, παρότι το ψευδώνυμο θα βασίζεται στο nonce και στην χρονική τιμή που στάλθηκε. Το παραπάνω μας δείχνει ότι ακόμα και αν ο επιτιθέμενος ανακτήσει το TID_{d_time} από τον πράκτορα κλώνο δεν θα είναι σε θέση να εξάγει καμία περαιτέρω πληροφορία που αφορά την ετικέτα [6].

3. Η πλατφόρμα JADE (Java Agent DEvelopment Framework)

3.1. Ιστορία της πλατφόρμας

Η πλατφόρμα JADE ξεκίνησε το 1998 από την Telecom Italia (τώρα CSELT), με σκοπό να ικανοποιηθούν οι ανάγκες από τις προδιαγραφές που είχαν τεθεί από τον FIPA (The Foundation for Intelligent Physical Agents). Στην αρχή ξεκίνησε απλά για να καλύψει τις προδιαγραφές FIPA αλλά με το πέρασμα του χρόνου εξελίχθηκε σε μία πλατφόρμα, με γνώμονα την απλότητα και την χρηστικότητα έτσι ώστε να είναι εύκολο ακόμα και σε αρχάριους χρήστες να αναπτύξουν τις εφαρμογές τους.
Το 2000 το JADE έγινε πλέον ανοιχτού κώδικα με την LGPL (Library Gnu Public Licence) άδεια. Με αυτήν την άδεια εξασφαλίζεται η χρήση του JADE και σε εμπορικά προϊόντα, το δικαίωμα για να γίνουν αντίγραφα της πλατφόρμας καθώς και το δικαίωμα για πρόσβαση και αλλαγή στον πηγαίο κώδικα. Το 2003 για να διευκολυνθεί η ανάπτυξη της πλατφόρμας η Telecom Italia μαζί με την Motorola δημιούργησαν έναν μη κερδοσκοπικό οργανισμό τον JADE Governeing Board, στον οποίο συμμετείχαν εταιρίες όπως η France Telecom R&D, Whitestein Technologies AG και Profactor Gmbh.
Το JADE μπορεί να το κατεβάσει κάποιος από το http://jade.tilab.com μαζί με παραδείγματα, βιβλιογραφία που θα βοηθήσουν στην χρήση του.

3.2. Το JADE και το μοντέλο των πρακτόρων λογισμικού

Το JADE είναι μια πλατφόρμα λογισμικού που παρέχει βασικές middleware λειτουργίες οι οποίες είναι ανεξάρτητες από την συγκεκριμένη εφαρμογή και μπορεί να απλοποιήσει την πραγματοποίηση των κατανεμημένων εφαρμογών που εκμεταλλεύονται του πράκτορες λογισμικού (Wooldridge and Jennings, 1995).

Ένας πράκτορας είναι αυτόνομος και προνοητικός: Ένας πράκτορας δεν μπορεί να παρέχει κλήσεις του δικού του προγράμματος σε άλλους πράκτορες έτσι ώστε να μειώσει τις πιθανότητες άλλες οντότητες να πάρουν τον έλεγχο στις υπηρεσίες του. Ένας πράκτορας πρέπει να έχει το δικό του νήμα εκτέλεσης, χρησιμοποιώντας το για να ελέγχει τον κύκλο ζωής του και να αποφασίζει αυτόνομα για το πότε και ποιες πράξεις θα εκτελέσει.

Οι πράκτορες μπορούν να πουν 'όχι', και είναι χαλαρά συνδεδεμένοι: Η ασύγχρονη επικοινωνία που βασίζεται σε μηνύματα είναι η βασική μορφή για την επικοινωνία μεταξύ πρακτόρων στο JADE. Ένας πράκτορας που επιθυμεί να επικοινωνήσει πρέπει να στείλει ένα μήνυμα προς ένα αναγνωρισμένο προορισμό (ή

ένα σύνολο προορισμών). Δεν υπάρχει εξάρτηση μεταξύ αποστολέα και παραλήπτη, δηλαδή ένας παραλήπτης μπορεί να μην είναι διαθέσιμος όταν ένας αποστολέας εκδίδει το μήνυμα. Δεν χρειάζεται επίσης να εξασφαλίσει την αναφορά αντικειμένου του παραλήπτη-πράκτορα, αλλά μόνο ταυτότητες που το σύστημα μεταφοράς μηνυμάτων είναι σε θέση να ανάγει στις σωστές διευθύνσεις μεταφοράς.

Το σύστημα είναι Peer-to-Peer: Κάθε πράκτορας αναγνωρίζεται από ένα παγκόσμια μοναδικό όνομα (το AgentIdentifier ή AID, όπως ορίζει ο FIPA) και μπορεί να συμμετέχει ή να φύγει σε μία πλατφόρμα οποιαδήποτε ώρα και να ανακαλύψει άλλου πράκτορες διαμέσου των υπηρεσιών white και yellow pages (που παρέχονται στο JADE από τους AMS και DF πράκτορες, όπως ορίζονται από τον FIPA). Ένας πράκτορας μπορεί να αρχίσει επικοινωνία με οποιονδήποτε άλλο πράκτορα οποιαδήποτε ώρα, καθώς και να δεχθεί αίτημα επικοινωνίας οποιαδήποτε στιγμή.

3.3. Η αρχιτεκτονική του JADE

Μια πλατφόρμα JADE συνθέτεται από δοχεία (containers) πρακτόρων λογισμικού που μπορεί να είναι κατανεμημένα σε ένα δίκτυο. Οι πράκτορες λογισμικού ζουν στα δοχεία οποία είναι οι η διεργασία της Java που παρέχει το JADE run-time και τις υπηρεσίες που χρειάζονται για την φιλοξενία και την εκτέλεση των πρακτόρων λογισμικού. Υπάρχει ένα ειδικό δοχείο, που ονομάζεται κύριο (main container), το οποίο αναπαριστά το σημείο εκκίνησης για την πλατφόρμα: είναι το πρώτο δοχείο που ξεκινάει και όλα τα άλλα δοχεία πρέπει να ενωθούν με το κύριο κάνοντας την καταχώρηση μαζί με αυτό.
Στο σημείο εκκίνησης, το κύριο δοχείο έχει τις ακόλουθες αρμοδιότητες:

- Διαχείριση του πίνακα δοχείων (container table, CT), που είναι η καταχώρηση για την κλήση αντικειμένων και όλες τις διευθύνσεις μεταφοράς όλων των δοχείων που απαρτίζουν την πλατφόρμα.
- Διαχείριση του παγκόσμιου πίνακα περιγραφής πρακτόρων λογισμικού (global agent descriptor table, GADT), που είναι η καταχώρηση όλων των πρακτόρων που βρίσκονται στην πλατφόρμα, συμπεριλαμβανομένου της τρέχοντας κατάστασης και της θέσης.
- Φιλοξενία των AMS και DF πρακτόρων, που είναι δυο πράκτορες που παρέχουν διαχείριση των πρακτόρων, την υπηρεσία λευκών σελίδων (white page) καθώς και την υπηρεσία κίτρινων σελίδων (yellow page) της πλατφόρμας.

Η ταυτότητα του πράκτορα λογισμικού είναι ένα αναγνωριστικό του πράκτορα (Agent Identifier, AID), που αποτελείται από ένα σύνολο από στοιχεία που συμμορφώνονται με τη δομή που ορίζει ο FIPA. Τα κύρια στοιχεία του AID είναι το όνομα του πράκτορα και η διεύθυνση του. Το όνομα του πράκτορα είναι ένα παγκόσμια μοναδικό αναγνωριστικό που το JADE δημιουργεί ενώνοντας ένα όνομα που δημιουργείται από τον χρήστη (τοπικό όνομα) και το όνομα της πλατφόρμας. Οι διευθύνσεις των πρακτόρων είναι διευθύνσεις μεταφοράς που κληρονομούνται από την πλατφόρμα, που κάθε διεύθυνση πλατφόρμας αντιστοιχίζεται σε ένα MTP (Message Transport Protocol) τερματικό όπου μηνύματα που έχουν δημιουργηθεί σύμφωνα με τον FIPA μπορούν να ληφθούν και να αποσταλούν.
Όταν εκκινήσει το κύριο δοχείο, δημιουργούνται δυο ξεχωριστοί πράκτορες και ξεκινάνε στο JADE.

1. Ο πράκτορας διαχείρισης του συστήματος (Agent Management System, AMS) είναι ο πράκτορας που επιβλέπει την πλατφόρμα. Είναι ο συνδετικός κρίκος για όλους του πράκτορες που χρειάζονται να αλληλεπιδράσουν και να χρησιμοποιήσουν τις λευκές σελίδες της πλατφόρμας και για να διαχειριστούν τον κύκλο ζωής τους. Κάθε πράκτορας χρειάζεται να καταχωρηθεί με τον AMS για να μπορέσει να λάβει ένα έγκυρο AID.
2. Ο πράκτορας συντονιστής καταλόγου (Directory Facilitator, DF) είναι ο πράκτορας που υλοποιεί την υπηρεσία κίτρινων σελίδων, που χρησιμοποιείται από άλλους πράκτορες που επιθυμούν να καταχωρήσουν τις υπηρεσίες τους ή να αναζητήσουν διαθέσιμες υπηρεσίες. Επίσης ο DF μπορεί να ειδοποιήσει κάποιον πράκτορα όταν μια υπηρεσία είναι διαθέσιμη ή έχει τροποποιηθεί, με συγκεκριμένα κριτήρια.

3.4. Υπηρεσία μεταφοράς μηνυμάτων

Σύμφωνα με τις προδιαγραφές του FIPA, η υπηρεσία μεταφοράς μηνυμάτων (Message Transport Service, MTS) είναι μία από τις τρεις πιο σημαντικές υπηρεσίες που μια πλατφόρμα πρακτόρων απαιτείται να παρέχει (οι άλλες δυο είναι οι πράκτορες AMS και DF). Μια υπηρεσία μεταφοράς μηνυμάτων διαχειρίζεται όλη την ανταλλαγή μηνυμάτων μέσα στην ίδια πλατφόρμα αλλά και μεταξύ πλατφόρμων.

3.4.1. Πρωτόκολλα μεταφοράς μηνυμάτων

Για να προαχθεί η διαλειτουργικότητα μεταξύ διαφορετικών (π.χ. όχι JADE) πλατφόρμων, το JADE ενσωματώνει όλα τα πρότυπα πρωτόκολλα μεταφοράς μηνυμάτων (Message Transport Protocols, MTPs) που ορίζονται από τον FIPA, όπου κάθε MTP περιλαμβάνει τον ορισμό ενός πρωτοκόλλου μεταφοράς και ένα πρότυπο για να κωδικοποιεί τον φάκελο του μηνύματος.
Το JADE πάντα εκκινεί ένα MTP βασισμένο στο HTTP με την αρχικοποίηση του κύριου δοχείου, ενώ κανένα MTP δεν ενεργοποιείται σε κανονικά δοχεία. Έτσι δημιουργείται ένα server socket στο κύριο δοχείο που φιλοξενεί και περιμένει για εισερχόμενες συνδέσεις μέσω του HTTP για το URL που καθορίζεται. Οποτεδήποτε εγκατασταθεί μια εισερχόμενη σύνδεση και ληφθεί ένα έγκυρο μήνυμα από αυτή τη σύνδεση, το MTP δρομολογεί το μήνυμα στον τελικό του προορισμό, που είναι γενικά ένας πράκτορας λογισμικού που βρίσκεται στην κατανεμημένη πλατφόρμα. Εσωτερικά η πλατφόρμα χρησιμοποιεί ένα proprietary πρωτόκολλο μεταφοράς που ονομάζεται εσωτερικό πρωτόκολλο μεταφοράς μηνυμάτων (Internal Message Transport Protocol, IMTP).

3.4.2. IMTP

Το JADE IMTP χρησιμοποιείται αποκλειστικά για την ανταλλαγή μηνυμάτων μεταξύ πρακτόρων που βρίσκονται σε διαφορετικά δοχεία στην ίδια πλατφόρμα. Το ξεχωρίζουμε από διαπλατφορμικά MTPs όπως το HTTP. Κατά αρχάς επειδή χρησιμοποιείται για επικοινωνία εσωτερικά της πλατφόρμας και μόνο, δεν χρειάζεται να είναι συμβατό με τα πρότυπα του FIPA, και έτσι σχεδιάστηκε με γνώμονα την απόδοση. Το JADE IMTP χρησιμοποιείται στην πραγματικότητα όχι μόνο για να μεταφέρει μηνύματα, αλλά και για να μεταφέρει εσωτερικές εντολές που χρειάζονται για την διαχείριση της κατανεμημένης πλατφόρμας, όπως και να επιτηρεί την

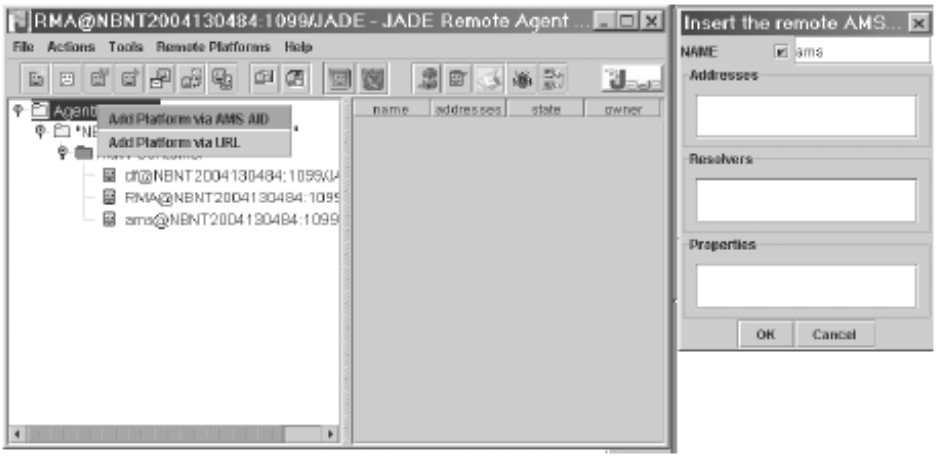
κατάσταση των απομακρυσμένων δοχείων. Για παράδειγμα, χρησιμοποιείται για να μεταφέρει μια εντολή που θα θέσει εκτός λειτουργίας ένα δοχείο, όπως επίσης και για να παρακολουθεί πότε ένα δοχείο τίθεται εκτός λειτουργίας ή γίνεται απροσπέλαστο.
Το JADE σχεδιάστηκε για να επιτρέπει την επιλογή του IMTP την στιγμή της εκκίνησης. Υπάρχουν δύο υλοποιήσεις για το IMTP. Η μία βασίζεται στην Java RMI (Remote Method Invocation) και είναι η προεπιλεγμένη. Η δεύτερη είναι βασισμένη σε ένα proprietary πρωτόκολλο που χρησιμοποιεί TCP sockets και παρακάμπτει την απουσία υποστήριξης της Java RMI στο J2ME περιβάλλον.

3.5. Εργαλεία διαχείρισης και εντοπισμού σφαλμάτων

Οι εφαρμογές με πράκτορες λογισμικού είναι γενικά, αρκετά πολύπλοκες. Πολύ συχνά είναι κατανεμημένες σε πολλά τερματικά και αποτελούνται από πιθανόν εκατοντάδες πολύ-νηματικές διεργασίες. Αυτές οι πλευρές συνεπάγονται δυσκολίες στην διαχείριση και στον εντοπισμό σφαλμάτων. Για να μετριαστεί αυτό, το JADE έχει μια υπηρεσία ειδοποίησης γεγονότων (Event Notification Service, ENS) το οποίο αποτελεί την βάση της κονσόλας διαχείρισης Java RMA (Remote Monitoring Agent) και ένα σύνολο εργαλείων με γραφικό περιβάλλον που παρέχονται για να βοηθήσουν στην διαχείριση και στον εντοπισμό σφαλμάτων.

3.5.1. Η κονσόλα διαχείρισης της πλατφόρμας

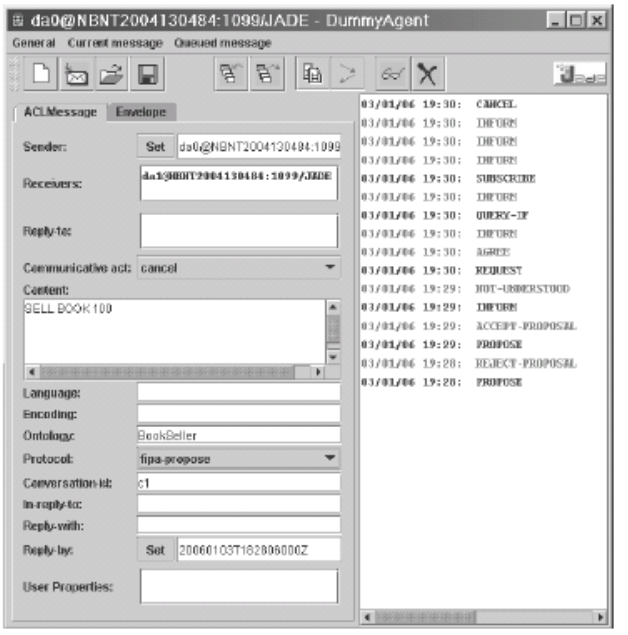
Ο JADE RMA είναι ένα εργαλείο του συστήματος που ενσωματώνει μια κονσόλα γραφικού περιβάλλοντος της πλατφόρμας. Το εργαλείο παρέχει μια γραφική διεπαφή για την παρακολούθηση και την διαχείριση μια κατανεμημένης JADE πλατφόρμας που αποτελείται από ένα ή περισσότερα τερματικά. Περιλαμβάνει ένα μενού 'Tools' διάμεσο του οποίου μπορούν να εκκινήσουν και άλλα εργαλεία. Πολλά RMAs μπορούν να εκκινήσουν στην ίδια πλατφόρμα εφόσον δοθεί διαφορετικό όνομα για τον πράκτορα λογισμικού σε κάθε στιγμιότυπο.
Εφόσον επιλέξουμε ένα πράκτορα μας δίνονται οι εξής επιλογές : διακοπή, συνέχεια, κλωνοποίηση, αποθήκευση. Επίσης επιτρέπει την δημιουργία μηνυμάτων για να αποσταλούν στους πράκτορες λογισμικού.



Εικόνα 2. *Το γραφικό περιβάλλον του JADE RMA*

3.5.2. Ο πλαστός πράκτορας λογισμικού (THE DUMMYAGENT)

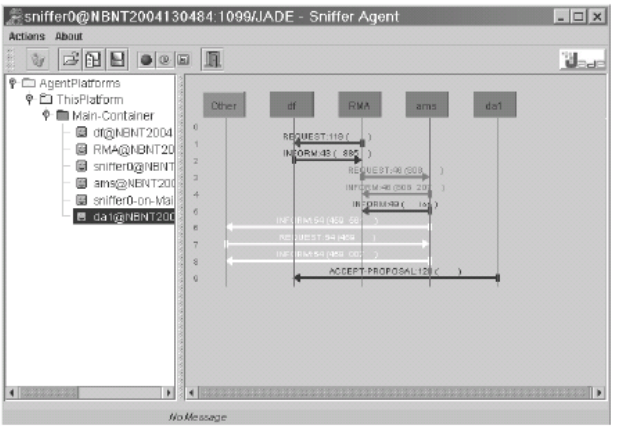
Ο πλαστός πράκτορας είναι ένα πολύ απλό εργαλείο για την αποστολή ερεθισμάτων, στην μορφή προσαρμοσμένων ACL μηνυμάτων, για να δοκιμαστεί η συμπεριφορά άλλων πρακτόρων. Η μόνη του δυνατότητα είναι να στέλνει και να παραλαμβάνει προσαρμοσμένα μηνύματα που μπορούν να δημιουργηθούν χρησιμοποιώντας ένα απλό GUI. Είναι ένα απλό αλλά αποτελεσματικό εργαλείο που χρησιμοποιείται κατά την διάρκεια της ανάπτυξης των προγραμμάτων για να δοκιμαστούν οι συμπεριφορές των πρακτόρων.



Εικόνα 3. *Το γραφικό περιβάλλον του Dummy Agent*

3.5.3. Ο πράκτορας λογισμικού Sniffer

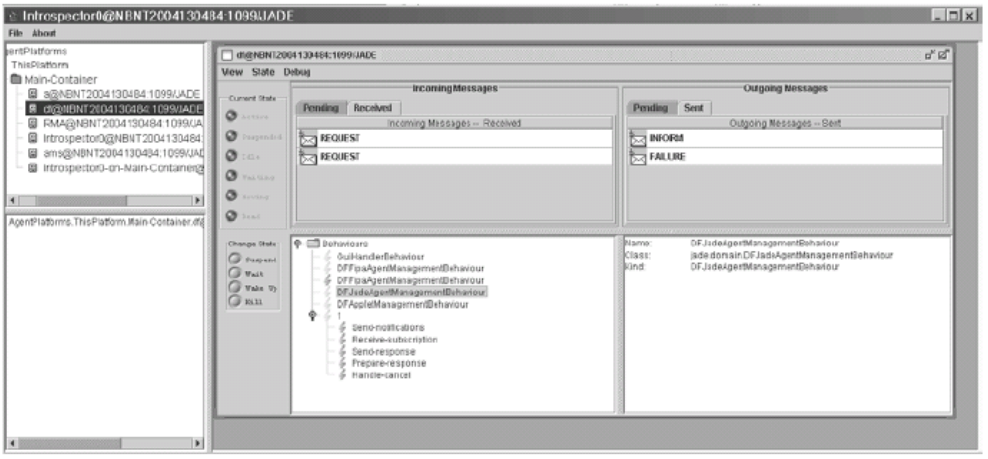
Ενώ όλα τα εργαλεία έχουν σαν σκοπό τον εντοπισμό σφαλμάτων, αυτό το εργαλείο απλά καταγράφει την επικοινωνία μεταξύ των πρακτόρων λογισμικού. Όταν ο χρήστης αποφασίσει να παρακολουθήσει έναν πράκτορα λογισμικού, όλα τα μηνύματα από και προς αυτόν τον πράκτορα καταγράφεται και εμφανίζεται στο GUI του sniffer. Ο χρήστης μπορεί να επιλέξει να δει τις λεπτομέρειες κάθε μηνύματος, να το αποθηκεύσει στον δίσκο σαν αρχείο κειμένου. Πολλά στιγμιότυπα του sniffer μπορούν να δημιουργηθούν από μενού ‘Tools’ ή και από την γραμμή εντολών.



