ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη συστήματος ηλεκτρονικής ψηφοφορίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μαρία Ν. Κουκοβίνη
Ευγενία Ι. Παπαγιαννακοπούλου

Επιβλέπων : Ιάκωβος Στ. Βενιέρης
Καθηγητής ΕΜΠ

Αθήνα, Δεκέμβριος 2007

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη συστήματος ηλεκτρονικής ψηφοφορίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Μαρία Ν. Κουκοβίνη
Ευγενία Ι. Παπαγιαννακοπούλου

Επιβλέπων : Ιάκωβος Στ. Βενιέρης
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Δεκεμβρίου 2007.

Αθήνα, Δεκέμβριος 2007

.....	.....	.....
Ιάκωβος Στ. Βενιέρης	Δήμητρα-Θεοδώρα Ι. Κακλαμάνη	Γ. Ι. Στασινόπουλος
Καθηγητής	Αναπλ.Καθηγήτρια	Καθηγητής

.....
Μαρία Ν. Κουκοβίνη

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευγενία Ι. Παπαγιαννακοπούλου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός αυτής της εργασίας ήταν η υλοποίηση ενός συστήματος ηλεκτρονικής ψηφοφορίας, βασισμένου σε μια πλατφόρμα κινητών πρακτόρων. Οι κινητοί πράκτορες αποτελούν αιχμή της σημερινής τεχνολογίας δικτύων, ενώ στο θέμα της ηλεκτρονικής ψηφοφορίας έχουν γίνει κατά καιρούς και σε παγκόσμιο επίπεδο διάφορες προσπάθειες.

Η παρούσα μορφή της εργασίας χρησιμοποιεί τις βασικές λειτουργίες μιας ήδη υπάρχουσας πλατφόρμας κινητών πρακτόρων, προσδίδοντάς τους επιπλέον τα απαραίτητα χαρακτηριστικά ασφάλειας, ώστε να καλύπτονται οι απαιτήσεις ενός τυπικού συστήματος ψηφοφορίας. Πρόκειται για ένα αποκεντρωμένο σύστημα, στο οποίο ένας κεντρικός εξυπηρετητής αναλαμβάνει τη συλλογή και καταμέτρηση των ψήφων, καθώς επίσης και συντονιστικές λειτουργίες, ενώ πολλοί περιφερειακοί εξυπηρετητές αντιστοιχούν στα κατά τόπους εκλογικά κέντρα. Οι τελευταίοι διαθέτουν την απαραίτητη γραφική διεπαφή ψηφοφόρου και δρομολογούν την αποστολή της ψήφου στον κεντρικό εξυπηρετητή. Στα πλαίσια αυτά, χρησιμοποιήθηκαν μηχανισμοί κρυπτογραφίας για την αυθεντικοποίηση, πιστοποίηση και προστασία των δεδομένων. Συγκεκριμένα, κάθε πράκτορας καλείται να μεταφέρει την ψήφο, αφού πρώτα την κρυπτογραφήσει και την υπογράψει ψηφιακά. Οι απαιτήσεις σε ασφάλεια καλύπτονται επιπρόσθετα με τη συνεχή ενημέρωση όλων των συνιστωσών μέσω ενός ευέλικτου συστήματος διαχείρισης βάσεων δεδομένων.

Λέξεις Κλειδιά

Ηλεκτρονική ψηφοφορία, κινητοί πράκτορες, ασφάλεια, Java, JSP, βάσεις δεδομένων, κρυπτογραφία, ψηφιακή υπογραφή, ψηφιακό πιστοποιητικό.

Abstract

The objective of this thesis was the implementation of an electronic voting system, based on a mobile agent platform. Mobile agents are on the top of today's network technology, while numerous efforts have been made globally regarding electronic voting.

The current form of the project uses the basic functionalities of an already existing mobile agent platform, while adding the necessary security features, so that the requirements of a typical voting system are fulfilled. A decentralized system is implemented, in which a central server is responsible for the collection and counting of votes, as well as administrative functions, while various peripheral servers correspond to the polling places, providing the necessary graphical voter interface and also sending the vote to the central server. In order to achieve the above, cryptographic technics for authentication, verification and protection of data have been used. Specifically, each agent is called in order to transport the vote, after encrypting and digitally signing it. Security features are further enhanced with the constant update of all participating components by means of a flexible database management system.