Εικόνα 4. *Το γραφικό περιβάλλον του πράκτορα sniffer*

3.5.4. Ο πράκτορας λογισμικού Introspector

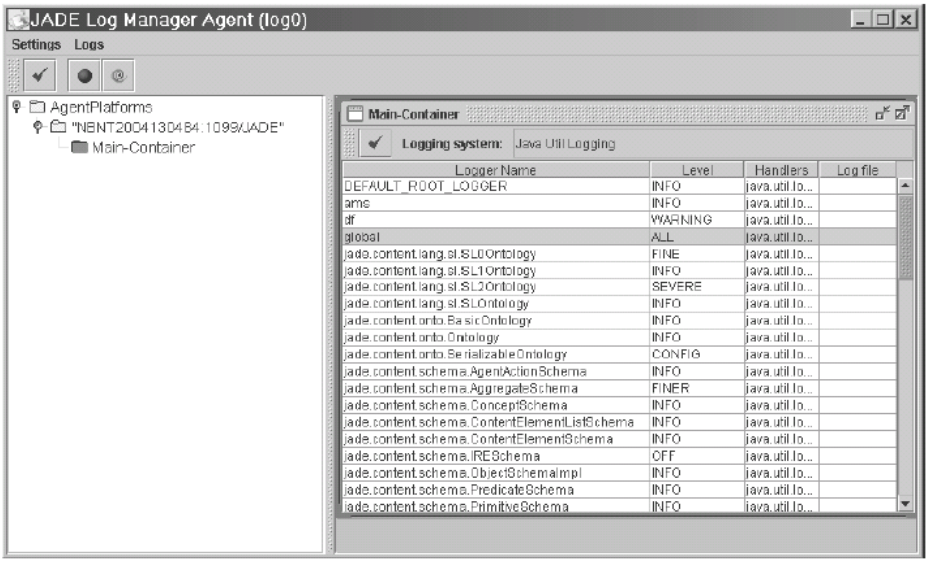
Ενώ ο πράκτορας Sniffer είναι χρήσιμος για να παρακολουθεί και εντοπίζει τα σφάλματα στις συζητήσεις μεταξύ των πρακτόρων, ο πράκτορας Introspector χρησιμοποιείται για να εντοπίσει τα σφάλματα της συμπεριφοράς ενός πράκτορα. Αυτό το εργαλείο στην πραγματικότητα επιτρέπει την παρακολούθηση και τον έλεγχο του κύκλου ζωής ενός πράκτορα, και των λιστών με τα μηνύματα τους. Το συγκεκριμένο εργαλείο επιτρέπει στον χρήστη να δει ποιες συμπεριφορές εκτελούνται, ποιες είναι σε αναμονή και να παρακολουθήσει τις αντιδράσεις τους σε εξωτερικά ερεθίσματα, δηλαδή εισερχόμενα μηνύματα.
Η εικόνα 5, δείχνει το GUI του πράκτορα Introspector, όταν παρακολουθεί τον πράκτορα DF.



Εικόνα 5. *Το γραφικό περιβάλλον του πράκτορα Introspector*

3.5.5. Ο πράκτορας διαχειριστής των καταγραφών

Ο πράκτορας διαχειριστής των καταγραφών (Log Manager Agent) είναι ένα εργαλείο που απλοποιεί την δυναμική και κατανεμημένη διαχείριση των καταγραφών παρέχοντας μια γραφική διεπαφή που επιτρέπει να αλλαχθούν τα επίπεδα καταγραφής κάθε συστατικού της πλατφόρμας JADE στην διάρκεια εκτέλεσης. Κάθε αντικείμενο της κλάσης Logger μπορεί να παραμετροποιηθεί με το δικό του επίπεδο καταγραφής.



Εικόνα 6. *Το GUI του πράκτορα διαχειριστή των καταγραφών*

3.6. Δημιουργία πρακτόρων με το JADE

Η δημιουργία ενός πράκτορα λογισμικού είναι πολύ απλή. Ουσιαστικά χρειάζεται μια κλάση που να επεκτείνει την κλάση jade.core.Agent και χρησιμοποιώντας την μέθοδο setup () όπως γίνεται με το ακόλουθο παράδειγμα:

```
import jade.core.Agent;
public class HelloWorldAgent extends Agent {
  protected void setup() {
  // Printout a welcome message
  System.out.println("Hello World. I'm an agent!");
  }
}
```

Πιο συγκεκριμένα μια κλάση, όπως η HelloWorldAgent που βλέπουμε παραπάνω, αναπαριστά ένα τύπο πράκτορα όπως μια κανονική Java κλάση αναπαριστά ένα τύπο αντικειμένου. Πολλά στιγμιότυπα της κλάσης HelloWorldAgent μπορούν να εκκινήσουν στον χρόνο εκτέλεσης. Αντίθετα με τα αντικείμενα της Java, που χειρίζονται με τις κλήσεις τους, ένας πράκτορας εκκινεί πάντα στον χρόνο εκτέλεσης του JADE και η κλήση του δεν αποκαλύπτεται ποτέ εκτός του πράκτορα. Οι πράκτορες δεν επικοινωνούν με κλήση μεθόδων αλλά με την ανταλλαγή ασύγχρονων μηνυμάτων.

3.7. Αναγνωριστικά πρακτόρων

Σύμφωνα με της προδιαγραφές του FIPA, κάθε στιγμιότυπο πράκτορα ξεχωρίζεται με ένα αναγνωριστικό πράκτορα. Στο JADE ένα αναγνωριστικό αναπαριστάται σαν στιγμιότυπο της κλάσης jade.core.AID. Η μέθοδος getAID() της κλάσης Agent επιτρέπει την λήψη του τοπικού αναγνωριστικού του πράκτορα. Ένα αντικείμενο AID περιλαμβάνει ένα παγκόσμια μοναδικό όνομα GUID (Globally Unique Identifier, GUID). Η μορφή του ονόματος στο JADE είναι <local-name>@<platform-name>, έτσι ώστε ένας πράκτορας που ονομάζεται Peter και ζει στην πλατφόρμα foo-platform να έχει σαν GUID το Peter@foo-platform. Οι διευθύνσεις που

περιλαμβάνονται στην κλάση AID είναι οι διευθύνσεις της πλατφόρμας που κατοικεί ο πράκτορας. Αυτές οι διευθύνσεις χρησιμοποιούνται μόνο αν ένας πράκτορας πρέπει να επικοινωνήσει με κάποιον πράκτορα που ζει σε διαφορετική πλατφόρμα.
Η κλάση AID παρέχει μεθόδους για ανάκτηση του τοπικού ονόματος (getLocalName()), το GUID (getName()) και τις διευθύνσεις (getAllAdrdresses()).

3.8. Εργασίες των πρακτόρων

Όπως αναφέρθηκε και προηγουμένως, η πραγματική δουλεία, ή δουλειές, που έχει να εκτελέσει ένας πράκτορας γίνεται μέσα σε 'συμπεριφορές'. Η συμπεριφορά αναπαριστά μια εργασία που ένας πράκτορας μπορεί να διενεργήσει και μπορεί να υλοποιηθεί σαν ένα αντικείμενο που επεκτείνει την κλάση jade.core.behaviours.Behaviour. Για να εκτελέσει ο πράκτορας την εργασία που ενσωματώνεται από ένα αντικείμενο συμπεριφοράς, η συμπεριφορά πρέπει να προστεθεί με την μέθοδο addBehaviour() της κλάσης Agent. Κάθε κλάση που επεκτείνει την κλάση Behaviour πρέπει να περιλαμβάνει δυο μεθόδους. Την μέθοδο action() που ορίζει της λειτουργίες που θα πραγματοποιηθούν όταν η συμπεριφορά εκτελεστεί. Η μέθοδος done() επιστρέφει μια τιμή Boolean για να δείξει ότι μια συμπεριφορά έχει ολοκληρωθεί ή όχι και είναι σε θέση να αφαιρεθεί από τον χώρο που εκτελούνται οι συμπεριφορές ενός πράκτορα.

3.8.1. Προγραμματισμός και εκτέλεση συμπεριφορών

Ένας πράκτορας μπορεί να εκτελέσει πολλές συμπεριφορές ταυτόχρονα. Ωστόσο, είναι σημαντικό να σημειωθεί ότι ο προγραμματισμός των συμπεριφορών σε ένα πράκτορα είναι συνεργατικός. Αυτό σημαίνει όταν μία συμπεριφορά προγραμματίζεται για εκτέλεση καλείται η μέθοδος action() και τρέχει μέχρι να επιστρέψει κάτι. Έτσι ο προγραμματιστής είναι αυτός που καθορίζει πότε ένας πράκτορας μεταβαίνει από την εκτέλεση μιας συμπεριφοράς σε μια άλλη. Αυτή η πρακτική έχει τα ακόλουθα πλεονεκτήματα:

- Επιτρέπει ένα νήμα (thread) Java για κάθε πράκτορα το οποίο είναι πολύ σημαντικό για περιβάλλοντα με περιορισμένου περιοριστικού πόρου όπως τα κινητά τηλέφωνα.
- Παρέχεται αυξημένη απόδοση αφού η μετάβαση μεταξύ συμπεριφορών είναι πιο γρήγορη από την μετάβαση μεταξύ νημάτων Java.
- Εξαφανίζει τα θέματα συγχρονισμού μεταξύ ταυτόχρονων συμπεριφορών που χρησιμοποιούν τους ίδιους πόρους αφού όλες οι συμπεριφορές εκτελούνται από το ίδιο νήμα Java.

3.8.2. Συμπεριφορές μίας εκτέλεσης, κυκλικές και γενικές

Υπάρχουν τρεις κύριοι τύποι συμπεριφορών διαθέσιμοι στο JADE:

1. Οι συμπεριφορές μιας εκτέλεσης σχεδιάζονται για να ολοκληρωθούν σε μία φάση εκτέλεσης, έτσι η μέθοδος action() εκτελείται μια φορά.

```
public class MyOneShotBehaviour extends OneShotBehaviour {
 public void action() {
  // perform operation X
  }
 }
```

   Η λειτουργία X εκτελείται μόνο μια φορά.

2. Οι κυκλικές συμπεριφορές είναι σχεδιασμένες για να μην τελειώνουν, αφού η action μέθοδος τους εκτελείται κάθε φορά που καλείται.

```
public class MyCyclicBehaviour extends CyclicBehaviour {
 public void action() {
  // perform operation Y
 }
}
```

Η λειτουργία Y εκτελείται επαναλαμβανόμενα μέχρι ο πράκτορας που εκτελεί την συμπεριφορά να τερματιστεί.

3. Οι γενικές συμπεριφορές ενσωματώνουν μια κατάσταση εκκίνησης και εκτελούν διαφορετικές λειτουργίες ανάλογα με την τιμή της κατάστασης. Ολοκληρώνονται όταν μια συγκεκριμένη συνθήκη υλοποιηθεί.

```
public class ThreeStepBehaviour extends Behaviour {
private int step = 0;
 public void action() {
  switch (step) {
  case 0:
  // perform operation X
  step++;
  break;
  case 1:
  // perform operation Y
  step++;
  break;
  case 2:
  // perform operation Z
  step++;
  break;
  }
 }
 public boolean done() {
  return step == 3;
 }
}
```

Σε αυτό το παράδειγμα, το step είναι μεταβλητή που αντιπροσωπεύει την κατάσταση της συμπεριφοράς. Οι λειτουργίες X,Y και Z εκτελούνται σειριακά με την ολοκλήρωση των οποίων η συμπεριφορά ολοκληρώνεται.

3.8.3. Προγραμματισμός εργασιών

Το JADE παρέχει δυο έτοιμες κλάσεις (στο πακέτο jade.core.behaviours) που μπορούν να χρησιμοποιηθούν για να παράγουν συμπεριφορές σε συγκεκριμένες στιγμές στον χρόνο.

1. Η WakerBehaviour έχει υλοποιημένες τις μεθόδους action() και done() για να εκτελέσει την μέθοδο onWake() μετά από συγκεκριμένο χρόνο (timeout) που ορίζεται.

```
public class MyAgent extends Agent {
  protected void setup() {
   System.out.println("Adding waker behaviour");
    addBehaviour(new WakerBehaviour(this, 10000) {
     protected void onWake() {
     // perform operation X
     }
    } );
   }
  }
```

Στο συγκεκριμένο παράδειγμα, η λειτουργία X εκτελείται 10 δευτερόλεπτα μετά την εκτύπωση του κειμένου ‘Adding waker behaviour’.

2. Η TickerBeaviour έχει υλοποιημένες τις μεθόδους action() και done() για να εκτελέσει την μέθοδο onTick() επαναλαμβανόμενα, αναμένοντας μια χρονική περίοδο μέχρι την επόμενη εκτέλεση. Μια TickerBehaviour ποτέ δεν ολοκληρώνεται εκτός αν αφαιρεθεί ή γίνει κλήση της μεθόδου stop().

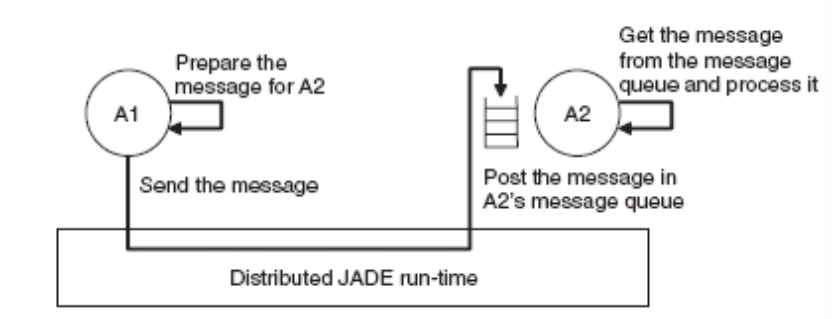
```
public class MyAgent extends Agent {
 protected void setup() {
  addBehaviour(new TickerBehaviour(this, 10000) {
   protected void onTick() {
    // perform operation Y
   }
  } );
 }
}
```

Στο συγκεκριμένο παράδειγμα η λειτουργία Υ εκτελείται κάθε 10 δευτερόλεπτα.

3.9. Επικοινωνία πρακτόρων

Η επικοινωνία μεταξύ των πρακτόρων είναι το πιο σημαντικό χαρακτηριστικό που υποστηρίζεται στο JADE και υλοποιείται σε συμφωνία με τις προδιαγραφές του FIPA. Το μοντέλο της επικοινωνίας βασίζεται στην ασύγχρονη μετάδοση μηνυμάτων. Κατά συνέπεια, κάθε πράκτορας έχει ένα 'mailbox' (η λίστα μηνυμάτων του πράκτορα) όπου το τα μηνύματα από τους άλλους πράκτορες αποθηκεύονται στον χρόνο εκτέλεσης. Όταν ένα μήνυμα αποθηκεύεται στην λίστα μηνυμάτων ο παραλήπτης-πράκτορας ειδοποιείται. Το πρότυπο που ακολουθείται για τα μηνύματα είναι το FIPA-ACL. Κάθε μήνυμα περιέχει τα ακόλουθα πεδία:

- Ο αποστολέας του μηνύματος
- Την λίστα των παραληπτών
- Η επικοινωνιακή πράξη υποδηλώνει τι θέλει να πετύχει με την αποστολή του μηνύματος. Εάν η πράξη είναι τύπου REQUEST, ο αποστολέας θέλει ο παραλήπτης να κάνει κάποια ενέργεια, εάν είναι INFORM ο αποστολέας θέλει να έχει γνώση ο παραλήπτης για ένα γεγονός, και αν είναι PROPOSE, ο αποστολέας θέλει να μπει σε μια διαπραγμάτευση.



Σχήμα 14. *Το μοντέλο ανταλλαγής μηνυμάτων του JADE*

- Το περιεχόμενο που περιέχει την πραγματική πληροφορία που ανταλλάσετε με το μήνυμα.
- Η γλώσσα του περιεχομένου του μηνύματος υποδηλώνει το συντακτικό που χρησιμοποιείται για να εκφραστεί το περιεχόμενο. Και ο παραλήπτης και ο αποστολέας πρέπει να κωδικοποιήσουν το περιεχόμενο στο ίδιο συντακτικό για να γίνει σωστά η επικοινωνία.

- Η οντολογία υποδηλώνει το λεξιλόγιο των συμβόλων που χρησιμοποιείται στο περιεχόμενο.
- Μερικά επιπρόσθετα πεδία για να συνεχιστούν πολύπλοκες ανταλλαγές μηνυμάτων όπως το αναγνωριστικό της συζήτησης.

Ένα μήνυμα στο JADE υλοποιείται ως ένα αντικείμενο της κλάσης jade.lang.acl.ACLMessage που παρέχει μεθόδους set και get για να υπάρχει πρόσβαση στα πεδία του μηνύματος.

3.9.1. Αποστολή μηνυμάτων

Η αποστολή ενός μηνύματος σε ένα άλλο πράκτορα είναι πολύ απλή και το μόνο που χρειάζεται είναι γεμίσει τα πεδία ενός αντικειμένου ACLMessage και μετά κλήση της μεθόδου send() της κλάσης Agent. Παρακάτω ακολουθεί ένα παράδειγμα δημιουργίας ενός μηνύματος για να ενημερώσει τον πράκτορα Peter ότι σήμερα βρέχει:

```
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Peter", AID.ISLOCALNAME));
msg.setLanguage("English");
msg.setOntology("Weather-forecast-ontology");
msg.setContent("Today it's raining");
send(msg);
```

3.9.2. Παραλαβή μηνυμάτων

Όπως αναφέρθηκε και προηγουμένως, το JADE αποθηκεύει τα μηνύματα στην λίστα μηνυμάτων του παραλήπτη μόλις φτάσουν. Ένας πράκτορας μπορεί να παραλάβει με την μέθοδο receive(). Αυτή η μέθοδος επιστρέφει το πρώτο μήνυμα στην λίστα μηνυμάτων (αφαιρώντας από την λίστα), ή null εάν η λίστα είναι άδεια, και τερματίζει.

```
ACLMessage msg = receive();
if (msg != null) {
// Process the message
}
```

Παρόλα αυτά ο τρόπος γραφής για την παραλαβή μηνυμάτων εντός μια συμπεριφοράς διαφοροποιείται ελάχιστα για την εξοικονόμηση υπολογιστικής ισχύος άρα και ενέργειας, χρησιμοποιώντας την μέθοδο block() για την περίπτωση που δεν υπάρχει νέο μήνυμα, αφού όπως είπαμε υπάρχουν και συμπεριφορές που επαναλαμβάνονται ανά κάποιο χρονικό διάστημα :

```
public void action() {
 ACLMessage msg = myAgent.receive();
  if (msg != null) {
   // Message received. Process it
   ...
   }
  else {
   block();
  }
}
```

3.9.3. Επιλογή μηνυμάτων από την λίστα του παραλήπτη

Πολλές φορές (κυρίως σε κυκλικές συμπεριφορές) παρατηρείται το φαινόμενο η παραλαβή ενός μηνύματος σε λάθος σημείο του κώδικα μας. Αυτό γίνεται επειδή η μέθοδος receive() παραλαμβάνει το πρώτο μήνυμα που βρίσκεται στην λίστα του παραλήπτη. Για να αποφύγουμε αυτό μπορούμε να χρησιμοποιήσουμε κάποιο πρότυπο σαν όρισμα στην μέθοδο receive() για να ξεχωρίσουμε ουσιαστικά τα μηνύματα που θα παραληφθούν. Τα πρότυπα αυτά είναι ουσιαστικά στιγμιότυπα της κλάσης jade.lang.acl.MessageTemplate. Παρακάτω ακολουθεί ένα παράδειγμα όπου χρησιμοποιείται ένα πρότυπο για να παραλαμβάνονται μόνο τα μήνυμα τύπου CFP (Call For Proposal) και τα υπόλοιπα να αγνοούνται [17].

```
private MessageTemplate mt=
MessageTemplate.MatchPerformative(ACLMessage.CFP);
public void action() {
  ACLMessage msg = myAgent.receive(mt);
   if (msg != null) {
   // CFP Message received. Process it
   ...
   }
   else {
    block();
   }
}
```

4. Υλοποίηση

Στο κεφάλαιο αυτό θα αναλύσουμε το πρακτικό μέρος της εργασίας. Η εργασία αφόρα την υλοποίηση σε προσομοίωση του πρωτοκόλλου αυθεντικοποίησης για RFID ετικέτες με πράκτορες λογισμικού και συγκεκριμένα με την χρήση της πλατφόρμας JADE, γιατί όπως είδαμε παρέχει αρκετές δυνατότητες και ευκολία δημιουργίας προς τον χρήστη. Επειδή το JADE έχει αναπτυχθεί εδώ και πολλά χρόνια σαν ανοιχτό λογισμικό με πολύ μεγάλες εταιρίες να το υποστηρίζουν.

4.1. Σύνδεση JADE με NetBeans

Επειδή είναι χρήσιμο να διασυνδέεται το JADE με το NetBeans για πιο γρήγορο debugging και εκτέλεση των προγραμμάτων (κλάσεων) από το Netbeans στο JADE RMA, ακολουθεί ένας οδηγός για να δουλέψουν όλα μαζί.

1. Εγκατάσταση του NetBeans
2. Κατεβάζουμε το JADE από το http://jade.tilab.com/. Κάνουμε extract τα περιεχόμενα των αρχείων .zip σε ένα φάκελο π.χ C:\ JADE
3. Δημιουργούμε ένα project στο NetBeans
4. Στο NetBeans πηγαίνουμε στις επιλογές Tools->Libraries
5. Πατάμε το New Library… ονομάζουμε την βιβλιοθήκη μας JADE
6. Πατάμε την επιλογή Add JAR/Folder… και προσθέτετε τα 2 jar αρχεία από το τα directory που κάνατε extract to JADE αν εγίνε στο C:\ JADE τότε τα jar θα βρίσκονται στο **C:\JADE\lib\jade.jar** και **C:\JADE\lib\commons-codec\commons-codec-1.3.jar**

7. Στο project που δημιουργήσαμε πατάμε δεξί κλικ Set Configuration, Customize…
8. Στην κατηγορία Run και εκεί που γράφει Main Class βάζουμε **jade.Boot** και στα arguments **-gui**
9. Στο project μας και συγκεκριμένα στο Libraries πατάμε δεξί κλικ και μετά Add Library μετά επιλέγουμε την JADE, που είχαμε δημιουργήσει στο βήμα 5.
10. Το ίδιο κάνουμε και για το Test Libraries στο project μας [18].

Για χρήστες που επιθυμούν εκτέλεση μέσω γραμμής εντολών μπορούν να ανατρέξουν στην ιστοσελίδα του JADE http://jade.tilab.com/ και να βρουν τις αντίστοιχες οδηγίες.

4.2. Αρχιτεκτονική υλοποίησης

Με το πρωτόκολλο για τα RFID που αναλύσαμε σε προηγούμενο κεφάλαιο το πρώτο πρέπει να ορίσουμε είναι οι οντότητες. Στην περίπτωση μας, ορίζουμε 3 οντότητες:

1. Tag, θα είναι η οντότητα που θα αντιπροσωπεύει την ετικέτα. Σύμφωνα με το πρωτόκολλο η ετικέτα πρέπει να υποστηρίζει μια ψευδοτυχαία γεννήτρια αριθμών και μία μονόδρομη συνάρτηση hash.
2. Reader, είναι η οντότητα που θα αντιπροσωπεύει τον αναγνώστη και το repository μαζί. Θα μπορούσαν να διαχωριστούν οι δυο οντότητες αλλά επειδή ο αναγνώστης αν ήταν μια οντότητα δεν θα έκανε τίποτα περισσότερο από το να μεταφέρει τα μηνύματα μεταξύ των ετικετών και του repository, θεωρήθηκε ορθότερο να είναι μια οντότητα. Το Repository ουσιαστικά είναι ένας δυσδιάστατος πίνακας με προκαθορισμένο μήκος, στην προσομοίωση μας τριάντα (κάθε γραμμή είναι τα στοιχεία για κάθε ετικέτα) αλλά μπορεί να αυξηθεί αλλάζοντας τον αριθμό στο πρόγραμμα του Reader. Επίσης θα πρέπει να σημειωθεί ότι το όνομα της οντότητας που υλοποιεί την κλάση Reader.RfidAgent πρέπει να είναι προκαθορισμένο. Στην υλοποίηση μας έχουμε δηλώσει ότι είναι READER.
3. DReader, είναι η οντότητα που θα αντιπροσωπεύει τον προσωρινό χρήστη μιας ετικέτας, που του έχουν μεταβιβαστεί τα δικαιώματα ανάγνωσης από τον ιδιοκτήτη της ετικέτας.

Οι παραπάνω οντότητες παίρνουν το όνομα τους από το Java Package, μέσα σε αυτό περιέχονται όλες οι βοηθητικές κλάσεις (π.χ. RNG, Hash) που καλούνται στο κύριο πρόγραμμα που ονομάζεται RfidAgent και περιέχει την μέθοδο setup(). Παρότι το όνομα των κύριων προγραμμάτων είναι ίδιο και για τα τρία Packages, αυτά ξεχωρίζουν από το όνομα του package στο Jade RMA GUI.

4.2.1. Ψευδοτύχαια γεννήτρια (κλάση RNG)

Η ψευδοτυχαία γεννήτρια που χρησιμοποιήθηκε ονομάζεται RngPack μπορεί κάποιος να την κατεβάσει από εδώ http://www.honeylocust.com/RngPack/. Το RngPack είναι ένα πακέτο ψευδοτυχαίων γεννητριών για Java του Paul Houle. Ψευδοτυχαίο σημαίνει ότι οι τυχαίοι αριθμοί παράγονται με ντετερμινιστικό μαθηματικό μοντέλο. Το RngPack περιέχει βασικές κλάσεις για να προσθέσουν αξία στις γεννήτριες, τέσσερις γεννήτριες ερευνητικού επιπέδου και μία γεννήτρια της Java (τη random). Επιπρόσθετα ξεχωριστά από το RngPack έχει υλοποιηθεί και η secure random της Java που παράγει τυχαίους αριθμούς και προτείνεται για όλων των τύπων

κρυπτογραφικές συναρτήσεις με υψηλής τυχαιότητας αριθμούς. Η κλάση RNG στο πρόγραμμα μας υλοποιεί έξι μεθόδους, μια για κάθε γεννήτρια του RngPack, και μια για την secure random. Έτσι ο χρήστης έχει την επιλογή της γεννήτριας ανάλογα με την κλήση της μεθόδου που θα επιθυμήσει. Οι αριθμοί που παράγονται από τις γεννήτριες μπαίνουν σαν είσοδο στην συνάρτηση hash και προκύπτει και το αποτέλεσμα που προκύπτει είναι ένας δεκαεξαδικός 40 ψηφίων (160bits). Αυτό αυξάνει σίγουρα το υπολογιστικό φορτίο που χρειάζεται για να παραχθούν τα μυστικά αλλά ήταν αναγκαίο γιατί πρέπει όλα τα μυστικά να έχουν τα ίδιο μήκος με τα ψευδώνυμα και τα TID. Η secure random παράγει κατευθείαν 40 ψηφία δεκαεξαδικού. Όλοι οι πηγαίοι κώδικες παρατίθενται στο παράρτημα.

4.2.2. Μονόδρομη hash συνάρτηση (κλάση AeSimpleSHA1)

Η συνάρτηση hash που χρησιμοποιήθηκε είναι είναι η SHA-1 (Secure Hash Algorithm-1) που παράγει σύνοψη 160bits. Η κλάση SHA1 δέχεται σαν είσοδο ένα String και παράγει έναν byte[] 20 θέσεων και μετά με την μέθοδο convertToHex το μετατρέπει σε 40 ψηφία δεκαεξαδικού. Ο SHA-1 θεωρείται υπολογιστικά ανέφικτος για τις ετικέτες RFID, παρόλα αυτά στα πλαίσια της προσομοίωσης το δεχόμαστε εφόσον δεν υπάρχει κάποια χαμηλού υπολογιστικού κόστους hash στην βιβλιοθήκη java.security.

4.2.3. Κλάση Converter

Η κλάση Converter είναι μια βοηθητική συνάρτηση με πέντε μεθόδους με μετασχηματισμούς πολύ χρήσιμους για την ανανέωση μυστικού και γενικά για όλο το πρόγραμμα:

- Η hexStringToByteArray δέχεται σαν όρισμα έναν String και το μετατρέπει σε byte[]
- Η getHexString, δέχεται σαν όρισμα ένα byte[] και το μετατρέπει σε δεκαεξαδικό String
- Η xorByteArray, δέχεται σαν όρισμα δυο byte[] και επιστρέφει έναν byte[] που είναι το XOR μεταξύ των δύο πρώτων.
- Η concatThreeStrings, δέχεται σαν όρισμα τρία String και επιστρέφει ένα string που είναι η συνένωση των τριών.
- Η concatTwoStrings, είναι αντίστοιχη με την concatThreeStrings απλά δέχεται δυο ορίσματα.

4.2.4. Κλάση Horizon

Η κλάση Horizon είναι βοηθητική κλάση και υπάρχει μόνο στο package του reader και έχει δυο μεθόδους την setHorizon, που δίνει ένα μενού στον χρήστη για να δώσει το χρονικό ορίζοντα που θέλει (χρησιμοποιείται στην ανανέωση του χρονικού ορίζοντα) και την getHorizon που επιστρέφει την ημερομηνία από το ημερολόγιο του συστήματος. Να σημειωθεί ότι και οι δυο μέθοδοι επιστρέφουν την ημερομηνία σε μορφή σύμφωνα με το ISO 8601, κάτι που εξυπηρετεί στις συγκρίσεις των χρονικών τιμών που απαιτείται από το πρωτόκολλο.

4.2.5. Κλάση DelTID

Η κλάση DelTID είναι μια κλάση που υπάρχει αποκλειστικά στο package του Reader και είναι πολύ σημαντική για την συμπεριφορά της μεταβίβασης δικαιωμάτων. Δέχεται σαν όρισμα το όνομα του προσωρινού χρήστη (DRepId) και το Rep[][] , το repository ουσιαστικά του reader. Δίνει ένα μενού στον ιδιοκτήτη της ετικέτας που τον ρωτάει ποια ετικέτα θέλει να μεταβιβάσει σε προσωρινό χρήστη και για πόσες μέρες. Και με τον κατάλληλο κώδικα επιστέφει έναν String [][] που κάθε γραμμή είναι για κάθε μέρα της μεταβίβασης. Η κάθε γραμμή αποτελείται από το πρώτο πεδίο το τοπικό όνομα της ετικέτας, στο δεύτερο το d_time που ουσιαστικά είναι η ημερομηνία που χρειάζεται για την παραγωγή του TID και στο τρίτο το TID_{d_time} .

4.3. Αλληλεπίδραση χρήστη με JADE RMA (Remote Management Agent) GUI

Στο σχεδιασμό του πρωτοκόλλου μπορεί να υπάρξει επικοινωνία μεταξύ του χρήστη με τους πράκτορες μέσω του JADE RMA GUI που χρειάζεται για να ενεργοποιηθεί η αντίστοιχη συμπεριφορά. Πιο συγκεκριμένα ενεργοποιούμε τις ακόλουθες συμπεριφορές:

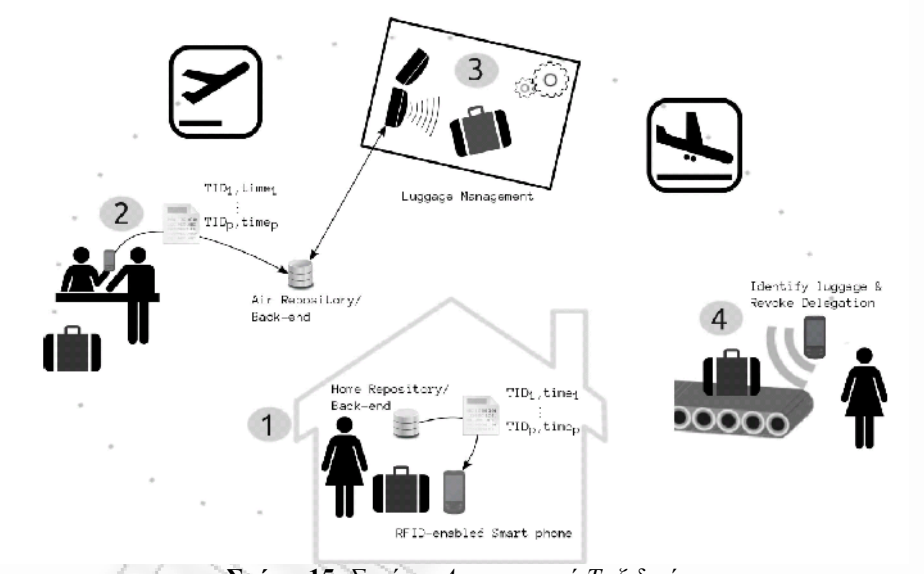
- **Αυθεντικοποίηση:** Αποστολή μηνύματος τύπου CFP (Call For Proposal) στον Reader και στο περιεχόμενο (content) του μηνύματος το TagInstanceId, ουσιαστικά πρόκειται για το local name του JADE που δίνει ο χρήστης κατά την δημιουργία του αντίστοιχου agent.Δηλαδή αν ένας agent που υλοποιεί την κλάση Tag.RfidAgent ονομάστηκε TAG, το TAG πρέπει να γραφτεί στο περιεχόμενο του μηνύματος για να αυθεντικοποιηθεί.
- **Ανανέωση μυστικού ή του χρονικού ορίζοντα:** Αποστολή μηνύματος τύπου Propose στον Reader στο περιεχόμενο την τιμή του TagInstanceId, όπως ακριβώς και στην συμπεριφορά της αυθεντικοποίησης. Το συγκεκριμένο μήνυμα ενεργοποιεί ένα μενού επιλογών στο οποίο ο χρήστης μπορεί να επιλέξει αν θα αναβαθμίσει το μυστικό, ή τον χρονικό ορίζοντα, ή και τα δύο.
- **Μεταβίβαση δικαιώματος ανάγνωσης:** Για να έχει κάποιος άλλος Reader (Delegation Reader), δικαίωμα ανάγνωσης θα πρέπει να επικοινωνήσει με τον Reader/Repository του ιδιοκτήτη αναφέροντας του το όνομα του στο περιεχόμενο του μηνύματος. Αυτό γίνεται με την αποστολή μηνύματος τύπου Proxy προς τον Reader (ιδιοκτήτη).
- **Αυθεντικοποίηση στον Delegation Reader :** Εφόσον ο Delegation Reader έχει λάβει τις τιμές που απαιτούνται από τον Reader μπορεί να εκκινήσει την διαδικασία της αυθεντικοποίησης με το Tag που επέλεξε. Αυτό γίνεται με την αποστολή μηνύματος τύπου Agree στον delegation Reader.
- **Διαγραφή στοιχείων από το Repository του Reader :** Ο Reader μπορεί να διαγράψει από το Repository του στοιχεία που πλέον δεν χρειάζεται. Αυτό γίνεται με την αποστολή μηνύματος τύπου FAILURE και στο περιεχόμενο το TagInstanceId.
- **Διαγραφή στοιχείων από το Repository του Delegation Reader :** Ο Delegation Reader μπορεί να διαγράψει από το Repository του στοιχεία που πλέον δεν χρειάζεται. Αυτό γίνεται με την αποστολή μηνύματος τύπου REFUSE και στο περιεχόμενο το TagInstanceId.

4.4. Σενάριο Αεροπορικού Ταξιδιού

Για να εξηγήσουμε καλύτερα το πρωτόκολλο και να κάνουμε μια επίδειξη του πρακτικού μέρους μέσα από το JADE, θα κάνουμε ένα σενάριο εμπνευσμένοι από την μελέτη του ENISA's Flying 2.0? [19].

4.4.1. Περιγραφή σεναρίου

Η Αλίκη είναι μια κοπέλα που της αρέσει η τεχνολογία, και χρησιμοποιεί RFID ετικέτες για την διαχείριση των πραγμάτων της, τόσο εντός όσο και εκτός σπιτιού. Στο σπίτι της έχει ένα Back-end/ Repository σύστημα που επικοινωνεί με τις ετικέτες της με αναγνώστες RFID. Το κινητό της τηλέφωνο είναι και αυτό ένας αναγνώστης RFID. Η Αλίκη ετοιμάζει της βαλίτσες της επειδή πρόκειται να μεταβεί σε ένα συνέδριο ασφάλειας πληροφοριών για μια παρουσίαση. Πλέον η τεχνολογία RFID κάνει ακόμα και το πακετάρισμα πιο εύκολο αφού μπορεί να συλλέξει μια λίστα με τα απαραίτητα που θέλει να πάρει μαζί της και να γνωρίζει την ακριβή ποσότητα του εκάστοτε είδους.

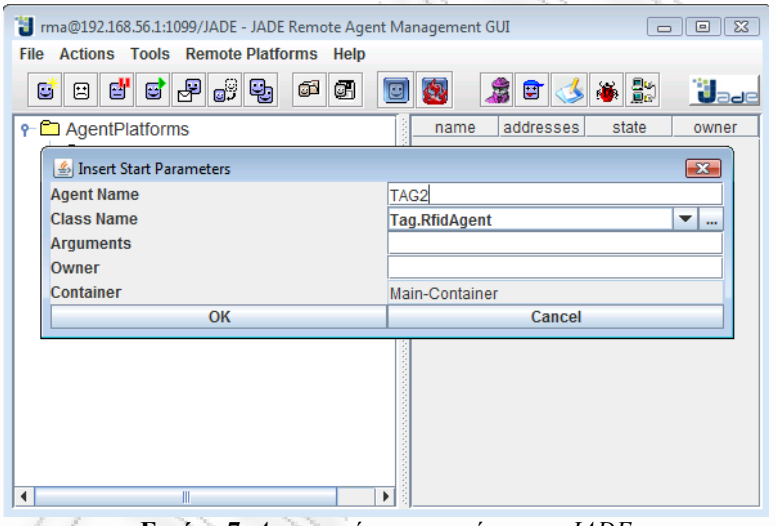


Σχήμα 15. *Σενάριο Αεροπορικού Ταξιδιού*

4.4.2. Βήμα 1: Αρχικοποίηση

Όσο η Αλίκη βρίσκεται στο σπίτι, νιώθει ασφαλής και έχει της ελάχιστες δυνατές απαιτήσεις ασφάλειας και ιδιωτικότητας. Για την διαχείριση των ετικετών, οι RFID αναγνώστες της είναι εξουσιοδοτημένη για να έχουν πρόσβαση στο Repository και για να κάνουν αιτήματα προς τις ετικέτες. Η εξουσιοδότηση χορηγείται μέσω κατάλληλα διαμορφωμένων πολιτικών στο Repository που αξιολογεί τα διαπιστευτήρια των αναγνωστών. Σαν μέτρο προφύλαξης οι αναγνώστες έχουν περιορισμένη πρόσβαση ή και καθόλου στο internet. Τα αιτήματα ετικέτας γίνονται με το πρωτόκολλο αιτήματος ετικέτας αλλά για να επιταχύνει τις διαδικασίες έχει προϋπολογισμένα TID από τα σχετικά προϊόντα, σε αυτήν την περίπτωση ρούχα και άλλα αντικείμενα που θέλει να πάρει μαζί της. Εξάλλου, το Repository αξιοποιεί της πληροφορίες θέσης (π.χ. δωμάτιο) του αναγνώστη και περιορίζει τον χώρο αναζήτησης των ετικετών. Όταν η Αλίκη τελειώσει με το πακετάρισμα ένας τελευταίος έλεγχος πραγματοποιείται για δει ότι δεν ξέχασε τίποτα. Σε αυτό το σημείο επιλέγει να ελαχιστοποιήσει των θόρυβο των ετικετών (δηλαδή της

αποκρίσεις των ετικετών σε άγνωστους αναγνώστες), και βάζει σε κατάσταση αναμονής όλες τις ετικέτες της βαλίτσας. Η Αλίκη όταν έκλεισε τα εισιτήρια δεν ήξερε πόσες αποσκευές θα πάρει μαζί της έτσι προτίμησε να κάνει καταχώρηση των αποσκευών στο αεροδρόμιο. Κατά συνέπεια ζητάει από το Home Rep μεταφέρει όλες τις πληροφορίες για την βαλίτσας στο τηλέφωνο της που περιέχει το μυστικό, τον ορίζοντα και το ID της βαλίτσας.
Για να υλοποιηθεί το συγκεκριμένο βήμα στο JADE, μόλις ανοίξει το JADE RMA GUI πατάμε δεξί κλικ και την επιλογή Start New Agent. Στο όνομα δίνουμε READER (όπως αναφέρθηκε αυτό είναι προκαθορισμένο) και επιλέγουμε την κλάση Reader.RfidAgent και πατάμε οκ και βλέπουμε στο Main-Container τον πράκτορα που μόλις δημιουργήσαμε. Στην συγκεκριμένη περίπτωση η οντότητα READER αντιστοιχίζεται στο κινητό τηλέφωνο της Αλίκης. Με την ίδια διαδικασία ξεκινάμε έναν νέο πράκτορα που τον ονομάζουμε όπως θέλουμε και υλοποιεί την κλάση Tag.RfidAgent. Αυτή η οντότητα προσδιορίζει μια ετικέτα οπότε αν κάποιος θέλει μπορεί να δημιουργήσει και άλλες ετικέτες. Το πρώτο πράγμα που γίνεται είναι η αρχικοποίηση δηλαδή καταχώρηση στο Repository και στο Tag του μυστικού και του ορίζοντα. Μετά εμφανίζεται μια ερώτηση στον χρήστη αν θέλει αυθεντικοποίηση και αν απαντήσει θετικά υλοποιείται το πρωτόκολλο αιτήματος ετικέτας.



Εικόνα 7. *Δημιουργία μιας ετικέτας στο JADE*

4.4.3. Βήμα 2: Μεταβίβαση δικαιωμάτων ανάγνωσης ετικέτας

Η αεροπορική εταιρεία απαιτεί δικαιώματα ανάγνωσης για κάθε ένα αντικείμενο που θα μεταφέρει (π.χ. τσάντες, αθλητικός εξοπλισμός) σε όλη την διάρκεια του ταξιδιού. Έτσι ζητάει από τους επιβάτες να υποβάλλουν TIDs για 5 μέρες για κάθε αντικείμενο (1 TID για κάθε μέρα). Η υποβολή των TID μπορεί να γίνει στην ηλεκτρονική κράτηση ή στο αεροδρόμιο, με την προϋπόθεση για το τελευταίο ότι ο χρήστης έχει μια φορητή συσκευή που μπορεί να μεταδώσει τέτοιες πληροφορίες. Ο πάροχος της υπηρεσίας σώζει στο Repository (Air_Rep) για κάθε επιβάτη, για κάθε είδος το αντίστοιχο TID. Η εγκυρότητα τους μπορεί να διαπιστωθεί κάνοντας αιτήματα προς τα συγκεκριμένα αντικείμενα κατά την καταμέτρηση. Αν η απόκριση της ετικέτας είναι ακατανόητη ο επιβάτης πρέπει να υποβάλει ξανά τις σωστές πληροφορίες. Επίσης ζητείται από του επιβάτες να βάλουν σε αναμονή ή να τερματίσουν οποιαδήποτε άλλη ετικέτα. Στο αεροδρόμιο η Αλίκη, η οποία έχει βάλει σε αναμονή ζητείτε να παράγει το συγκεκριμένο αριθμό TIDs για την βαλίτσα της. Χρησιμοποιώντας το Air_Rep σαν Rep_ID το τηλέφωνο της για κάθε μέρα:

TID_{d_time} = h(Air_Rep, secret, d_time)
Όπου d_time είναι 5 διακριτές τιμές ουσιαστικά πρόκειται για τις ημερομηνίες της σημερινής μέρας και για τις 4 επόμενες. Μεταβιβάζει τα ζεύγη [d_time, TID] στην αεροπορική εταιρεία μέσω ενός ασφαλούς καναλιού. Η αεροπορική εταιρία θα μεταβιβάσει μέρος από τα TIDs στον έλεγχο των αποσκευών, στις αρχές του αεροδρομίου για να παρασχεθούν οι απαιτούμενες υπηρεσίες.
Στο πρακτικό μέρος έχοντας κρατήσει τους δυο πράκτορες του 1ου βήματος δημιουργούμε μια τρίτη οντότητα την ονομάζουμε Air_Rep και υλοποιεί την κλάση DReader.RfidAgent. Με το που γίνει αυτό εμφανίζεται ένα μενού στον ιδιοκτήτη (κινητό Αλίκης) που ρωτάει για ποια ετικέτα θέλει μεταβίβαση δικαιωμάτων και για πόσο χρονικό διάστημα, αυτό παίρνει αυτά που πληκτρολογεί ο χρήστης και μέσω της κλάσης DelTID υπολογίζει τα TID και ακολούθως τα στέλνει στην οντότητα Air_Rep που δημιουργήσαμε. Παρακάτω βλέπουμε τα ζεύγη d_time – TID όπως υπάρχουν στο repository του προσωρινού χρήστη (Air_Rep). Για να επιβεβαιώσει ο προσωρινός χρήστης ότι τα TID είναι έγκυρα αρκεί να αυθεντικοποίησει την ετικέτα. Στο JADE RMA πάνω στον πράκτορα Air_Rep πατάμε δεξί κλικ και στέλνουμε ένα μήνυμα τύπου AGREE. Αν εμφανιστεί ένα μήνυμα που λέει ότι ο χρήστης έχει δικαίωμα ανάγνωσης σημαίνει ότι όλα πήγαν σωστά. Σε αντίθετη περίπτωση σημαίνει ότι έγινε κάποιο λάθος ή ότι το δικαίωμα ανάγνωσης έχει ανακληθεί.

Dreader[0][1]	20110427
Dreader[0][2]	271f21868e3372c52ee990de80b8c4ab0b7465d5
Dreader[1][1]	20110428
Dreader[1][2]	226f0ded97dd43fa9533676db930aa389e399a64
Dreader[2][1]	20110429
Dreader[2][2]	e463711562b01b6b89b33094f5a37661eba9fd3e
Dreader[3][1]	20110430
Dreader[3][2]	d3ae7d849551f42d63fd513f59d9c5b6b53c9768
Dreader[4][1]	20110501
Dreader[4][2]	41d9e6f6578191dfc4f3128fafd19d24422f8749

Εικόνα 8. *Ζεύγη d_time και* TID_{d_time}

4.4.4. Βήμα 3: Αίτηση για ετικέτες από προσωρινούς χρήστες

Κατά την διάρκεια της παραμονής της αποσκευής στην υπηρεσία διαχείρισης θα χρειαστεί να γίνουν αρκετά αιτήματα πως την ετικέτα. Κάθε συσκευή περνάει από σημεία ελέγχου (καταμέτρηση, ακτίνες-Χ, φόρτωση στο αεροπλάνο, εκφόρτωση, κ.α.), και πολλές φορές μπορεί να γίνονται από κάποιον εξωτερικό συνεργάτη. Οπότε η μεταβίβαση συγκεκριμένων TID για να ολοκληρωθούν οι εργασίες αυτές κρίνεται απαραίτητη εφόσον η πολιτική προβλέπει ελέγχους πρόσβαση στο Repository (Air_Rep).
Για να επιβεβαιώσει ο προσωρινός χρήστης ότι τα TID είναι έγκυρα αρκεί να αυθεντικοποίησει την ετικέτα. Στο JADE RMA πάνω στον πράκτορα Air_Rep πατάμε δεξί κλικ και στέλνουμε ένα μήνυμα τύπου AGREE. Αν εμφανιστεί ένα μήνυμα που λέει ότι ο χρήστης έχει δικαίωμα ανάγνωσης σημαίνει ότι όλα πήγαν σωστά. Σε αντίθετη περίπτωση σημαίνει ότι έγινε κάποιο λάθος ή ότι το δικαίωμα ανάγνωσης έχει ανακληθεί.

4.4.5. Βήμα 4: Ανάκληση δικαιωμάτων ανάγνωσης

Εφόσον η Αλίκη προσγειωθεί και παραλάβει την βαλίτσα της. Μπορεί χρησιμοποιώντας το αίτημα ετικέτας να αυθεντικοποίησει την βαλίτσα για

πιστοποιήσει ότι είναι δικιά της. Και μπορεί να ανακαλέσει την μεταβίβαση των δικαιωμάτων ανανεώνοντας τον χρονικό ορίζοντα σε μια τιμή μετά την ‘πέμπτη μέρα’. Από αυτό το σημείο η αεροπορική εταιρεία δεν έχει δικαίωμα ανάγνωσης στην βαλίτσα της Αλίκης.
Για να ανακληθούν τα δικαιώματα ανάγνωσης από τον Air_Rep χρειάζεται να ανανεώσουμε τον ορίζοντα. Για να γίνει αυτό πατάμε δεξί κλικ και μετά send message και αποστέλλουμε ένα μήνυμα τύπου PROPOSE και στο content του μηνύματος το τοπικό όνομα του πράκτορα όπως φαίνεται στο JADE στην περίπτωση μας είναι TAG. Εμφανίζεται ένα μενού στον χρήστη που τον ρωτάει τη θέλει να κάνει ανανέωση μυστικού, ανανέωση ορίζοντα και τα δυο ή τίποτα. Πατάμε το h γιατί θέλουμε ανανέωση ορίζοντα και μας ρωτάει να εισάγουμε το έτος, το μήνα και την ημερομηνία του ορίζοντα που επιθυμούμε. Στην περίπτωση μας βάλαμε το 15/5/2011 και ανακλήθηκε το δικαίωμα ανάγνωσης.

4.5. Μελλοντικές κατευθύνσεις

Στο πρακτικό κομμάτι μπορούμε να προτείνουμε κάποια πράγματα:

1. Κατανεμημένη σχεδίαση: Η τωρινή σχεδίαση του πρωτοκόλλου είναι για τοπική χρήση. Δηλαδή όλοι οι πράκτορες εκτέλουνται σε ένα μόνο μηχάνημα. Με την κατάλληλη χρήση των GUID που παρέχει το JADE για να αλληλεπιδρούν οι οντότητες (Reader, Tag, DReader) μέσω του HTTP σε διαφορετικά μηχανήματα.
2. Αλληλεπίδραση με τον πράκτορα DF: Ο συγκεκριμένος πράκτορας παρέχει την υπηρεσίας κίτρινων σελίδων (yellow pages) που επιτρέπει στους πράκτορες να καταχωρούν και να αναζητούν τις υπηρεσίες που παρέχουν. Αυτό ενδέχεται να μην αλλάξει την λειτουργικότητα της συγκεκριμένης υλοποίησης αλλά να λύσει το πρόβλημα που δημιουργείται με το προκαθορισμένο όνομα της μιας οντότητας.
3. Αλληλεπίδραση με τον πράκτορα AMS : Ο συγκεκριμένος πράκτορας είναι ένας πράκτορας που έχει τη δυνατότητα να δημιουργεί και να σκοτώνει άλλους πράκτορες. Για τη συγκεκριμένη υλοποίηση ίσως να μην χρειαστεί αυτή η δυνατότητα αλλά μελλοντικά για άλλες αντίστοιχες πλατφόρμες σε άλλα δίκτυα (Cloud, MANet) θα φανεί αρκετά χρήσιμο.

5. Συμπέρασματα

Στην εργασία αυτό παρουσιάσαμε τις απειλές κατά τις ιδιωτικότητας που προκύπτουν από τη χρήση συστημάτων RFID. Αφού πρώτα περιγράψαμε τα τεχνικά χαρακτηριστικά των συστημάτων αυτών, αλλά και τα πρότυπα που τα συνοδεύουν, παραθέσαμε ένα σύνολο εφαρμογών και χρήσεών τους. Στη συνέχεια εξηγήσαμε τον τρόπο που οι εφαρμογές αυτές, σε συνδυασμό με τη ραγδαία εξάπλωση των συστημάτων RFID, μπορούν να αποτελέσουν απειλή για την ιδιωτικότητα των χρηστών τους. Κατηγοριοποιήσαμε τις απειλές και αναλύσαμε με παραδείγματα καθεμία από αυτές. Μετά είδαμε ποιες λύσεις έχουν προταθεί.
Στη συνέχεια μελετήσαμε ένα πρωτόκολλο που προτείνεται με την χρήση πρακτόρων. Αναλύθηκε η τεχνολογία των πρακτόρων λογισμικού και συγκεκριμένα μια πλατφόρμα το JADE που χρησιμποιήθηκε. Στην υλοποίηση ουσιαστικά δημιουργήσαμε προγραμματιστικά το πρωτόκολλο στο δεύτερο κεφάλαιο πάνω στην

πλατφόρα JADE και το τρέξαμε σε ένα σενάριο και είδαμε ότι λειτουργεί κανονικά. Επίσης προτείναμε μελλοντικές κατευθύνσεις στο πρακτικό και μόνο κομμάτι, αφού θεωρούμε ότι η πλατφόρμα είναι η πιο σημαντική και επειδή βασίζεται σε πράκτορες λογισμικού θα μπορούσε να έχει και άλλες εφαρμογές.

Παράρτημα

1. Οι πηγαίοι κώδικες για την οντότητα Reader

- **Η κύρια κλάση RfidAgent**

```
package Reader;

import jade.core.AID;
import jade.core.Agent;
import java.util.*;
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.util.logging.Level;
import java.util.logging.Logger;
import jade.lang.acl.ACLMessage;
import jade.lang.acl.MessageTemplate;
import jade.core.behaviours.*;

/**
*
* @author jim
*/
public class RfidAgent extends Agent {

public int i, j, z, tempCurrentTime, temphorizon;
public String unTID, tagInstanceId,nonceB, nonceA, nonceC, pseudTag, pseud,
unPseud, tID, repId,currentTime, ns, secretold, secretnewXORnonceB, nh, dRepId,
delTagInstanceId;
public String horizon, menu, menu2, sessionKey, sCheck, secretNew,
secretoldXORnonceC, secretoldXORnonceB,deleteTagId;
public String[][] Rep = new String[30][8];
private MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.CFP);
private MessageTemplate in =
MessageTemplate.MatchPerformative(ACLMessage.INFORM);
private MessageTemplate req =
MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
private MessageTemplate pro =
MessageTemplate.MatchPerformative(ACLMessage.PROPOSE);
private MessageTemplate px =
MessageTemplate.MatchPerformative(ACLMessage.PROXY);
private MessageTemplate fa =
MessageTemplate.MatchPerformative(ACLMessage.FAILURE);

@Override
public void setup() {

for (z = 0; z < Rep.length; z++) {
```

```
for (j = 0; j < 8.; j++) {
Rep[z][j] = null;
}
}

addBehaviour(new TickerBehaviour(this, 5000) { // Initialization Behaviour
protected void onTick() {

ACLMessage msg = receive(in);  //Receive tagInstanceId (for each tag created)
if (msg != null) {
for (z = 0; z < Rep.length; z++) {
if (Rep[z][0] == null) {
Rep[z][0] = msg.getContent();
try {
Rep[z][2] = RNG.getRandom3(); //Registry secret (old) to Repository
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

Rep[z][1] = "" +RNG.getRandomTagId();  //Registry TagID to Repository
Rep[z][4] = Horizon.getHorizon(); //Registry horizon(old) to Repository
ACLMessage msg1 = new ACLMessage(ACLMessage.INFORM);
msg1.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg1.setContent(Rep[z][4]);
send(msg1);                    //send horizon
ACLMessage msg2 = new ACLMessage(ACLMessage.INFORM);
msg2.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg2.setContent(Rep[z][2]);
send(msg2);  //Send secret to the corresponding TAG
System.out.println("To tag vrisketai stin  : " + z + "grammi");
break;
}
}
} else {
block();
}

}
});


addBehaviour(new TickerBehaviour(this, 5000) { //Authentication Behaviour
protected void onTick() {

ACLMessage msg3 = receive(mt);//Receive CFP (Call For Proposal) messages
Sending those messages can be done through JADE RMA.
if (msg3 != null) {
tagInstanceId = msg3.getContent(); //In the message's content should be
tagInstanceId,which agent's local name at JADE RMA.
for (z = 0; z < Rep.length; z++) {
```

```
if (Rep[z][0] == null ? tagInstanceId == null : Rep[z][0].equals(tagInstanceId)) {
System.out.println("Thelete authentikopoihsh gia to Tag  " + Rep[z][0] + " patiste y
gia sinexeia kai n gia termatismo ");//Authentication Question
Scanner keyboard = new Scanner(System.in);
menu = keyboard.nextLine();
while (menu.equalsIgnoreCase("y") == false & menu.equalsIgnoreCase("n") == false)
{
System.out.println("Thelete authentikopoihsh gia to Tag  " + Rep[z][0] + " patiste y
gia sinexeia kai n gia termatismo ");
menu = keyboard.nextLine();
}
if (menu.equalsIgnoreCase("y")) {
ACLMessage msg4 = new ACLMessage(ACLMessage.CFP);
msg4.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
repId = getAID().getLocalName();
msg4.setContent(repId);
send(msg4);// Create and send message with content RepositoryID and Receiver
TagInstance Id (the corresponding tag), in our case repId is predefined READER
ACLMessage msg5 = new ACLMessage(ACLMessage.CFP);
msg5.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
currentTime = Horizon.getHorizon();
System.out.println("CurrentTime : " + currentTime);
msg5.setContent(currentTime);
send(msg5);//Create and send message with content the current Date and it's format
YYYYMMDD according to ISO 8601
ACLMessage msg6 = new ACLMessage(ACLMessage.CFP);
msg6.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));

try {
nonceA = RNG.getRandom2(); //nonceA
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
System.out.println("nonceA sto Reader:   " + nonceA);
msg6.setContent(nonceA);//Create and send message with content nonceA which is
created by an RNG
send(msg6);

} else if (menu.equalsIgnoreCase("n")) {
System.out.println("Zitasate na min authentikopoihthei to Tag  " + Rep[z][0]);
Rep[z][7] = "n";
System.out.println("tipwnei to swsto???  " + Rep[z][7]);
System.out.println("To tag vrisketai stin  : " + z + "grammi");
}
}//end if

} //end for

}
ACLMessage msg7 = receive(req); //Receive request messages
```

```
if (msg7 != null) {
pseudTag = msg7.getContent(); //Receive pseud ,calculated from tag
System.out.println("Tag Pseud     " + pseudTag);
ACLMessage msg8 = receive(req);
if (msg8 != null) {
for (z = 0; z < Rep.length; z++) {
if (Rep[z][0] == null ? tagInstanceId == null : Rep[z][0].equals(tagInstanceId)) {
Rep[z][6] = msg8.getContent(); //Receive nonceB,which was used to calculate pseud
System.out.println(" nonceB APO TO TAG  " + Rep[z][6]);
} else {
continue;
}
}
for (z = 0; z < Rep.length; z++) {
if (Rep[z][0] != null) {
temphorizon = 0;
tempCurrentTime = 0;
currentTime = Horizon.getHorizon();
temphorizon = Integer.parseInt(Rep[z][4].trim());
tempCurrentTime = Integer.parseInt(currentTime.trim());
if (temphorizon >= tempCurrentTime) { //Check if horizon>currentTime
currentTime = "" + temphorizon;
} else {
currentTime = "" + tempCurrentTime;
}
unTID = Converter.concatThreeStrings(repId, Rep[z][2], currentTime);
System.out.println("unTID ston reader :  " + unTID);
try {
tID = AeSimpleSHA1.SHA1(unTID);
System.out.println(tID);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

unPseud = Converter.concatThreeStrings(nonceA, tID, Rep[z][6]);
System.out.println("To unPseud tou reader:     " + unPseud);
try {
pseud = AeSimpleSHA1.SHA1(unPseud);
System.out.println("To pseud tou reader :    " + pseud);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

if (pseud == null ? pseudTag == null : pseud.equals(pseudTag)) { //check if pseud
which was calculated by the tag is the same with the one that reader calculate.
System.out.println("To TAG  " + Rep[z][0] + "  EINAI SWSTO");
```

```
System.out.println("To tag vrisketai stin  : " + z + "grammi");
Rep[z][7] = "y";
System.out.println("to vlepeis swsta  " + Rep[z][7]);
break;
} else {
continue;
}

} else if (Rep[z][0] == null & z==Rep.length) {
System.out.println("To tag den uparxei");
} else {
break;
}
}
} else {
block();
}
}

}
});

addBehaviour(new TickerBehaviour(this, 10000) { // Secret Update Behaviour
protected void onTick() {

ACLMessage msg23 = receive(pro);
if (msg23 != null) {
tagInstanceId = msg23.getContent();
for (z = 0; z < Rep.length; z++) {
if (Rep[z][0] == null ? tagInstanceId == null : Rep[z][0].equals(tagInstanceId)) {

System.out.println("Patiste S gia secret update,h gia horizon update,b kai gia ta dio,n
na min ginei tipota gia to Tag   " + Rep[z][0]);
Scanner keyb = new Scanner(System.in);
menu2 = keyb.nextLine();
while (menu2.equalsIgnoreCase("s") == false & menu2.equalsIgnoreCase("h") ==
false & menu2.equalsIgnoreCase("b") == false & menu2.equalsIgnoreCase("n") ==
false) {
System.out.println("Patiste S gia secret update,h gia horizon update,b kai gia ta dio,n
na min ginei tipota gia to Tag   " + Rep[z][0]);
menu2 = keyb.nextLine();
}
if (menu2.equalsIgnoreCase("s")) {
try {
secretNew = RNG.getRandom3(); //secretNew
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
System.out.println("To secret new tou reader :    " + secretNew);
try {
```

```
nonceC = RNG.getRandom(); //nonceC
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
byte[] SecretoldByte = Converter.hexStringToByteArray(Rep[z][2]);
byte[] nonceBbyte = Converter.hexStringToByteArray(Rep[z][6]);
byte[] SecretoldXORnonceBbyte = Converter.xorByteArray(SecretoldByte,
nonceBbyte);
try {
secretoldXORnonceB = Converter.getHexString(SecretoldXORnonceBbyte);
//Convert from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

try {
sessionKey =
AeSimpleSHA1.SHA1(Converter.concatTwoStrings(secretoldXORnonceB,
nonceC)); //sessionKey in HEX format
System.out.println("To session key tou reader :    " + sessionKey);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}


byte[] SecretnewByte = Converter.hexStringToByteArray(secretNew); //SecretNew to
byte[]
byte[] SessionkeyByte = Converter.hexStringToByteArray(sessionKey); //sessionKey
to byte[]
byte[] NSbyte = Converter.xorByteArray(SecretnewByte, SessionkeyByte); //xor
sessionKey and SecretN
try {
ns = Converter.getHexString(NSbyte); //Convert ns from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
byte[] nonceCbyte = Converter.hexStringToByteArray(nonceC);
byte[] secretnewXORnonceBbyte = Converter.xorByteArray(SecretnewByte,
nonceBbyte); //xor secretnew and nonceB
byte[] secretoldXORnonceCbyte = Converter.xorByteArray(SecretoldByte,
nonceCbyte); //xor secretold and nonceC

try {
secretoldXORnonceC = Converter.getHexString(secretoldXORnonceCbyte);
//Convert from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
```

```
try {
secretnewXORnonceB = Converter.getHexString(secretnewXORnonceBbyte);
//Convert from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

try {
sCheck = AeSimpleSHA1.SHA1(Converter.concatTwoStrings(secretoldXORnonceC,
secretnewXORnonceB));
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

System.out.println("To nonceC  :    " + nonceC);
System.out.println("To NS:    " + ns);
System.out.println("To Scheck tou reader :    " + sCheck);
ACLMessage msg8 = new ACLMessage(ACLMessage.REQUEST);
msg8.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg8.setContent(nonceC);
send(msg8);
ACLMessage msg9 = new ACLMessage(ACLMessage.REQUEST);
msg9.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg9.setContent(ns);
send(msg9);
ACLMessage msg10 = new ACLMessage(ACLMessage.REQUEST);
msg10.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg10.setContent(sCheck);
send(msg10);
Rep[z][3] = "" + secretNew;
Rep[z][2] = Rep[z][3];


} //end of menu "s"
else if (menu2.equalsIgnoreCase("h")) {//start of horizon update
horizon = Horizon.setHorizon(Rep[z][0]);

try {
nh = AeSimpleSHA1.SHA1(Converter.concatThreeStrings(horizon, Rep[z][2],
Rep[z][6]));
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
ACLMessage msg11 = new ACLMessage(ACLMessage.PROPAGATE);
msg11.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg11.setContent(horizon);
```

```
send(msg11);
ACLMessage msg12 = new ACLMessage(ACLMessage.PROPAGATE);
msg12.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg12.setContent(nh);
send(msg12);
Rep[z][5] = horizon; //horizon (new)
Rep[z][4] = Rep[z][5]; //change horizon old with new
} //end of menu h
else if (menu2.equalsIgnoreCase("b")) { //start of menu b
try {

secretNew = RNG.getRandom3(); //secretNew
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
System.out.println("To secret new tou reader :    " + secretNew);

try {
nonceC = RNG.getRandom(); //Create nonceC
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
byte[] SecretoldByte = Converter.hexStringToByteArray(Rep[z][2]);
byte[] nonceBbyte = Converter.hexStringToByteArray(Rep[z][6]);
byte[] SecretoldXORnonceBbyte = Converter.xorByteArray(SecretoldByte,
nonceBbyte);
try {
secretoldXORnonceB = Converter.getHexString(SecretoldXORnonceBbyte);
//Convert from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
try {
sessionKey =
AeSimpleSHA1.SHA1(Converter.concatTwoStrings(secretoldXORnonceB,
nonceC)); //sessionKey in HEX format
System.out.println("To session key tou reader :    " + sessionKey);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
byte[] SecretnewByte = Converter.hexStringToByteArray(secretNew);
byte[] SessionkeyByte = Converter.hexStringToByteArray(sessionKey);
byte[] NSbyte = Converter.xorByteArray(SecretnewByte, SessionkeyByte); //xor
sessionKey and SecretN
try {
ns = Converter.getHexString(NSbyte); //Convert ns from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
```

```
}
byte[] nonceCbyte = Converter.hexStringToByteArray(nonceC);
byte[] secretnewXORnonceBbyte = Converter.xorByteArray(SecretnewByte,
nonceBbyte); //xor secretnew and nonceB
byte[] secretoldXORnonceCbyte = Converter.xorByteArray(SecretoldByte,
nonceCbyte); //xor secretold and nonceC

try {
secretoldXORnonceC = Converter.getHexString(secretoldXORnonceCbyte);
//Convert from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
try {
secretnewXORnonceB = Converter.getHexString(secretnewXORnonceBbyte);
//Convert from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}


try {
sCheck = AeSimpleSHA1.SHA1(Converter.concatTwoStrings(secretoldXORnonceC,
secretnewXORnonceB));
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
System.out.println("To nonceC  :    " + nonceC);
System.out.println("To NS:    " + ns);
System.out.println("To Scheck tou reader :    " + sCheck);
ACLMessage msg8 = new ACLMessage(ACLMessage.REQUEST);
msg8.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg8.setContent(nonceC); //send nonceC to the corresponding tag
send(msg8);
ACLMessage msg9 = new ACLMessage(ACLMessage.REQUEST);
msg9.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg9.setContent(ns); //send ns to the corresponding tag
send(msg9);
ACLMessage msg10 = new ACLMessage(ACLMessage.REQUEST);
msg10.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg10.setContent(sCheck); //send ns to the corresponding tag
send(msg10);
Rep[z][3] = secretNew;
Rep[z][2] = Rep[z][3];
try {
Thread.sleep(2000);
} catch (InterruptedException e) {
System.out.println("awakened prematurely");
```

```
}
horizon = Horizon.setHorizon(Rep[z][0]);
try {
nh = AeSimpleSHA1.SHA1(Converter.concatThreeStrings(horizon, Rep[z][2],
Rep[z][6]));
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
ACLMessage msg11 = new ACLMessage(ACLMessage.PROPAGATE);
msg11.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg11.setContent(horizon);
send(msg11);
ACLMessage msg12 = new ACLMessage(ACLMessage.PROPAGATE);
msg12.addReceiver(new AID(Rep[z][0], AID.ISLOCALNAME));
msg12.setContent(nh);
send(msg12);
Rep[z][5] = horizon;
Rep[z][4] = Rep[z][5];
} //end of menu b
else if (menu2.equalsIgnoreCase("n")) {
continue;
}
} else {
continue;
}
} //end of for
} else {
block();
}
}
});
addBehaviour(new TickerBehaviour(this, 10000) { // Delegation Behaviour
protected void onTick() {

ACLMessage msg13 = receive(px); //Receive Deleagation Reader ID
if (msg13 != null) {
dRepId = msg13.getContent();
System.out.println("Brethike Delegation Reader    " + dRepId);
String[][] Drep = DelTID.getDelTID(dRepId, Rep); //Create TID for the
corresponding Delegation Reader
for (i = 0; i < 1; i++) {
delTagInstanceId = Drep[i][0];
ACLMessage msg14 = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
msg14.addReceiver(new AID(dRepId, AID.ISLOCALNAME));
msg14.setContent(delTagInstanceId);
send(msg14);
}
```

```
for (i = 0; i < Drep.length; i++) {
ACLMessage msg15 = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
msg15.addReceiver(new AID(dRepId, AID.ISLOCALNAME));
msg15.setContent(Drep[i][1]);
send(msg15);
ACLMessage msg16 = new ACLMessage(ACLMessage.ACCEPT_PROPOSAL);
msg16.addReceiver(new AID(dRepId, AID.ISLOCALNAME));
msg16.setContent(Drep[i][2]);
send(msg16);
}
} else {
block();
}
}
});



addBehaviour(new TickerBehaviour(this, 5000) { //Delete Tag's from Reader
Repository

protected void onTick() {

ACLMessage msg24 = receive(fa);
if (msg24 != null) {
deleteTagId = msg24.getContent();
}
for (i = 0; i < Rep.length; i++) {
if (Rep[i][0] == null ? deleteTagId == null : Rep[i][0].equals(deleteTagId)) {
for (j = 0; j < 8; j++) {
Rep[i][j] = null;
}
} else {
continue;
}

}
}
});
}
}
```

- **Η κλάση DelTID**

```
package Reader;

import java.text.*;
import java.util.*;
import java.io.UnsupportedEncodingException;
```

```
import java.security.NoSuchAlgorithmException;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
*
* @author jim
*/
public class DelTID {

private static String DelTagInstanceId, Dtime, DTID;
private static int Days, i, z;

public static String[][] getDelTID(String DRepId, String Rep[][]) {

System.out.println("Gia poio tag theleis na ginei delegation   ");
Scanner keyboard = new Scanner(System.in);
DelTagInstanceId = keyboard.nextLine();
System.out.println("Gia poses meres tha ginei to delegation gia to tag   " +
DelTagInstanceId);
Days = keyboard.nextInt();
String[][] DRep = new String[Days][3];
for (i = 0; i < Days; i++) {
if (DRep[i][0] == null) {
Calendar origDay = Calendar.getInstance();
Calendar nextDay = (Calendar) origDay.clone();
nextDay.add(Calendar.DAY_OF_YEAR, i);
Date tomorrow = nextDay.getTime();
SimpleDateFormat dateFormat = new SimpleDateFormat("yyyyMMdd");
Dtime = dateFormat.format(tomorrow);
System.out.println("FORMATdate: " + Dtime);
DRep[i][0] = DelTagInstanceId;
DRep[i][1] = Dtime;
for (z = 0; z < Rep.length; z++) {
if (Rep[z][0] == null ? DelTagInstanceId == null :
Rep[z][0].equals(DelTagInstanceId)) {

try {
DTID = AeSimpleSHA1.SHA1(Converter.concatThreeStrings(DRepId, Rep[z][2],
Dtime));
System.out.println(DTID);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

DRep[i][2] = DTID;
break;
```

```
} else {
continue;
}
}
} else {
continue;
}
}
return DRep;
}
}
```

- **Η κλάση Horizon**

```
package Reader;

/**
*
* @author jim
*/
import java.util.*;
import java.util.Calendar;

public class Horizon {

private static int horizonyear,horizonmonth,horizondate,year,month,date;
private static String h,Hmonth,Hdate,Smonth,Sdate,hc;

public static String setHorizon(String TagInstanceId){
System.out.println("Dwse etos (px. 2011) gia horizon gia Tag me ID    " +
TagInstanceId);
Scanner hy = new Scanner(System.in);
horizonyear = hy.nextInt();
System.out.println("Dwse mina (1-12) gia horizon gia Tag me ID    " + TagInstanceId
);
Scanner hm = new Scanner(System.in);
horizonmonth = hm.nextInt();
while (horizonmonth >12  ||  horizonmonth <1) {
System.out.println("Dwse mina (1-12) gia horizon gia Tag me ID    " +
TagInstanceId);
Scanner hmt = new Scanner(System.in);
horizonmonth = hmt.nextInt();
}
if (horizonmonth<10) {
Hmonth="0"+horizonmonth;
}
else{
Hmonth=""+horizonmonth;
```

```
}
System.out.println("Dwse hmerominia (1-31) gia horizon gia Tag me ID    " +
TagInstanceId );
Scanner hd = new Scanner(System.in);
horizondate = hd.nextInt();
while (horizondate >31 || horizondate <1){
System.out.println("Dwse hmerominia (1-31) gia horizon gia Tag me ID    " +
TagInstanceId );
Scanner hdt = new Scanner(System.in);
horizondate = hdt.nextInt();
}
if (horizondate<10) {
Hdate="0"+horizondate;
}
else{
Hdate=""+horizondate;
}
h=""+horizonyear+Hmonth+Hdate;
System.out.println("horizon :    " + h );
return h;
}

public static String getHorizon(){
Calendar calendar = new GregorianCalendar();
year = calendar.get(Calendar.YEAR);
month = calendar.get(Calendar.MONTH);
date = calendar.get(Calendar.DATE);
month=month+1;
if (month<10) {
Smonth="0"+month;
}
else {
Smonth=""+month;
}
if (date<10) {
Sdate="0"+date;
}
else {
Sdate=""+date;
}
hc=""+year+Smonth+Sdate;
return hc;
}


}
```

2. Οι πηγαίοι κώδικες για την οντότητα TAG

- **Η κύρια κλάση RfidAgent**

```
package Tag;

import jade.core.Agent;
import jade.core.AID;
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.util.logging.Level;
import java.util.logging.Logger;
import jade.lang.acl.ACLMessage;
import jade.core.behaviours.*;
import jade.lang.acl.MessageTemplate;

/**
*
* @author jim
*/
public class RfidAgent extends Agent {

public int i, j, tempCurrentTime, temphorizon, tempDtime;
public String unTID, repId, CurrentTime, nonceA, nonceB, unPseud, tID, pseud,
tagInstanceId, secretnewXORnonceB, secretoldXORnonceC, nhTag, dRepId;
public String horizon,nonceC, ns, sCheck, secretoldXORnonceB, sessionKey,
secretNew, unScheckTag, sCheckTag, nh, unNHtag, dTime, dtempnonceB;
public String[] Tag = new String[3];
private MessageTemplate mt =
MessageTemplate.MatchPerformative(ACLMessage.CFP);
private MessageTemplate in =
MessageTemplate.MatchPerformative(ACLMessage.INFORM);
private MessageTemplate req =
MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
private MessageTemplate prg =
MessageTemplate.MatchPerformative(ACLMessage.PROPAGATE);
private MessageTemplate px =
MessageTemplate.MatchPerformative(ACLMessage.PROXY);

@Override
public void setup() {

System.out.println("My local-name is     " + getAID().getLocalName());
tagInstanceId = getAID().getLocalName();
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("READER", AID.ISLOCALNAME));
msg.setContent(tagInstanceId);
send(msg);
```

```
addBehaviour(new TickerBehaviour(this, 5000) { // Initialization Behaviour
protected void onTick() {
ACLMessage msg1 = receive(in);
if (msg1 != null) {
Tag[0] = msg1.getContent();
System.out.println("Tag horizon       " + Tag[0]);
ACLMessage msg2 = receive(in);
if (msg2 != null) {
Tag[1] = msg2.getContent();
System.out.println("Secret       " + Tag[1]);
ACLMessage msg3 = new ACLMessage(ACLMessage.CFP);     //this message
begins the authentication behaviour(optional)
msg3.addReceiver(new AID("READER", AID.ISLOCALNAME));
msg3.setContent(tagInstanceId);
send(msg3);
} else {
block();
}

} else {
block();
}


}
});   // end of Initialization Behaviour

addBehaviour(new TickerBehaviour(this, 5000) {

protected void onTick() {

CurrentTime = "";

ACLMessage msg4 = receive(mt);
if (msg4 != null) {
repId = msg4.getContent();
System.out.println("Tag RepId    " + repId);
ACLMessage msg5 = receive(mt);
if (msg5 != null) {
CurrentTime = msg5.getContent();
System.out.println("Tag CurrentTime    " + CurrentTime);
ACLMessage msg6 = receive(mt);
if (msg6 != null) {
nonceA = msg6.getContent();
System.out.println("Tag nonceA        " + nonceA);
temphorizon = 0;
tempCurrentTime = 0;
temphorizon = Integer.parseInt(Tag[0].trim());
tempCurrentTime = Integer.parseInt(CurrentTime.trim());
```

```
if (temphorizon >= tempCurrentTime) {
CurrentTime = "" + temphorizon;
} else {
CurrentTime = "" + tempCurrentTime;
}
unTID = Converter.concatThreeStrings(repId, Tag[1], CurrentTime);
System.out.println("unTID     " + unTID);
try {
tID = AeSimpleSHA1.SHA1(unTID);
System.out.println(tID);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
try {
Tag[2] = RNG.getRandom(); //nonceB
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

System.out.println("nonceB     " + Tag[2]);
unPseud = Converter.concatThreeStrings(nonceA, tID, Tag[2]);
System.out.println("unPseud     " + unPseud);
try {
pseud = AeSimpleSHA1.SHA1(unPseud);
System.out.println(pseud);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
ACLMessage msg7 = new ACLMessage(ACLMessage.REQUEST);
msg7.addReceiver(new AID(repId, AID.ISLOCALNAME));
msg7.setContent(pseud);
send(msg7);
ACLMessage msg8 = new ACLMessage(ACLMessage.REQUEST);
msg8.addReceiver(new AID(repId, AID.ISLOCALNAME));
msg8.setContent(Tag[2]);
send(msg8);


} else {
block();
}
} else {
block();
}
} else {
block();
```

```
}
}
});



addBehaviour(new TickerBehaviour(this, 5000) {  //Secret Update behaviour
protected void onTick() {


ACLMessage msg8 = receive(req);
if (msg8 != null) {
nonceC = msg8.getContent();
System.out.println("nonceC sto tag  " + nonceC);
ACLMessage msg9 = receive(req);
if (msg9 != null) {
ns = msg9.getContent();
System.out.println("NS sto tag    " + ns);
ACLMessage msg10 = receive(req);
if (msg10 != null) {
sCheck = msg10.getContent();
System.out.println("Scheck sto tag    " + sCheck);
byte[] SecretoldByte = Converter.hexStringToByteArray(Tag[1]);
byte[] nonceBbyte = Converter.hexStringToByteArray(Tag[2]);
byte[] secretoldXORnonceBbyte = Converter.xorByteArray(SecretoldByte,
nonceBbyte);
try {
secretoldXORnonceB = Converter.getHexString(secretoldXORnonceBbyte);
//Convert ns from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

try {
sessionKey =
AeSimpleSHA1.SHA1(Converter.concatTwoStrings(secretoldXORnonceB,
nonceC)); //sessionKey in HEX format
System.out.println("To session key tou tag :    " + sessionKey);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

byte[] NSbyte = Converter.hexStringToByteArray(ns);
byte[] SessionkeyByte = Converter.hexStringToByteArray(sessionKey);
byte[] NSXORsessionkeybyte = Converter.xorByteArray(NSbyte, SessionkeyByte);
//secretNew in bytre[]
try {
```

```
secretNew = Converter.getHexString(NSXORsessionkeybyte); //Convert secretNew
from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

byte[] nonceCbyte = Converter.hexStringToByteArray(nonceC);
byte[] secretoldXORnonceCbyte = Converter.xorByteArray(SecretoldByte,
nonceCbyte);
byte[] secretnewXORnonceBbyte = Converter.xorByteArray(NSXORsessionkeybyte,
nonceBbyte);
try {
secretoldXORnonceC = Converter.getHexString(secretoldXORnonceCbyte);
//Convert ns from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
try {
secretnewXORnonceB = Converter.getHexString(secretnewXORnonceBbyte);
//Convert ns from byte[] to hex string
} catch (Exception ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

try {
sCheckTag =
AeSimpleSHA1.SHA1(Converter.concatTwoStrings(secretoldXORnonceC,
secretnewXORnonceB)); //sCheck'
System.out.println("To Scheck tou tag:     " + sCheckTag);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
if (sCheckTag == null ? sCheck == null : sCheckTag.equals(sCheck)) {
System.out.println("egine to secret update");
Tag[1] = secretNew;
System.out.println("to secret new sto tag : " + Tag[1]);

} else {
System.out.println("den egine to secret update");
}
} else {
block();
}
} else {
block();
}
} else {
block();
```

```
}

ACLMessage msg10 = receive(prg);
if (msg10 != null) {
horizon = msg10.getContent();                //horizon (new)
System.out.println("horizon sto tag  " + horizon);
ACLMessage msg11 = receive(prg);
if (msg11 != null) {
nh = msg11.getContent();
System.out.println("NH sto tag    " + nh);
try {
nhTag = AeSimpleSHA1.SHA1(Converter.concatThreeStrings(horizon, Tag[1],
Tag[2])); //nh'
System.out.println("To NH tou tag:    " + nhTag);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
if (nhTag == null ? nh == null : nhTag.equals(nh)) {
System.out.println("egine to horizon update");
Tag[0] = horizon;
System.out.println("to horizon new sto tag : " + Tag[0]);
} else {
System.out.println("den egine to horizon update");
}
} else {
block();
}
} else {
block();
}

}
});



addBehaviour(new TickerBehaviour(this, 5000) {   //Delegation Behaviour
protected void onTick() {

ACLMessage msg17 = receive(px);
if (msg17 != null) {
dRepId = msg17.getContent();
System.out.println("Tag DRepId    " + dRepId);
ACLMessage msg18 = receive(px);
if (msg18 != null) {
nonceA = msg18.getContent();
System.out.println("nonceA sto tag  " + nonceA);
```

```
ACLMessage msg19 = receive(px);
if (msg19 != null) {
dTime = msg19.getContent();
temphorizon = 0;
tempDtime = 0;
System.out.println("Tag Dtime       " + dTime);
temphorizon = Integer.parseInt(Tag[0].trim());
tempDtime = Integer.parseInt(dTime.trim());
if (temphorizon >= tempDtime) {
dTime = "" + temphorizon;
} else {
dTime = "" + tempDtime;
}
unTID = Converter.concatThreeStrings(dRepId, Tag[1], dTime);
System.out.println("unTID     " + unTID);
try {
tID = AeSimpleSHA1.SHA1(unTID);
System.out.println(tID);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
try {
nonceB = RNG.getRandom(); //nonceB
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}

System.out.println("nonceB     " + nonceB);
unPseud = Converter.concatThreeStrings(nonceA, tID, nonceB);
System.out.println("unPseud     " + unPseud);
try {
pseud = AeSimpleSHA1.SHA1(unPseud);
System.out.println(pseud);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
ACLMessage msg20 = new ACLMessage(ACLMessage.CONFIRM);
msg20.addReceiver(new AID(dRepId, AID.ISLOCALNAME));
msg20.setContent(pseud);
send(msg20);
ACLMessage msg21 = new ACLMessage(ACLMessage.CONFIRM);
msg21.addReceiver(new AID(dRepId, AID.ISLOCALNAME));
msg21.setContent(nonceB);
send(msg21);
} else {
block();
```

```
}
} else {
block();
}
} else {
block();
}
}
});
}
}
```

3. Οι πηγαίοι κώδικες για την οντότητα DReader

- **Η κύρια κλάση RfidAgent**

```
package DReader;

import jade.core.Agent;
import jade.core.AID;
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.util.logging.Level;
import java.util.logging.Logger;
import jade.lang.acl.ACLMessage;
import jade.core.behaviours.*;
import jade.lang.acl.MessageTemplate;

/**
*
* @author jim
*/
public class RfidAgent extends Agent {

public int i, j;
public String currentTime, nonceA, nonceB, unPseud, pseud, delTagInstanceId,
pseudTag, dRepId, deleteTagId;
public String[][] Dreader = new String[30][3];
private MessageTemplate px =
MessageTemplate.MatchPerformative(ACLMessage.PROXY);
private MessageTemplate cm =
MessageTemplate.MatchPerformative(ACLMessage.CONFIRM);
private MessageTemplate ag =
MessageTemplate.MatchPerformative(ACLMessage.AGREE);
private MessageTemplate fd =
MessageTemplate.MatchPerformative(ACLMessage.REFUSE);
```

```
private MessageTemplate ap =
MessageTemplate.MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);

@Override
public void setup() {
for (i = 0; i < Dreader.length; i++) {
for (j = 0; j < 3.; j++) {
Dreader[i][j] = null;
}
}
System.out.println("My local-name is     " + getAID().getLocalName());
dRepId = getAID().getLocalName();
ACLMessage msg13 = new ACLMessage(ACLMessage.PROXY);
msg13.addReceiver(new AID("READER", AID.ISLOCALNAME));
msg13.setContent(dRepId);
send(msg13);

addBehaviour(new TickerBehaviour(this, 5000) { // Delegation Behaviour

protected void onTick() {
ACLMessage msg14 = receive(ap); //Receive dtime from reader
if (msg14 != null) {
delTagInstanceId = msg14.getContent();
} else {
block();
}
for (i = 0; i < Dreader.length; i++) {
if (Dreader[i][0] == null) {
ACLMessage msg15 = receive(ap);
if (msg15 != null) {
Dreader[i][1] = msg15.getContent();
Dreader[i][0] = delTagInstanceId;
System.out.println("Dreader[" + i + "][1]     " + Dreader[i][1]);
ACLMessage msg16 = receive(ap);//Receive TIDs from reader
if (msg16 != null) {
Dreader[i][2] = msg16.getContent();
System.out.println("Dreader[" + i + "][2]     " + Dreader[i][2]);
} else {
block();
}
} else {
block();
}
} else {
continue;
}
}
ACLMessage msg22 = receive(ag);
if (msg22 != null) {
currentTime = Time.getCTime();
```

```
for (i = 0; i < Dreader.length; i++) {
if (Dreader[i][1] == null ? currentTime == null : Dreader[i][1].equals(currentTime)) {
//compare dtime with current time
ACLMessage msg17 = new ACLMessage(ACLMessage.PROXY);
msg17.addReceiver(new AID(Dreader[i][0], AID.ISLOCALNAME));
msg17.setContent(dRepId);    //send delegation RepID
send(msg17);
try {
nonceA = RNG.getRandom2(); //nonceA
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
ACLMessage msg18 = new ACLMessage(ACLMessage.PROXY);
msg18.addReceiver(new AID(Dreader[i][0], AID.ISLOCALNAME));
msg18.setContent(nonceA);   //send nonceA
send(msg18);
ACLMessage msg19 = new ACLMessage(ACLMessage.PROXY);
msg19.addReceiver(new AID(Dreader[i][0], AID.ISLOCALNAME));
msg19.setContent(Dreader[i][1]);   //send dtime
send(msg19);
}
}//end for
}
ACLMessage msg20 = receive(cm); //Receive pseud from Tag
if (msg20 != null) {
pseudTag = msg20.getContent();
System.out.println("Tag Pseud     " + pseudTag);
ACLMessage msg21 = receive(cm);
nonceB = msg21.getContent();
for (i = 0; i < Dreader.length; i++) {
if (Dreader[i][0] != null) {
unPseud = Converter.concatThreeStrings(nonceA, Dreader[i][2], nonceB);
System.out.println("To unPseud tou reader:     " + unPseud);
try {
pseud = AeSimpleSHA1.SHA1(unPseud);
System.out.println("To pseud tou reader :     " + pseud);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
if (pseud == null ? pseudTag == null : pseud.equals(pseudTag)) {
System.out.println("To TAG  " + Dreader[i][0] + "  mporei na to anagnwsei o
Delreader");
break;
} else {
continue;
}

} else if (Dreader[i][0] == null & i == Dreader.length) {
```

```
System.out.println("Den uparxei dikaiwma anagnwsis");
} else {
continue;
}

}
}
}
});

addBehaviour(new TickerBehaviour(this, 5000) { //Delete Tag's from Delegation
Reader Repository

protected void onTick() {
ACLMessage msg23 = receive(fd);
if (msg23 != null) {
deleteTagId = msg23.getContent();
}
for (i = 0; i < Dreader.length; i++) {
if (Dreader[i][0] == null ? deleteTagId == null : Dreader[i][0].equals(deleteTagId)) {
for (j = 0; j < 3; j++) {
Dreader[i][j] = null;
}
} else {
continue;
}
}
}
});
}
}
```

- **Η κλάση Time**

```
package DReader;

/**
*
* @author jim
*/
import java.util.*;
import java.util.Calendar;

public class Time {

private static int year,month,date;
private static String h,Smonth,Sdate,ct;
```

```
public static String getCTime(){

Calendar calendar = new GregorianCalendar();
year = calendar.get(Calendar.YEAR);
month = calendar.get(Calendar.MONTH);
date = calendar.get(Calendar.DATE);
month=month+1;
if (month<10) {
Smonth="0"+month;
}
else {
Smonth=""+month;
}
if (date<10) {
Sdate="0"+date;
}
else {
Sdate=""+date;
}
ct=""+year+Smonth+Sdate;
return ct;
}
}
```

4. Οι πηγαίοι κώδικες για κλάσεις που χρησιμποιπούνται και στις 3 οντότητες

- Η κλάση AeSimpleSHA1

```
package Reader;

/**
*
* @author jim
*/
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class AeSimpleSHA1 {

private static String convertToHex(byte[] data) {
StringBuffer buf = new StringBuffer();
for (int i = 0; i < data.length; i++) {
int halfbyte = (data[i] >>> 4) & 0x0F;
int two_halfs = 0;
do {
if ((0 <= halfbyte) && (halfbyte <= 9))
buf.append((char) ('0' + halfbyte));
```

```
else
buf.append((char) ('a' + (halfbyte - 10)));
halfbyte = data[i] & 0x0F;
} while(two_halfs++ < 1);
}
return buf.toString();
}

public static String SHA1(String text)
throws NoSuchAlgorithmException, UnsupportedEncodingException  {
MessageDigest md;
md = MessageDigest.getInstance("SHA-1");
byte[] sha1hash = new byte[40];
md.update(text.getBytes("iso-8859-1"), 0, text.length());
sha1hash = md.digest();
return convertToHex(sha1hash);
}
}
```

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/

package Reader;



/**
*
* @author jim
*/
public class Converter {

public static byte[] hexStringToByteArray(String s) {
int len = s.length();
byte[] data = new byte[len / 2];
for (int i = 0; i < len; i += 2) {
data[i / 2] = (byte) ((Character.digit(s.charAt(i), 16) << 4)
+ Character.digit(s.charAt(i+1), 16));
}
return data;
}
public static String getHexString(byte[] b) throws Exception {
String result = "";
for (int i=0; i < b.length; i++) {
```

```
result +=Integer.toString( ( b[i] & 0xff ) + 0x100, 16).substring( 1 );
}
return result;
}

public static byte[] xorByteArray (byte a[], byte b[]){
byte[] c=new byte [a.length];
for (int i=0; i<a.length; i++){
c[i]=(byte) (a[i] ^ b[i]);
}
return c;
}

public static String concatThreeStrings (String S, String F,String D){
String B="";
StringBuilder sb = new StringBuilder(40);
B= sb.append(S).append(F).append(D).toString();
return B;
}

public static String concatTwoStrings (String S, String F){
String B="";
StringBuilder sb = new StringBuilder(40);
B= sb.append(S).append(F).toString();
return B;
}
}
```

- Η κλάση RNG

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
package Reader;

/**
*
* @author jim
*/
import java.io.UnsupportedEncodingException;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
public class RNG {

static final int RANMAR = 0, RANECU = 1, RANLUX = 2, RANJAVA = 3, NULL
= 4, RANMT = 5;
static final int FLAT = 0, GAUSSIAN = 1, CHOOSE1 = 2, CHOOSE2 = 3, COIN1 =
4, COIN2 = 5;
private static double x;

public static String getRandom() throws UnsupportedEncodingException {
boolean noprint = false;
String lengthString, unrandom, random;
RandomElement e;
int i, generator, distribution, n;
int luxury;
long seed;
int number;
generator = RANLUX;
distribution = FLAT;
seed = RandomSeedable.ClockSeed();
luxury = Ranlux.maxlev;
n = 1;
e = null;
random = "";
unrandom = "";

if (generator == RANMAR) {
e = new Ranmar(seed);
} else if (generator == RANECU) {
e = new Ranecu(seed);
} else if (generator == RANLUX) {
e = new Ranlux(luxury, seed);
} else if (generator == RANJAVA) {
e = new RandomJava();
} else if (generator == RANMT) {
e = new RanMT(seed);
}


for (i = 1; i <= n; i++) {
x = 0.0;
if (generator != NULL) {
if (distribution == FLAT) {
x = e.raw();
} else if (distribution == GAUSSIAN) {
x = e.gaussian();
} else if (distribution == CHOOSE1) {
x = e.choose(5);
} else if (distribution == CHOOSE2) {
x = e.choose(7, 10);
} else if (distribution == COIN1) {
```

```
x = e.coin() ? 1.0 : 0.0;
} else if (distribution == COIN2) {
x = e.coin(0.7) ? 1.0 : 0.0;
} else {
die("Invalid distribution: " + distribution);
}
}

if (!noprint) {
x = x * 100000000;
}
number = (int) x;
lengthString = Integer.valueOf(number).toString();
if (lengthString.length() < 8) { //check the lenght of secret (8)
x = x * 1000000000;
number = (int) x;
lengthString = Integer.valueOf(number).toString();

}
unrandom = "" + number;
}
try {
random = AeSimpleSHA1.SHA1(unrandom);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
return random;

}

public static String getRandom2() throws UnsupportedEncodingException {
boolean noprint = false;
String lengthString, unrandom, random;
RandomElement e;
int i, generator, distribution, n;
int luxury;
long seed;
int number;
generator = RANMAR;
distribution = FLAT;
seed = RandomSeedable.ClockSeed();
luxury = Ranlux.lxdflt;
n = 1;
e = null;
random = "";
unrandom = "";

if (generator == RANMAR) {
```

```
e = new Ranmar(seed);
} else if (generator == RANECU) {
e = new Ranecu(seed);
} else if (generator == RANLUX) {
e = new Ranlux(luxury, seed);
} else if (generator == RANJAVA) {
e = new RandomJava();
} else if (generator == RANMT) {
e = new RanMT(seed);
}


for (i = 1; i <= n; i++) {
x = 0.0;
if (generator != NULL) {
if (distribution == FLAT) {
x = e.raw();
} else if (distribution == GAUSSIAN) {
x = e.gaussian();
} else if (distribution == CHOOSE1) {
x = e.choose(5);
} else if (distribution == CHOOSE2) {
x = e.choose(7, 10);
} else if (distribution == COIN1) {
x = e.coin() ? 1.0 : 0.0;
} else if (distribution == COIN2) {
x = e.coin(0.7) ? 1.0 : 0.0;
} else {
die("Invalid distribution: " + distribution);
}
}

if (!noprint) {
x = x * 100000000;
}
number = (int) x;
lengthString = Integer.valueOf(number).toString();
if (lengthString.length() < 8) { //check the lenght of secret (8)
x = x * 1000000000;
number = (int) x;
lengthString = Integer.valueOf(number).toString();

}
unrandom = "" + number;
}
try {
random = AeSimpleSHA1.SHA1(unrandom);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
```

```
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
return random;

}

public static String getRandom3() throws UnsupportedEncodingException {

boolean noprint = false;
String lengthString, unrandom, random;
RandomElement e;
int i, generator, distribution, n;
int luxury;
long seed;
int number;
generator = RANECU;
distribution = FLAT;
seed = RandomSeedable.ClockSeed();
luxury = Ranlux.lxdflt;
n = 1;
e = null;
random = "";
unrandom = "";

if (generator == RANMAR) {
e = new Ranmar(seed);
} else if (generator == RANECU) {
e = new Ranecu(seed);
} else if (generator == RANLUX) {
e = new Ranlux(luxury, seed);
} else if (generator == RANJAVA) {
e = new RandomJava();
} else if (generator == RANMT) {
e = new RanMT(seed);
}


for (i = 1; i <= n; i++) {
x = 0.0;
if (generator != NULL) {
if (distribution == FLAT) {
x = e.raw();
} else if (distribution == GAUSSIAN) {
x = e.gaussian();
} else if (distribution == CHOOSE1) {
x = e.choose(5);
} else if (distribution == CHOOSE2) {
x = e.choose(7, 10);
} else if (distribution == COIN1) {
x = e.coin() ? 1.0 : 0.0;
```

```
} else if (distribution == COIN2) {
x = e.coin(0.7) ? 1.0 : 0.0;
} else {
die("Invalid distribution: " + distribution);
}
}

if (!noprint) {
x = x * 100000000;
}
number = (int) x;
lengthString = Integer.valueOf(number).toString();
if (lengthString.length() < 8) { //check the lenght of secret (8)
x = x * 1000000000;
number = (int) x;
lengthString = Integer.valueOf(number).toString();

}
unrandom = "" + number;
}
try {
random = AeSimpleSHA1.SHA1(unrandom);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
return random;
}

public static String getRandom4() throws UnsupportedEncodingException {
boolean noprint = false;
String lengthString, unrandom, random;
RandomElement e;
int i, generator, distribution, n;
int luxury;
long seed;
int number;
generator = RANMT;
distribution = FLAT;
seed = RandomSeedable.ClockSeed();
luxury = Ranlux.lxdflt;
n = 1;
e = null;
random = "";
unrandom = "";


if (generator == RANMAR) {
e = new Ranmar(seed);
```

```
} else if (generator == RANECU) {
e = new Ranecu(seed);
} else if (generator == RANLUX) {
e = new Ranlux(luxury, seed);
} else if (generator == RANJAVA) {
e = new RandomJava();
} else if (generator == RANMT) {
e = new RanMT(seed);
}


for (i = 1; i <= n; i++) {
x = 0.0;
if (generator != NULL) {
if (distribution == FLAT) {
x = e.raw();
} else if (distribution == GAUSSIAN) {
x = e.gaussian();
} else if (distribution == CHOOSE1) {
x = e.choose(5);
} else if (distribution == CHOOSE2) {
x = e.choose(7, 10);
} else if (distribution == COIN1) {
x = e.coin() ? 1.0 : 0.0;
} else if (distribution == COIN2) {
x = e.coin(0.7) ? 1.0 : 0.0;
} else {
die("Invalid distribution: " + distribution);
}
}

if (!noprint) {
x = x * 100000000;
}
number = (int) x;
lengthString = Integer.valueOf(number).toString();
if (lengthString.length() < 8) { //check the lenght of secret (8)
x = x * 1000000000;
number = (int) x;
lengthString = Integer.valueOf(number).toString();

}
unrandom = "" + number;
}
try {
random = AeSimpleSHA1.SHA1(unrandom);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
```

```
}
return random;

}

public static String getRandom5() throws UnsupportedEncodingException {
boolean noprint = false;
String lengthString, unrandom, random;
RandomElement e;
int i, generator, distribution, n;
int luxury;
long seed;
int number;
generator = RANJAVA;
distribution = FLAT;
seed = RandomSeedable.ClockSeed();
luxury = Ranlux.lxdflt;
n = 1;
e = null;
random = "";
unrandom = "";


if (generator == RANMAR) {
e = new Ranmar(seed);
} else if (generator == RANECU) {
e = new Ranecu(seed);
} else if (generator == RANLUX) {
e = new Ranlux(luxury, seed);
} else if (generator == RANJAVA) {
e = new RandomJava();
} else if (generator == RANMT) {
e = new RanMT(seed);
}


for (i = 1; i <= n; i++) {
x = 0.0;
if (generator != NULL) {
if (distribution == FLAT) {
x = e.raw();
} else if (distribution == GAUSSIAN) {
x = e.gaussian();
} else if (distribution == CHOOSE1) {
x = e.choose(5);
} else if (distribution == CHOOSE2) {
x = e.choose(7, 10);
} else if (distribution == COIN1) {
x = e.coin() ? 1.0 : 0.0;
} else if (distribution == COIN2) {
```

```
x = e.coin(0.7) ? 1.0 : 0.0;
} else {
die("Invalid distribution: " + distribution);
}
}

if (!noprint) {
x = x * 100000000;
}
number = (int) x;
lengthString = Integer.valueOf(number).toString();
if (lengthString.length() < 8) { //check the lenght of secret (8)
x = x * 1000000000;
number = (int) x;
lengthString = Integer.valueOf(number).toString();

}
unrandom = "" + number;
}
try {
random = AeSimpleSHA1.SHA1(unrandom);
} catch (NoSuchAlgorithmException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
Logger.getLogger(RfidAgent.class.getName()).log(Level.SEVERE, null, ex);
}
return random;

}

public static String getRandom6() throws Exception { //secure random(optional)

String r;
Random ranGen = new SecureRandom();
byte[] Key = new byte[20]; // 20 bytes = 160 bits
ranGen.nextBytes(Key);
r = Converter.getHexString(Key);
return r;
}

public static String getRandomTagId() {
boolean noprint = false;
String lengthString,random;
RandomElement e;
int i, generator, distribution, n;
int luxury;
long seed;
int number;
generator = RANLUX;
distribution = FLAT;
```

```
seed = RandomSeedable.ClockSeed();
luxury = Ranlux.maxlev;
n = 1;
e = null;
random = "";


if (generator == RANMAR) {
e = new Ranmar(seed);
} else if (generator == RANECU) {
e = new Ranecu(seed);
} else if (generator == RANLUX) {
e = new Ranlux(luxury, seed);
} else if (generator == RANJAVA) {
e = new RandomJava();
} else if (generator == RANMT) {
e = new RanMT(seed);
}


for (i = 1; i <= n; i++) {
x = 0.0;
if (generator != NULL) {
if (distribution == FLAT) {
x = e.raw();
} else if (distribution == GAUSSIAN) {
x = e.gaussian();
} else if (distribution == CHOOSE1) {
x = e.choose(5);
} else if (distribution == CHOOSE2) {
x = e.choose(7, 10);
} else if (distribution == COIN1) {
x = e.coin() ? 1.0 : 0.0;
} else if (distribution == COIN2) {
x = e.coin(0.7) ? 1.0 : 0.0;
} else {
die("Invalid distribution: " + distribution);
}
}

if (!noprint) {
x = x * 100000000;
}
number = (int) x;
lengthString = Integer.valueOf(number).toString();
if (lengthString.length() < 8) { //check the lenght of secret (8)
x = x * 1000000000;
number = (int) x;
lengthString = Integer.valueOf(number).toString();
```

```
}
random = "" + number;
}
return random;
}

static void die(String s) {
System.err.println(s);
System.exit(-1);
}

static double nullgen() {
return 0.0;
}
}
```

- **Η κλάση RanMT (βοηθητική της RNG)**

```
public class RanMT extends RandomSeedable implements Serializable {

private static final int N = 624;
private static final int M = 397;
private static final int MATRIX_A = 0x9908b0df;   //    private static final * constant
vector a
private static final int UPPER_MASK = 0x80000000; // most significant w-r bits
private static final int LOWER_MASK = 0x7fffffff; // least significant r bits


// Tempering parameters
private static final int TEMPERING_MASK_B = 0x9d2c5680;
private static final int TEMPERING_MASK_C = 0xefc60000;

private int mt[]; // the array for the state vector
private int mti; // mti==N+1 means mt[N] is not initialized
private int mag01[];

public RanMT() {
this.setSeed(4357);
}

/*
*
* Note that this uses only the LSB 32 bits from the long
*
*/

public RanMT(long seed) {
this.setSeed(seed);
}
```

```
public RanMT(Date d) {
this.setSeed(d.getTime());
}

/**
*
* If a 32 bit seed isn't enough for you,  you can pass an array of
* 624 integers.  Any array of integers is fine so long as they
* aren't all zero.
*
*/

public RanMT(int array[]) {
this.setSeed(array);
}

private void setSeed(final long seed)
{
mt = new int[N];

mag01 = new int[2];
mag01[0] = 0x0;
mag01[1] = MATRIX_A;

mt[0]= (int)(seed & 0xffffffff);
for (mti=1; mti<N; mti++)
{
mt[mti] =
(1812433253 * (mt[mti-1] ^ (mt[mti-1] >>> 30)) + mti);
/* See Knuth TAOCP Vol2. 3rd Ed. P.106 for multiplier. */
/* In the previous versions, MSBs of the seed affect   */
/* only MSBs of the array mt[].                        */
/* 2002/01/09 modified by Makoto Matsumoto             */
mt[mti] &= 0xffffffff;
/* for >32 bit machines */
}
}

/**
* An alternative, more complete, method of seeding the
* pseudo random number generator.  array must be an
* array of 624 ints, and they can be any value as long as
* they're not *all* zero.
*/

private void setSeed(final int[] array) {
int i, j, k;
setSeed(19650218);
i=1; j=0;
k = (N>array.length ? N : array.length);
```

```
for (; k!=0; k--) {
mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >>> 30)) * 1664525)) + array[j] + j; /* non linear
*/
mt[i] &= 0xffffffff; /* for WORDSIZE > 32 machines */
i++;
j++;
if (i>=N) { mt[0] = mt[N-1]; i=1; }
if (j>=array.length) j=0;
}
for (k=N-1; k!=0; k--) {
mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >>> 30)) * 1566083941)) - i; /* non linear */
mt[i] &= 0xffffffff; /* for WORDSIZE > 32 machines */
i++;
if (i>=N) {
mt[0] = mt[N-1]; i=1;
}
}
mt[0] = 0x80000000; /* MSB is 1; assuring non-zero initial array */
}
```

```
public final double raw() {
int y;
int z;

if (mti >= N) {   // generate N words at one time

int kk;

for (kk = 0; kk < N - M; kk++) {
y = (mt[kk] & UPPER_MASK) | (mt[kk+1] & LOWER_MASK);
mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
}
for (; kk < N-1; kk++) {
y = (mt[kk] & UPPER_MASK) | (mt[kk+1] & LOWER_MASK);
mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0x1];
}
y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

mti = 0;
}

y = mt[mti++];
y ^= y >>> 11;                          // TEMPERING_SHIFT_U(y)
y ^= (y << 7) & TEMPERING_MASK_B;       // TEMPERING_SHIFT_S(y)
y ^= (y << 15) & TEMPERING_MASK_C;      // TEMPERING_SHIFT_T(y)
y ^= (y >>> 18);                        // TEMPERING_SHIFT_L(y)
```

```
if (mti >= N) {  // generate N words at one time
int kk;

for (kk = 0; kk < N - M; kk++) {
z = (mt[kk] & UPPER_MASK) | (mt[kk+1] & LOWER_MASK);
mt[kk] = mt[kk+M] ^ (z >>> 1) ^ mag01[z & 0x1];
}
for (; kk < N-1; kk++) {
z = (mt[kk] & UPPER_MASK) | (mt[kk+1] & LOWER_MASK);
mt[kk] = mt[kk+(M-N)] ^ (z >>> 1) ^ mag01[z & 0x1];
}
z = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
mt[N-1] = mt[M-1] ^ (z >>> 1) ^ mag01[z & 0x1];

mti = 0;
}

z = mt[mti++];
z ^= z >>> 11;                          // TEMPERING_SHIFT_U(z)
z ^= (z << 7) & TEMPERING_MASK_B;       // TEMPERING_SHIFT_S(z)
z ^= (z << 15) & TEMPERING_MASK_C;      // TEMPERING_SHIFT_T(z)
z ^= (z >>> 18);                        // TEMPERING_SHIFT_L(z)

/* derived from nextDouble documentation in jdk 1.2 docs, see top */
return ((((long)(y >>> 6)) << 27) + (z >>> 5)) / (double)(1L << 53);
}
}
```

- **Η κλάση RandomElement (βοηθητική της RNG)**

```
package Reader;

import java.util.*;

//
// RngPack 1.1a by Paul Houle
// http://www.honeylocust.com/RngPack/
//

/**
*   RandomElement is an abstract class that encapsulates random number
* generators.  To base a class on it,  you must define the method
* <CODE>raw()</CODE> as described below.  It is also likely that you
* will want to define a constructor or another mechanism for seeding the
* the generator.  The other classes defined in <CODE>RandomElement</CODE>
* add value to the numbers generated by <CODE>raw()</CODE>
*
```

```
* <P>
* <A HREF="/RngPack/src/edu/cornell/lassp/houle/RngPack/RandomElement.java">
* Source code </A> is available.
*
* @author <A HREF="http://www.honeylocust.com/"> Paul Houle </A> (E-mail:
<A HREF="mailto:paul@honeylocust.com">paul@honeylocust.com</A>)
* @version 1.1a
*
* @see RandomJava
* @see RandomShuffle
*/

public abstract class RandomElement extends Object implements Cloneable {

   Double BMoutput;              // constant needed by Box-Mueller algorithm

/**
 * The abstract method that must be defined to make a working RandomElement.
 * See the class <CODE>RandomJava</CODE> for an example of how to do this.
 *
 * @see RandomJava
 *
 * @return a random double in the range [0,1]
 */

   abstract public double raw();


   public void raw(double d[],int n) {
         for(int i=0;i<n;i++)
            d[i]=raw();
   }


/**
* Fill an entire array with doubles.  This method calls <CODE>raw(double d[],int
n)</CODE>
* with <CODE>d=d.length</CODE>.  Since this adds little overhead for
<CODE>d.length</CODE>
* large,  it is only necessary to override <CODE>raw(double d[],int n)</CODE>
*
* @param d array to be filled with doubles.
*/

   public void raw(double d[]) {
         raw(d,d.length);
   }

/**
@param hi upper limit of range
```

```
@return a random integer in the range 1,2,... ,<STRONG>hi</STRONG>
*/

   public int choose(int hi) {
         return choose(1,hi);
   }

/**
 * @param lo lower limit of range
 * @param hi upper limit of range
 * @return a random integer in the range <STRONG>lo</STRONG>,
<STRONG>lo</STRONG>+1, ... ,<STRONG>hi</STRONG>
 */

   public int choose(int lo,int hi) {
         int value=lo + (int) ((hi-lo)*raw());
         if (value>hi)
            value=hi;         /* otherwise a generator on [0,1] could return
                        a result outside of the legal range:  you
                              can override this if you know a generator
                              returns [0,1) or (0,1). */
         return value;
   }

/**
@return a boolean that's true 0.5 of the time;  equivalent to coin(0.5).
*/

   public boolean coin() {
         return raw()<=0.5;
   }

/**
@param p probability that function will return true
@return a boolean that's true p of the time.
*/

   public boolean coin(double p) {
         return raw()<=p;
   }

/**
@param lo lower limit of range
@param hi upper limit of range
@return a uniform random real in the range
[<STRONG>lo</STRONG>,<STRONG>hi</STRONG>]
*/
   public double uniform(double lo,double hi) {
         return (lo+(hi-lo)*raw());
   }
```

```
/**
gaussian() uses the Box-Muller algorithm to transform raw()'s into
gaussian deviates.

@return a random real with a gaussian distribution,  standard deviation

*/

   public double gaussian() {
         double out,x,y,r,z;

         if (BMoutput != null)
            {
                  out = BMoutput.doubleValue();
                  BMoutput = null;
                  return(out);
            };

         do {
            x=uniform(-1,1);
            y=uniform(-1,1);
            r=x*x+y*y;
         } while (r >= 1.0);

         z=Math.sqrt(-2.0*Math.log(r)/r);
         BMoutput=new Double(x*z);
         return(y*z);
   }


/**
*
* @param sd standard deviation
* @return a gaussian distributed random real with standard deviation
<STRONG>sd</STRONG>
*/

   public double gaussian(double sd) {
         return(gaussian()*sd);
   }

/**
*
* generate a power-law distribution with exponent <CODE>alpha</CODE>
* and lower cutoff
* <CODE>cut</CODE>
* <CENTER>
* </CENTER>
*
```

```
*@param alpha the exponent
*@param cut the lower cutoff
*
*/

   public double powlaw(double alpha,double cut) {
         return cut*Math.pow(raw(), 1.0/(alpha+1.0) ) ;
   }

   public Object clone() throws java.lang.CloneNotSupportedException {
         return super.clone();
   }

};
```

- **Η κλάση RandomJava (βοηθητική της RNG)**

```
package Reader;
import java.util.*;

//
// RngPack 1.1a by Paul Houle
// http://www.honeylocust.com/~houle/RngPack/
//

/**
* RandomJava is a class wrapper for the <CODE>Math.random()</CODE>
* generator that comes with Java.  I know nothing about the
* quality of <CODE>Math.random()</CODE>, but I will warn the
* reader that system-supplied
* RNGs have a bad reputation;  <TT>RandomJava</TT> is <B>NOT</B>
* reccomended for general use,  it has only been included as a
* straightforward example of how to
* build a <CODE>RandomElement</CODE> wrapper for an existing RNG.
* The <TT>RANMAR</TT>, <TT>RANECU</TT> and <TT>RANLUX</TT>
* generators included in this package all appear to be faster than
* Math.random();  all three are well-studied,  portable and
* proven in use.
*
* <P>
* <A HREF="/RngPack/src/edu/cornell/lassp/houle/RngPack/RandomJava.java">
* Source code </A> is available.
*
* @author <A HREF="http://www.honeylocust.com/~houle/RngPack/"> Paul Houle
</A> (E-mail: <A
HREF="mailto:paul@honeylocust.com">paul@honeylocust.com</A>)
* @version 1.1a
* @see Ranmar
```

```
* @see Ranlux
* @see Ranecu
*/

public class RandomJava extends RandomElement {

/**
* Wrapper for <CODE>Math.random().</CODE>
@see RandomElement#raw
*/
  public double raw() {
        return Math.random();
  };

};
```

- **Η κλάση RandomSeedable (βοηθητική της RNG)**

```
package Reader;

import java.util.*;

//
// RngPack 1.1a by Paul Houle
// http://www.honeylocust.com/~houle/RngPack/
//

/**
*
* <CODE>RandomSeedable</CODE> is an abstract class that extends the
* <CODE>RandomElement</CODE> class to include the ability to
* automatically generate a valid <CODE>long</CODE> seed from the clock.
* Thus it provides a consistent interface for seeding interchangable
* generators.  It is reccomended that a <CODE>RandomSeedable</CODE> have
* a constructor that takes a <CODE>long</CODE> for a seed.  For example,
* if you write a generator called <CODE>ReallyRandom</CODE>,  you want
* to be able to do
*
* <PRE>
* long seed=ReallyRandom.ClockSeed();
* RandomSeedable e=new ReallyRandom(seed);
* </PRE>
*
* this makes it convenient to keep a copy of the seed in case you want
* to restart the generator with the same seed some time in the future.
*
* <P>
* If one is going to use a long to generate a smaller seed by taking
* <CODE>Clockseed()</CODE> modulus another number,  we reccomend that
```

```
* you use a prime number;   this ensures that the generator would have
* the maximum "period" if it were started at regular issues,  for
* instance,  by a batch job.   See <CODE>Ranmar</CODE> for an
* example.
*
* <P>
* <A
HREF="/RngPack/src/edu/cornell/lassp/houle/RngPack/RandomSeedable.java">
* Source code </A> is available.
*
* @author <A HREF="http://www.honeylocust.com/"> Paul Houle </A> (E-mail:
<A HREF="mailto:paul@honeylocust.com">paul@honeylocust.com</A>)
* @version 1.1a
*
* @see Ranecu
* @see Ranlux
* @see Ranmar
*/

public abstract class RandomSeedable extends RandomElement {

/**
*
* Return a long integer seed given a date
*
* @param d a date
* @return a long integer seed
*
*/
  public static long ClockSeed(Date d)
  {
    return d.getTime();
  };

/**
*
* Return a long integer seed calculated from the date.  Equivalent to
* <CODE>ClockSeed(new Date());
*
* @return a long integer seed
*
*/

  public static long ClockSeed()
  {
    return ClockSeed(new Date());
  };
};
```

- **Η κλάση RandomShuffle (βοηθητική της RNG)**

```
package Reader;

//
// RngPack 1.1a by Paul Houle
// http://www.honeylocust.com/RngPack/
//

/**
* RandomShuffle uses one random number generator to shuffle the numbers
* produced by another to obliterate sequential correlations.
* To initialize a RandomShuffle,  pass it two RandomElements.  The
* first RandomElement is used to generate a table of random numbers
* and the second is used to choose one from the table.  An example of
* usage is,
*
* <PRE>
* RandomElement markov=new RandomShuffle(new Ranecu(),new Ranmar(),32)
* </PRE>
*
* which would generate a deck of 32 numbers from <TT>RANECU</TT> and
* use <TT>RANMAR</TT> to choose from the deck.
*
* <BR>
* <B>References:</B>
* <UL>
* <LI> F. James; <CITE>Comp. Phys. Comm.</CITE> <STRONG>60</STRONG>
(1990) p 329-344
* <LI> D. Knuth; <CITE>The Art of Computer Programming</CITE> vol. 2, sec
3.2.2
* </UL>
*
* <P>
* <A HREF="/RngPack/src/edu/cornell/lassp/houle/RngPack/RandomShuffle.java">
* Source code </A> is available.
*
* @author <A HREF="http://www.honeylocust.com/"> Paul Houle </A> (E-mail:
<A HREF="paul@honeylocust.com">paul@honeylocust.com</A>)
* @version 1.1a
*/

public class RandomShuffle extends RandomElement {

  RandomElement generatorA,generatorB;
  int decksize;
  double deck[];

/**
* @param ga generator to fill shuffle deck
```

```
* @param gb geberator to choose from shuffle deck
* @param ds the size of the shuffle deck
*/

  public RandomShuffle(RandomElement ga,RandomElement gb,int ds) {

        generatorA=ga;
      generatorB=gb;
        decksize=ds;

        stackdeck();

};

/**
* The generator.
*
* @see RandomElement#raw
*/

public double raw() {
      double random;
        int i;

        i=generatorB.choose(0,decksize-1);
      random=deck[i];
      deck[i]=generatorA.raw();

        return random;
};

private void stackdeck() {

      int i;
        deck = new double[decksize];

        for(i=0;i<decksize;i++)
                deck[i]=generatorA.raw();
};

};
```

- **Η κλάση RandomSynchronized (βοηθητική της RNG)**

```
package Reader;

import java.util.*;
import java.io.Serializable;
```

```
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.io.ObjectInputStream;

//
// RngPack 1.1a by Paul Houle
// http://www.honeylocust.com/RngPack/
//

/**
* RandomSynchronized is a wrapper class that makes a random number generator
* safe for multi-threaded operation by serializing access in time.  For
* high-performance applications,  it is better for each thread to have it's
* own random number generator.
*
* Note this class is declared serializable,  but serialization won't be
* successful if it's wrapping a non-serializable generator.
* <P>
* <A
HREF="/RngPack/src/edu/cornell/lassp/houle/RngPack/RandomSynchronized.java">
* Source code </A> is available.
*
* @author <A HREF="http://www.honeylocust.com/"> Paul Houle </A> (E-mail:
<A HREF="mailto:paul@honeylocust.com">paul@honeylocust.com</A>)
* @version 1.1a
*
* @see RandomElement
*/

public abstract class RandomSynchronized extends RandomElement implements
Serializable {

   private RandomElement rng;

   public RandomSynchronized(RandomElement rng) {
         this.rng=rng;
   }

   /**
    * Synchronized the raw() method,  which is generally not threadsafe.
    *
    * @see RandomJava
    *
    * @return a random double in the range [0,1]
    */

   synchronized public double raw() {
         return rng.raw();
   }
```

```
/**
 * This method probably isn't threadsafe in implementations,  so it's
 * synchronized
 *
 *
 * @param d array to be filled with doubles
 * @param n number of doubles to generate
 */

   synchronized public void raw(double d[],int n) {
         rng.raw(d,n);
   }

/**
 *
 * Wrapped so generators can override.
 *
 * @param lo lower limit of range
 * @param hi upper limit of range
 * @return a random integer in the range <STRONG>lo</STRONG>,
<STRONG>lo</STRONG>+1, ... ,<STRONG>hi</STRONG>
 *
 */


   synchronized public int choose(int lo,int hi) {
         return rng.choose(lo,hi);
   }

/**
 *
 * Must be synchronized because state is stored in BMoutput
 *
 * @return a random real with a gaussian distribution,  standard deviation
 *
 */

   synchronized public double gaussian() {
         return rng.gaussian();
   }

/**
 *
 * We wouldn't want some sneaky person to serialize this in the middle
 * of generating a number
 *
 */


   private synchronized void writeObject(final ObjectOutputStream out)
```

```
        throws IOException {
        // just so we're synchronized.
        out.defaultWriteObject();
    }

    private synchronized void readObject (final ObjectInputStream in)
        throws IOException, ClassNotFoundException {
        // just so we're synchronized.
        in.defaultReadObject();
    }
}
```

- **Η κλάση Ranecu (βοηθητική της RNG)**

```
package Reader;
import java.util.*;
import java.io.Serializable;

//
// RngPack 1.1a by Paul Houle
// http://www.honeylocust.com/RngPack/
//

/**
*
* Ranecu is an advanced multiplicative linear congruential random number
* generator with a period of aproximately 10<SUP>18</SUP>.
* Ranecu is a direct translation from Fortran of the <B>RANECU</B>
* subroutine
* published in the paper
* <BR>
* F. James, <CITE>Comp. Phys. Comm.</CITE> <STRONG>60</STRONG>
(1990) p 329-344
* <BR>
* The algorithm was originally described in
* <BR>
* P. L'Ecuyer, <CITE>Commun. ACM.</CITE> <STRONG>1988</STRONG>
(1988) p 742
* <BR>
*
* <P>
* <A HREF="/RngPack/src/edu/cornell/lassp/houle/RngPack/Ranecu.java">
* Source code </A> is available.
*
* @author <A HREF="http://www.honeylocust.com/"> Paul Houle </A> (E-mail:
<A HREF="mailto:paul@honeylocust.com">paul@honeylocust.com</A>)
* @version 1.1a
*/
```

```
public class Ranecu extends RandomSeedable implements Serializable {

int iseed1,iseed2;

/**
* default iseed1 = 12345
*/
public static int DEFSEED1=12345;

/**
* default iseed2 = 67890
*/

public static int DEFSEED2=67890;

/**
*
* Initialize <BOLD>RANECU</BOLD> with the default seeds from
* James.
*
*/
public Ranecu() {

   iseed1=DEFSEED1;
   iseed2=DEFSEED2;
};

/**
*
* Initialize <BOLD>RANECU</BOLD> with two specified integer seeds.  Use
* this to introduce repeatable seeds.  Equivalent to
*
* <CODE>Ranecu(s1*(long) Integer.MAX_VALUE)+s2)</CODE>
*
* @param s1 seed integer 1 (MSW)
* @param s2 seed integer 2 (LSW)
*
*/

public Ranecu(int s1,int s2){

   iseed1=s1;
   iseed2=s2;
};

/*
*
* Initialize <TT>RANECU</TT> with a long seed.
*
* @param l long integer seed
```

```
*/

public Ranecu(long l) {

   iseed1 = (int) l/Integer.MAX_VALUE;
   iseed2 = (int) l%Integer.MAX_VALUE;
};

/*
*
* Initialize <TT>RANECU</TT> from the clock without saving a copy
* of the seed.  Example:
*
* <PRE>
* RandomElement e = new Ranecu(new Date());
* </PRE>
*
* to save the seed for future restarts,  see the <CODE>ClockSeed()</CODE>
* method defined in RandomSeedable.
*
* @param d a date,  typically <CODE>new Date()</CODE>
* @see RandomSeedable#ClockSeed()
*/

public Ranecu(Date d) {
   iseed1 = (int) d.getTime()/Integer.MAX_VALUE;
   iseed2 = (int) d.getTime()%Integer.MAX_VALUE;
};

/**

@see RandomElement#raw
*/

final public double raw() {

   int k,iz;

   k=iseed1/53668;
   iseed1=40014*(iseed1-k*53668)-k*12211;
   if (iseed1<0) iseed1=iseed1+2147483563;

   k=iseed2/52774;
   iseed2=40692*(iseed2-k*52774)-k*3791;
   if (iseed2<0) iseed2=iseed2+2147483399;

   iz=iseed1-iseed2;
   if(iz<1) iz=iz+2147483562;

   return(iz*4.656613e-10);
```

```
};

/**
* This is an inline version that returns an array of doubles for speed.
*/


final public void raw(double d[],int n)
{
   int i,k,iz;

   for(i=0;i<n;i++)
   {

      k=iseed1/53668;
      iseed1=40014*(iseed1-k*53668)-k*12211;
      if (iseed1<0) iseed1=iseed1+2147483563;

      k=iseed2/52774;
      iseed2=40692*(iseed2-k*52774)-k*3791;
      if (iseed2<0) iseed2=iseed2+2147483399;

      iz=iseed1-iseed2;
      if(iz<1) iz=iz+2147483562;

      d[i]=iz*4.656613e-10;
   };
};


/**
*
* @return the current generator state as a long.  Can be used to
* restart the generator where one left off.
*
*/

public long getSeed() { return iseed1*(long) Integer.MAX_VALUE + iseed2; };

};
```

- **Η κλάση Ranecu (βοηθητική της RNG)**

```
package Reader;
import java.util.*;
import java.io.Serializable;

//
// RngPack 1.1a by Paul Houle
```

```
// http://www.honeylocust.com/RngPack/
//

/**
*
*     <TT>RANLUX</TT> is an advanced pseudo-random number generator
based on the
*     <TT>RCARRY</TT> algorithm proposed in 1991 by Marsaglia and Zaman.
*     <TT>RCARRY</TT>
*     used a subtract-and-borrow algorithm with a period on the order of
*     10<SUP>171</SUP> but still had detectable correlations between
*     numbers.  Martin Luescher proposed the <TT>RANLUX</TT>
*     algorithm in 1993;  <TT>RANLUX</TT> generates pseudo-random numbers
using
*     <TT>RCARRY</TT> but throws away numbers to destroy correlations.
*     Thus RANLUX trades execution speed for quality;  by choosing a
*     larger luxury setting one gets better random numbers slower.
*     By the tests availible at the time it was proposed,
*     <TT>RANLUX</TT> at the default luxury setting appears to be a
*     significant advance quality over previous
*     generators.
*
* <BR>
* <BR>
* <CENTER>
* <TABLE BORDER WIDTH=80%>
* <TR>
* <TD ALIGN=center COLSPAN=3>
* <A NAME="luxury"><FONT SIZE=+2>LUXURY LEVELS</FONT></A>
* </TD>
* </TR>
* <TR>
* <TD>level</TD><TD ALIGN=center>p</TD><TD><BR></TD>
* </TR>
* <TR><TD ALIGN=center>0</TD> <TD ALIGN="center">24</TD>
* <TD>equivalent to the original <TT>RCARRY</TT> of Marsaglia
* and Zaman, very long period, but fails many tests. </TD></TR>
* <TR>
* <TD ALIGN=center>1</TD><TD ALIGN=center>48</TD><TD>considerable
improvement in quality over level 0,
* now passes the gap test, but still fails spectral test.</TD></TR>
* <TR>
* <TD ALIGN=center>2</TD><TD ALIGN=center>97</TD><TD> passes all
known tests, but theoretically still
* defective.</TD></TR><TR BGCOLOR="#FFA0A0">
* <TD ALIGN=center BGCOLOR="#FFA0A0">3</TD><TD
ALIGN=center>223</TD><TD>
* DEFAULT VALUE.  Any theoretically possible
* correlations have very small chance of being observed.</TD></TR><TR>
```

```
* <TD ALIGN=center>4<TD ALIGN=center>389</TD><TD>highest possible
luxury, all 24 bits chaotic.</TD></TR>
*
* </TABLE>
* </CENTER>
* <BR>
* <CENTER><FONT SIZE=+1>
* <B>VALIDATION</B></FONT>
* </CENTER>
*
* The Java version of <TT>RANLUX</TT> has been verified against published
* values of numbers 1-5 and 101-105 produced by the reference implementation
* of <TT>RANLUX</TT> for the following initial conditions:
*
* <UL>
* <LI> Default initialization:  <CODE>Ranlux()</CODE>
* <LI> Initialization with:     <CODE>Ranlux(0,0)</CODE>
* <LI> Initialization with:     <CODE>Ranlux(389,1)</CODE>
* <LI> Initialization with:     <CODE>Ranlux(75,0)</CODE>
* </UL>
*  References:
* <UL>
* <LI>
*  M. Luscher, <CITE> Computer Physics Communications</CITE>  <B>79</B>
(1994) 100
* <LI>
*  F. James, <CITE>Computer Physics Communications</CITE> <B>79</B>
(1994) 111
* <LI><A HREF="http://www.mpa-
garching.mpg.de/~tomek/htmls/refs/ranlux.about.html">About
<TT>RANLUX</TT> random number generator:  Excerpts from discussion in
the Usenet news groups</A>
* <LI><A HREF="http://www.mpa-
garching.mpg.de/~tomek/htmls/refs/ranlux.f90_2.html">Miller's FORTRAN 90
implementation of <TT>RANLUX</TT> with test code</A>

* </UL>
*
*
* <P>
* <A HREF="/RngPack/src/edu/cornell/lassp/houle/RngPack/Ranlux.java">
* Source code </A> is available.
*
* @author <A HREF="http://www.honeylocust.com/"> Paul Houle </A> (E-mail:
<A HREF="mailto:paul@honeylocust.com">paul@honeylocust.com</A>)
* @version 1.1a
*/


public class Ranlux extends RandomSeedable implements Serializable {
```

```
/**
* Maximum luxury level: <CODE>maxlev=4</CODE>
*/
public static final int maxlev=4;

/**
* Default luxury level:  <CODE>lxdflt=3</CODE>
*/

public static final int lxdflt=3;

static final int igiga = 1000000000, jsdflt = 314159265;
static final int twop12 = 4096,itwo24 = 1 << 24,icons = 2147483563;
static final int ndskip[]={0,24,73,199,365};

int iseeds[],isdext[],next[];
int luxlev=lxdflt,nskip,inseed,jseed;
int in24 = 0, kount = 0, mkount = 0, i24 = 24, j24 = 10;
float seeds[],carry= (float) 0.0,twom24,twom12;

boolean diagOn=false;


/**
* Default initialization of <TT>RANLUX</TT>.  Uses default seed
* <CODE>jsdflt=314159265</CODE> and luxury level 3.
*/

public Ranlux() {

init_arrays();
rluxdef();
};

/**
* Initialize <TT>RANLUX</TT> with specified <A HREF="#luxury">luxury
level</A>
* and seed.
*
* @param lux <A HREF="#luxury">luxury level</A> from 0-4.
* @param ins seed,  a positive integer.
*
*/

public Ranlux(int lux,int ins) {

init_arrays();
rluxgo(lux,Math.abs(ins));
};
```

```
/**
* Initialize <TT>RANLUX</TT> with specified <A HREF="#luxury">luxury
level</A>
* and seed.
*
* @param lux <A HREF="#luxury">luxury level</A> from 0-4.
* @param ins seed,  a positive long.
*
*/

public Ranlux(int lux,long ins) {

init_arrays();
rluxgo(lux,Math.abs((int) (ins%Integer.MAX_VALUE)));
};

/**
*
* Initialize <TT>RANLUX</TT> with default <A HREF="#luxury">luxury
level</A>
* and a specified seed.
*
* @param ins seed,  a positive integer
*/

public Ranlux(int ins) {
init_arrays();
rluxgo(lxdflt,Math.abs(ins));
};

/**
*
* Initialize <TT>RANLUX</TT> with default <A HREF="#luxury">luxury
level</A>
* and a specified seed.
*
* @param ins seed,  a positive integer
*/

public Ranlux(long ins) {
init_arrays();
rluxgo(lxdflt,Math.abs((int) (ins%Integer.MAX_VALUE)));
};

/**
*
* Initialize <TT>RANLUX</TT> with specified <A HREF="#luxury">luxury
level</A>
* and a Date object.  Can be used to conveniently initialize <TT>RANLUX</TT>
```

```
* from the clock,
*
* <PRE>
* RandomElement e = Ranlux(4,new Date());
* </PRE>
*
* @param lux <A HREF="#luxury">luxury</A> level from 0-4.
* @param d date used to generate seed
*
*/

public Ranlux(int lux,Date d) {
init_arrays();
rluxgo(lux,(int) (ClockSeed(d)%Integer.MAX_VALUE) );
};

/**
*
* Initialize <TT>RANLUX</TT> with default <A HREF="#luxury">luxury
level</A>
* and a Date object.  Can be used to conveniently initialize <TT>RANLUX</TT>
* from the clock,
*
* <PRE>
* RandomElement e = Ranlux(new Date());
* </PRE>
*
* @param d date used to generate seed
*
*/

public Ranlux(Date d) {
init_arrays();
rluxgo(lxdflt,(int) (ClockSeed(d)%Integer.MAX_VALUE) );
};

/**
* Turns diagnostic messages on and off.  If <TT>setDiag(true)</TT> is
* called,  <TT>RANLUX</TT> will print diagnostic information to
* <TT>System.err</TT>
*
* @param b diagnostic message status
*/

public void setDiag(boolean b) {
diagOn=b;
};

/**
*
```

```
* The random number generator.
*
* @returns a pseudo-random double in the range (0,1)
*/

public final double raw() {

int i,k,lp;
float uni,out;

uni=seeds[j24]-seeds[i24]-carry;
if (uni < (float) 0.0) {
uni=uni+ (float) 1.0;
carry = twom24;
} else carry = (float) 0.0;

seeds[i24]=uni;

i24=next[i24];
j24=next[j24];

out=uni;

if (uni<twom12)
out += twom24*seeds[j24];

/* zero is forbidden in case user wants logarithms */

if (out==0.0) out = twom24*twom24;

in24++;

if(in24 == 24) {
in24=0;
kount += nskip;
for(i=1;i<=nskip;i++) {
uni=seeds[j24]-seeds[i24]-carry;
if (uni < (float) 0.0) {
uni=uni+ (float) 1.0;
carry = twom24;
} else carry = (float) 0.0;

seeds[i24]=uni;

i24=next[i24];
j24=next[j24];
}
}

kount++;
```

```
if (kount>=igiga) {
mkount++;
kount -= igiga;
};

return out;
};

private void init_arrays() {

/*
*
* converted from fortran:  fortran arrays start at 1,  java arrays start at
* 0.  Here we take the low road to compatibility...  We declare arrays that
* are one bigger than the fortran code.  This wastes three ints and a double;
* If you're porting this to a Commodore 64 you might need the space.
*
*/

iseeds = new int[24+1];
isdext = new int[25+1];
next   = new int[24+1];
seeds  = new float[24+1];
};

private void rluxdef()
{

int lp,i,k;

jseed = jsdflt;
inseed = jseed;
diag("RANLUX DEFAULT INITIALIZATION: "+jseed);
luxlev = lxdflt;
nskip = ndskip[luxlev];
lp = nskip + 24;
in24 = 0;
kount = 0;
mkount = 0;
diag("RANLUX DEFAULT LUXURY LEVEL =  "+luxlev+"    p = "+lp);
twom24 = (float) 1.0;

for(i = 1;i <= 24;i++)
{
twom24 = twom24 * (float) 0.5;
k = jseed / 53668;
jseed = 40014 * (jseed-k*53668) - k * 12211;
if (jseed<0) jseed = jseed + icons;
iseeds[i] = jseed % itwo24;
};
```

```
twom12 = twom24 * (float) 4096.0;

for(i = 1 ; i<=24;i++)
{
seeds[i] = iseeds[i] * twom24;
next[i] = i - 1;
};

next[1] = 24;
i24 = 24;
j24 = 10;
carry = (float) 0.0;
if (seeds[24] == 0.0) carry = twom24;

};

private final void rluxgo(int lux,int ins)
{
int ilx, i, iouter, isk, k, inner, izip, izip2;
float uni;

if (lux<00)
{
luxlev = lxdflt;
}
else if (lux<=maxlev)
{
luxlev = lux;
}
else if (lux<24 || lux>2000)
{
luxlev = maxlev;
diag("RANLUX ILLEGAL LUXURY RLUXGO: "+lux);
} else
{
luxlev = lux;
for(ilx=0;ilx<=maxlev;ilx++)
if (lux == ndskip[ilx]+24)
luxlev=ilx;

};

if (luxlev <= maxlev) {
nskip = ndskip[luxlev];
diag("RANLUX LUXURY LEVEL SET BY RLUXGO : "+luxlev+" P=
"+(nskip+24));
} else {
nskip = luxlev-24;
diag("RANLUX P-VALUE SET BY RLUXGO TO: "+luxlev);
```

```
};

in24=0;

if (ins<0)
diag("Illegal initialization by RLUXGO, negative input seed");

if (ins>0) {
jseed = ins;
diag("RANLUX INITIALIZED BY RLUXGO FROM SEED "+jseed);
} else
{

jseed=jsdflt;
diag("RANLUX INITIALIZED BY RLUXGO FROM DEFAULT SEED");
};

inseed = jseed;
twom24 = (float) 1.0;

for(i=1;i<=24;i++) {
twom24 = twom24 * (float) 0.5;
k = jseed / 53668;
jseed = 40014 * (jseed-k*53668) - k * 12211;
if (jseed < 0) jseed = jseed + icons;
iseeds[i] = jseed % itwo24;
};

twom12=twom24*4096;

for(i=1;i<=24;i++) {
seeds[i] = iseeds[i] * twom24;
next[i] = i - 1;
};

next[1]=24;
i24=24;
j24=10;
carry = (float) 0.0;

if (seeds[24] == 0.0) carry = twom24;

kount = 0;
mkount = 0;

};

private void diag(String s)
{
if (diagOn)
```

```
System.err.println(s);
};

};
```

- **Η κλάση Ranlux (βοηθητική της RNG)**

```
package Reader;
import java.util.*;
import java.io.Serializable;

//
// RngPack 1.1a by Paul Houle
// http://www.honeylocust.com/RngPack/
//

/**
*
*     <TT>RANLUX</TT> is an advanced pseudo-random number generator
based on the
*     <TT>RCARRY</TT> algorithm proposed in 1991 by Marsaglia and Zaman.
*     <TT>RCARRY</TT>
*     used a subtract-and-borrow algorithm with a period on the order of
*     10<SUP>171</SUP> but still had detectable correlations between
*     numbers.  Martin Luescher proposed the <TT>RANLUX</TT>
*     algorithm in 1993;  <TT>RANLUX</TT> generates pseudo-random numbers
using
*     <TT>RCARRY</TT> but throws away numbers to destroy correlations.
*     Thus RANLUX trades execution speed for quality;  by choosing a
*     larger luxury setting one gets better random numbers slower.
*     By the tests availible at the time it was proposed,
*     <TT>RANLUX</TT> at the default luxury setting appears to be a
*     significant advance quality over previous
*     generators.
*
* <BR>
* <BR>
* <CENTER>
* <TABLE BORDER WIDTH=80%>
* <TR>
* <TD ALIGN=center COLSPAN=3>
* <A NAME="luxury"><FONT SIZE=+2>LUXURY LEVELS</FONT></A>
* </TD>
* </TR>
* <TR>
* <TD>level</TD><TD ALIGN=center>p</TD><TD><BR></TD>
```

* </TR>
* <TR><TD ALIGN=center>0</TD> <TD ALIGN="center">24</TD>
* <TD>equivalent to the original <TT>RCARRY</TT> of Marsaglia
* and Zaman, very long period, but fails many tests. </TD></TR>
* <TR>
* <TD ALIGN=center>1</TD><TD ALIGN=center>48</TD><TD>considerable improvement in quality over level 0,
* now passes the gap test, but still fails spectral test.</TD></TR>
* <TR>
* <TD ALIGN=center>2</TD><TD ALIGN=center>97</TD><TD> passes all known tests, but theoretically still
* defective.</TD></TR><TR BGCOLOR="#FFA0A0">
* <TD ALIGN=center BGCOLOR="#FFA0A0">3</TD><TD ALIGN=center>223</TD><TD>
* DEFAULT VALUE. Any theoretically possible
* correlations have very small chance of being observed.</TD></TR><TR>
* <TD ALIGN=center>4<TD ALIGN=center>389</TD><TD>highest possible luxury, all 24 bits chaotic.</TD></TR>
*
* </TABLE>
* </CENTER>
*
* <CENTER><FONT SIZE=+1>
* <B>VALIDATION</B></FONT>
* </CENTER>
*
* The Java version of <TT>RANLUX</TT> has been verified against published
* values of numbers 1-5 and 101-105 produced by the reference implementation
* of <TT>RANLUX</TT> for the following initial conditions:
*
* <UL>
* <LI> Default initialization: <CODE>Ranlux()</CODE>
* <LI> Initialization with: <CODE>Ranlux(0,0)</CODE>
* <LI> Initialization with: <CODE>Ranlux(389,1)</CODE>
* <LI> Initialization with: <CODE>Ranlux(75,0)</CODE>
* </UL>
* References:
* <UL>
* <LI>
* M. Luscher, <CITE> Computer Physics Communications</CITE> <B>79</B> (1994) 100
* <LI>
* F. James, <CITE>Computer Physics Communications</CITE> <B>79</B> (1994) 111
* <LI><A HREF="http://www.mpa-garching.mpg.de/~tomek/htmls/refs/ranlux.about.html">About <TT>RANLUX</TT> random number generator: Excerpts from discussion in the Usenet news groups</A>

```
* <LI><A HREF="http://www.mpa-
garching.mpg.de/~tomek/htmls/refs/ranlux.f90_2.html">Miller's FORTRAN 90
implementation of <TT>RANLUX</TT> with test code</A>

* </UL>
*
*
* <P>
* <A HREF="/RngPack/src/edu/cornell/lassp/houle/RngPack/Ranlux.java">
* Source code </A> is available.
*
* @author <A HREF="http://www.honeylocust.com/"> Paul Houle </A> (E-mail:
<A HREF="mailto:paul@honeylocust.com">paul@honeylocust.com</A>)
* @version 1.1a
*/


public class Ranlux extends RandomSeedable implements Serializable {

/**
* Maximum luxury level: <CODE>maxlev=4</CODE>
*/
public static final int maxlev=4;

/**
* Default luxury level:  <CODE>lxdflt=3</CODE>
*/

public static final int lxdflt=3;

static final int igiga = 1000000000, jsdflt = 314159265;
static final int twop12 = 4096,itwo24 = 1 << 24,icons = 2147483563;
static final int ndskip[]={0,24,73,199,365};

int iseeds[],isdext[],next[];
int luxlev=lxdflt,nskip,inseed,jseed;
int in24 = 0, kount = 0, mkount = 0, i24 = 24, j24 = 10;
float seeds[],carry= (float) 0.0,twom24,twom12;

boolean diagOn=false;


/**
* Default initialization of <TT>RANLUX</TT>.  Uses default seed
* <CODE>jsdflt=314159265</CODE> and luxury level 3.
*/

public Ranlux() {

init_arrays();
```

```
rluxdef();
};

/**
* Initialize <TT>RANLUX</TT> with specified <A HREF="#luxury">luxury
level</A>
* and seed.
*
* @param lux <A HREF="#luxury">luxury level</A> from 0-4.
* @param ins seed,  a positive integer.
*
*/

public Ranlux(int lux,int ins) {

init_arrays();
rluxgo(lux,Math.abs(ins));
};

/**
* Initialize <TT>RANLUX</TT> with specified <A HREF="#luxury">luxury
level</A>
* and seed.
*
* @param lux <A HREF="#luxury">luxury level</A> from 0-4.
* @param ins seed,  a positive long.
*
*/

public Ranlux(int lux,long ins) {

init_arrays();
rluxgo(lux,Math.abs((int) (ins%Integer.MAX_VALUE)));
};

/**
*
* Initialize <TT>RANLUX</TT> with default <A HREF="#luxury">luxury
level</A>
* and a specified seed.
*
* @param ins seed,  a positive integer
*/

public Ranlux(int ins) {
init_arrays();
rluxgo(lxdflt,Math.abs(ins));
};

/**
```

```
*
* Initialize <TT>RANLUX</TT> with default <A HREF="#luxury">luxury
level</A>
* and a specified seed.
*
* @param ins seed,  a positive integer
*/

public Ranlux(long ins) {
init_arrays();
rluxgo(lxdflt,Math.abs((int) (ins%Integer.MAX_VALUE)));
};

/**
*
* Initialize <TT>RANLUX</TT> with specified <A HREF="#luxury">luxury
level</A>
* and a Date object.  Can be used to conveniently initialize <TT>RANLUX</TT>
* from the clock,
*
* <PRE>
* RandomElement e = Ranlux(4,new Date());
* </PRE>
*
* @param lux <A HREF="#luxury">luxury</A> level from 0-4.
* @param d date used to generate seed
*
*/

public Ranlux(int lux,Date d) {
init_arrays();
rluxgo(lux,(int) (ClockSeed(d)%Integer.MAX_VALUE) );
};

/**
*
* Initialize <TT>RANLUX</TT> with default <A HREF="#luxury">luxury
level</A>
* and a Date object.  Can be used to conveniently initialize <TT>RANLUX</TT>
* from the clock,
*
* <PRE>
* RandomElement e = Ranlux(new Date());
* </PRE>
*
* @param d date used to generate seed
*
*/

public Ranlux(Date d) {
```

```
init_arrays();
rluxgo(lxdflt,(int) (ClockSeed(d)%Integer.MAX_VALUE) );
};

/**
* Turns diagnostic messages on and off.  If <TT>setDiag(true)</TT> is
* called,  <TT>RANLUX</TT> will print diagnostic information to
* <TT>System.err</TT>
*
* @param b diagnostic message status
*/

public void setDiag(boolean b) {
diagOn=b;
};

/**
*
* The random number generator.
*
* @returns a pseudo-random double in the range (0,1)
*/

public final double raw() {

int i,k,lp;
float uni,out;

uni=seeds[j24]-seeds[i24]-carry;
if (uni < (float) 0.0) {
uni=uni+ (float) 1.0;
carry = twom24;
} else carry = (float) 0.0;

seeds[i24]=uni;

i24=next[i24];
j24=next[j24];

out=uni;

if (uni<twom12)
out += twom24*seeds[j24];

/* zero is forbidden in case user wants logarithms */

if (out==0.0) out = twom24*twom24;

in24++;
```

```
if(in24 == 24) {
in24=0;
kount += nskip;
for(i=1;i<=nskip;i++) {
uni=seeds[j24]-seeds[i24]-carry;
if (uni < (float) 0.0) {
uni=uni+ (float) 1.0;
carry = twom24;
} else carry = (float) 0.0;

seeds[i24]=uni;

i24=next[i24];
j24=next[j24];
}
}

kount++;
if (kount>=igiga) {
mkount++;
kount -= igiga;
};

return out;
};

private void init_arrays() {

/*
*
* converted from fortran:  fortran arrays start at 1,  java arrays start at
* 0.  Here we take the low road to compatibility...  We declare arrays that
* are one bigger than the fortran code.  This wastes three ints and a double;
* If you're porting this to a Commodore 64 you might need the space.
*
*/

iseeds = new int[24+1];
isdext = new int[25+1];
next   = new int[24+1];
seeds  = new float[24+1];
};

private void rluxdef()
{

int lp,i,k;

jseed = jsdflt;
inseed = jseed;
```

```
diag("RANLUX DEFAULT INITIALIZATION: "+jseed);
luxlev = lxdflt;
nskip = ndskip[luxlev];
lp = nskip + 24;
in24 = 0;
kount = 0;
mkount = 0;
diag("RANLUX DEFAULT LUXURY LEVEL =  "+luxlev+"    p = "+lp);
twom24 = (float) 1.0;

for(i = 1;i <= 24;i++)
{
twom24 = twom24 * (float) 0.5;
k = jseed / 53668;
jseed = 40014 * (jseed-k*53668) - k * 12211;
if (jseed<0) jseed = jseed + icons;
iseeds[i] = jseed % itwo24;
};

twom12 = twom24 * (float) 4096.0;

for(i = 1 ; i<=24;i++)
{
seeds[i] = iseeds[i] * twom24;
next[i] = i - 1;
};

next[1] = 24;
i24 = 24;
j24 = 10;
carry = (float) 0.0;
if (seeds[24] == 0.0) carry = twom24;

};

private final void rluxgo(int lux,int ins)
{
int ilx, i, iouter, isk, k, inner, izip, izip2;
float uni;

if (lux<00)
{
luxlev = lxdflt;
}
else if (lux<=maxlev)
{
luxlev = lux;
}
else if (lux<24 || lux>2000)
{
```

```
luxlev = maxlev;
diag("RANLUX ILLEGAL LUXURY RLUXGO: "+lux);
} else
{
luxlev = lux;
for(ilx=0;ilx<=maxlev;ilx++)
if (lux == ndskip[ilx]+24)
luxlev=ilx;

};

if (luxlev <= maxlev) {
nskip = ndskip[luxlev];
diag("RANLUX LUXURY LEVEL SET BY RLUXGO : "+luxlev+" P=
"+(nskip+24));
} else {
nskip = luxlev-24;
diag("RANLUX P-VALUE SET BY RLUXGO TO: "+luxlev);
};

in24=0;

if (ins<0)
diag("Illegal initialization by RLUXGO, negative input seed");

if (ins>0) {
jseed = ins;
diag("RANLUX INITIALIZED BY RLUXGO FROM SEED "+jseed);
} else
{

jseed=jsdflt;
diag("RANLUX INITIALIZED BY RLUXGO FROM DEFAULT SEED");
};

inseed = jseed;
twom24 = (float) 1.0;

for(i=1;i<=24;i++) {
twom24 = twom24 * (float) 0.5;
k = jseed / 53668;
jseed = 40014 * (jseed-k*53668) - k * 12211;
if (jseed < 0) jseed = jseed + icons;
iseeds[i] = jseed % itwo24;
};

twom12=twom24*4096;

for(i=1;i<=24;i++) {
seeds[i] = iseeds[i] * twom24;
```

```
next[i] = i - 1;
};

next[1]=24;
i24=24;
j24=10;
carry = (float) 0.0;

if (seeds[24] == 0.0) carry = twom24;

kount = 0;
mkount = 0;

};

private void diag(String s)
{
if (diagOn)
System.err.println(s);
};

};
```

- **Η κλάση Ranlux (βοηθητική της RNG)**

```
package Reader;
import java.util.*;
import java.io.Serializable;
import jade.core.Agent;
//
// RngPack 1.1a by Paul Houle
// http://www.honeylocust.com/RngPack/
//

/**
*
* <TT>RANMAR</TT> is a lagged Fibonacci generator proposed by Marsaglia
and
* Zaman and is a good research grade generator.  This version of
* <TT>RANMAR</TT> is based on the paper by James,  which is a good
* reference for the properties of <TT>RANMAR</TT> and several other
* generators.
*
```

```
* <BR>
* <B>REFERENCES:</B>
* <BR>
* F. James, <CITE>Comp. Phys. Comm.</CITE> <STRONG>60</STRONG>
(1990) p 329-344
* <BR>
* and was originally described in
* <BR>
* G. Marsaglia, A. Zaman and W.-W Tsang, <CITE>Stat. Prob. Lett</CITE>
<STRONG>9</STRONG> (1990) p 35.
*
*
* <P>
* <A HREF="/RngPack/src/edu/cornell/lassp/houle/RngPack/Ranmar.java">
* Source code </A> is available.
*
* @author <A HREF="http://www.honeylocust.com/"> Paul Houle </A> (E-mail:
<A HREF="mailto:paul@honeylocust.com">paul@honeylocust.com</A>)
* @version 1.1a
*/


public class Ranmar extends RandomSeedable implements Serializable {

double c,cd,cm,u[],uvec[] ;
int i97,j97 ;

/**
* Default seed.  <CODE>DEFSEED=54217137</CODE>
*/
public static int DEFSEED=54217137;

/**
* The 46,009,220nd prime number,
* he largest prime less than 9*10<SUP>8</SUP>.  Used as a modulus
* because this version of <TT>RANMAR</TT> needs a seed between 0
* and 9*10<SUP>8</SUP> and <CODE>BIG_PRIME</CODE> isn't
commensurate
* with any regular period.
* <CODE>BIG_PRIME= 899999963</CODE>
*/
public static int BIG_PRIME=899999963;

/**
*
* Initialize Ranmar with a specified integer seed
*
* @param ijkl seed integer;  <TT>Ranmar(int ijkl)</TT> takes uses
* <TT>ijkl</TT> modulus <TT>BIG_PRIME</TT> as a seed for
<TT>RANMAR.</TT>
```

```
*
*/

public Ranmar(int ijkl)
{
ranmarin(Math.abs(ijkl % BIG_PRIME));
};

/**
*
* Initialize Ranmar with a specified long seed
*
* @param ijkl seed long;  <TT>Ranmar(long ijkl)</TT> takes uses
* <TT>ijkl</TT> modulus <TT>BIG_PRIME</TT> as a seed for
<TT>RANMAR.</TT>
*
*/

public Ranmar(long ijkl)
{
ranmarin((int) Math.abs(ijkl % BIG_PRIME));
};



/**
*
* Initialize Ranmar with a default seed taken from Marsaglia and
* Zaman's paper.  Equivalent to <CODE>Ranmar(54217137).</CODE>
*
*/

public Ranmar()
{
ranmarin(DEFSEED);
};

/**
*
* Seed <TT>RANMAR</TT> from the clock.
*
* <PRE>
* RandomElement e=new Ranmar(new Date());
* </PRE>
*
* @param d a Date object to seed Ranmar with,  typically <CODE>new
Date()</CODE>
*
*/
```

```
public Ranmar(Date d) {

ranmarin((int) ClockSeed(d) % BIG_PRIME);
};

/**
*
* Internal methods:  ranmarin is the initialization code for the
* generator.
*
*/

void ranmarin(int ijkl)
{

int ij,kl;
int i,ii,j,jj,k,l,m ;
double s,t ;

u = new double[97];
uvec = new double[97];


ij=ijkl/30082;
kl=ijkl-30082*ij;

i = ((ij/177) % 177) + 2 ;
j = (ij % 177) + 2 ;
k = ((kl/169) % 178) + 1 ;
l = kl % 169 ;
for (ii=0; ii<97; ii++)
{
s = 0.0 ;
t = 0.5 ;
for (jj=0; jj<24; jj++)
{
m = (((i*j) % 179) * k) % 179 ;
i = j ;
j = k ;
k = m ;
l = (53*l + 1) % 169 ;
if ( ((l*m) % 64) >= 32) s += t ;
t *= 0.5 ;
}
u[ii] = s ;
}
c   =  362436.0 / 16777216.0 ;
cd  = 7654321.0 / 16777216.0 ;
cm  =16777213.0 / 16777216.0 ;
i97 = 96 ;
```

```
j97 = 32 ;

};

/**
* The generator
* @return a pseudo random number
*/

final public double raw() {

double uni;

uni=u[i97]-u[j97];
if (uni<0.0) uni+=1.0;
u[i97]=uni;
if (--i97<0) i97=96;
if (--j97<0) j97=96;
c-=cd;
if (c<0.0) c+=cm;
uni-=c;
if (uni<0.0) uni+=1.0;
return(uni);
};

/**
*
* A version of the generator for filling arrays,  inlined for speed
*
* @param d an array of doubles to be filled
* @param n size of the array
*
*/

final public void raw(double d[],int n) {

double uni;

for(int i=0;i<n;i++)
{
uni=u[i97]-u[j97];
if (uni<0.0) uni+=1.0;
u[i97]=uni;
if (--i97<0) i97=96;
if (--j97<0) j97=96;
c-=cd;
if (c<0.0) c+=cm;
uni-=c;
if (uni<0.0) uni+=1.0;
d[i]=uni;
```

```
};
};
};
```

6. Βιβλιογραφία

[1] IDTechEx, "*RFID Forecasts, Players & Opportunities 2008-2018"*, IDTechEx, 2008, http://www.idtechex.com/research/reports/rfid_forecasts_players_and_opportunities_2008_2018_000193.asp.

[2] OECD, "*Radio-Frequency Identification (RFID): Drivers, Challenges and Public Policy Considerations"*, Paris: Organisation for Economic Co-operation and Development (OECD), 2006, http://www.oecd.org/dataoecd/57/43/36323191.pdf.

[3] OECD, "*Radio-Frequency Identification (RFID): A Focus on Information Security and Privacy"*, Organisation for Economic Co-operation and Development (OECD), 2008, http://www.olis.oecd.org/olis/2007doc.nsf/linkto/dsti-iccp-reg(2007)9-final.

[4] RFID Journal, "Military's RFID Alternative: IPv6", *RFID Journal*, 2003, http://www.rfidjournal.com/article/articleview/609/-1/1

[5] D. Vadhia, R. Gupta, IPv6 vs. EPC: a face off on the RFID battleground, February 2004, http://www.worldinternetcenter.com/Pubs/Pubs2004/feb05/IPv6vEPC.pdf.

[6] S. Sarma, S. Weis, Daniel W. Engels, "RFID systems, security and privacy implications", *Technical Report MIT-AUTOID-WH-014*, AutoID Center, MIT, 2002.

[7] X. Gao, Z. Xiang, H. Wang, J. Shen, J. Huang, S. Song; "An Approach to Security and Privacy of RFID System for Supply Chain," *IEEE International Conference on E-Commerce Technology for Dynamic E-Business, 2004.*, pp.164-168, 15-15 Sept. 2004

[8] A. Juels, D. Molnar, D. Wagner, "Security and Privacy Issues in E-passports", *First International Conference on Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005*, pp. 74-88, 05-09 Sept. 2005.

[9] A. Juels, R. Pappu, "Squealing Euros: Privacy Protection in RFID-Enabled Banknotes," R.N. Wright, ed., Le Gosier, Guadeloupe, French West Indies: *Springer-Verlag*, 2003, pp. 103-121.

[10] A. Juels, "RFID Security and Privacy: A Research Survey", *Journal of Selected Areas in Communication* (J-SAC), 24(2):381-395, February 2006.

[11] H. Knospe, H. Pohl. "RFID security", *Information Security Technical Report*, 9(4):39–50, November–December 2004.

[12] S. Garfinkel, A. Juel, R. Pappu, "RFID privacy: an overview of problems and proposed solutions", *IEEE Security & Privacy Magazine*, Vol. 3 Issue 3, May-June 2005.

[13] P. Peris-Lopez, J. C. Hernandez-Castro, J. Estevez-Tapiador, A. Ribagorda, "RFID Systems: A Survey on Security Threats and Proposed Solutions", *Lecture notes in computer science*, Springer-Verlag, 2006, pp. 159-170.

[14] BSI (Bundesamt für Sicherheit in der Informationstechnik), "Security Aspects and Prospective Applications of RFID Systems", 2005 http://www.bsi.de/english/publications/studies/rfid/RIKCHA_englisch.pdf.

[15] S. Weis, "Security and Privacy in Radio-Frequency Identification Devices", *Master's thesis*, Massachusetts Institute of Technology, Cambridge, MA, USA, May 2003.

[16] Κ.Λαμπρινουδάκης, Λ. Μήτρου, Σ. Κάτσικας, Σ. Γκρίτζαλης, "Προστασία της Ιδιωτικότητας & Τεχνολογίες Πληροφορικής και Επικοινωνιών", 2010.

[17] F. Bellifemine, G. Caire, D. Greenwood, "Developing Multi-Agent systems with JADE", 2007.

[18] http://towardsnothing.blogspot.com/2009/02/setting-up-jade-in-netbeans.html

[19] Flying 2.0 - enabling automated air travel by identifying and addressing the challenges of IoT & RFID technology. Technical Report, European Network and information Security Agency (ENISA) Apr 2010.

[20] Rekleitis E, Rizomiliotis P, Gritzalis S. An holistic approach to RFID security and privacy. Tokyo, Japan, 2010.