Key Words

Electronic voting, mobile agents, security, Java, JSP, databases, cryptography, digital signature, digital certificate.

Ευχαριστούμε θερμά τον επιβλέποντα καθηγητή μας, κ. Ιάκωβο Βενιέρη, για την ευκαιρία που μας έδωσε να ασχοληθούμε με ένα τόσο επίκαιρο και ενδιαφέρον θέμα, και βέβαια τους υποψήφιους διδάκτορες Γεώργιο Λιουδάκη και Ιωάννη Φουκαράκη για την πολύτιμη βοήθειά τους καθ'όλη τη διάρκεια εκπόνησης της παρούσας εργασίας.

Περιεχόμενα

1 Εισαγωγή

1.1 Εκλογές

Οι εκλογές ως «ανοιχτή» και δίκαιη διαδικασία αποτελούσαν και αποτελούν βασική προϋπόθεση σε κάθε δημοκρατία, ως εκ τούτου και σε κάθε σύγχρονη δημοκρατία. Για τους πολίτες μιας χώρας οι εκλογές είναι σημαντικές μόνο εξαιτίας και στη βάση του γεγονότος ότι με αυτό τον τρόπο οι ίδιοι καθορίζουν ουσιαστικά και κατά πλειοψηφία ποιοι θα τους εκπροσωπούν. Απώλεια εμπιστοσύνης στο εκλογικό σύστημα συνεπάγεται αναπόφευκτα και απώλεια εμπιστοσύνης στο κυβερνητικό. Σε μιαν αντιπροσωπευτική δημοκρατία, όπως είναι η ελληνική αλλά και όλες πλέον σήμερα οι δημοκρατίες στον πλανήτη, το ανώτατο όργανο του κράτους, που είναι ο λαός, ασκεί την εξουσία του με τις εκλογές. Με αυτές, το εκλογικό σώμα επιλέγει τους πράκτορές του οι οποίοι, για όλη τη διάρκεια της βουλευτικής περιόδου, θα ασκήσουν όλες εκείνες τις αρμοδιότητες που προβλέπουν το Σύνταγμα και οι νόμοι.

Το εκλογικό σώμα

Το εκλογικό σώμα είναι το σύνολο των ελλήνων πολιτών που έχουν το δικαίωμα της ψήφου (ή το "δικαίωμα του εκλέγειν" ή το "ενεργητικό εκλογικό δικαίωμα").Το δικαίωμα αυτό διαθέτουν όσοι έχουν την ελληνική ιθαγένεια, έχουν συμπληρώσει το δέκατο όγδοο (18ο) έτος της ηλικίας τους (ή πρόκειται να το συμπληρώσουν εντός του έτους στο οποίο διεξάγονται οι εκλογές), έχουν ικανότητα για δικαιοπραξία και δεν έχουν υποστεί αμετάκλητη ποινική καταδίκη για ορισμένα εγκλήματα. Για να ασκηθεί το εκλογικό δικαίωμα πρέπει ο εκλογέας να έχει εγγραφεί στους εκλογικούς καταλόγους.

Εκλογική διαδικασία

Τα μέλη του εκλογικού σώματος προσέρχονται στα εκλογικά τμήματα στα οποία είναι εγγεγραμμένοι την καθορισμένη μέρα των εκλογών, ενώ η ψηφοφορία διαρκεί από την ανατολή μέχρι τη δύση του ηλίου. Υπεύθυνη για την ομαλή διεξαγωγή της διαδικασίας σε κάθε εκλογικό κέντρο είναι η εφορευτική επιτροπή. Ο ψηφοφόρος προσκομίζει την αστυνομική του ταυτότητα ή το διαβατήριο και παραλαμβάνει το σύνολο των ψηφοδελτίων με ένα φάκελο. Χωρίς να είναι ορατός από τους άλλους, συνήθως πίσω από ένα παραβάν, τοποθετεί στο φάκελο το ψηφοδέλτιο της επιλογής του, στο οποίο, εφόσον το επιθυμεί, μπορεί να σημειώσει με σταυρό μέχρι ένα συγκεκριμένο αριθμό υποψηφίων. Στη συνέχεια ρίχνει το φάκελο στην κάλπη και το όνομά του διαγράφεται από τη λίστα.

Αφού κλείσουν οι κάλπες, ξεκινά η καταμέτρηση των ψήφων στα εκλογικά κέντρα. Σε αυτήν περιλαμβάνονται μόνο τα θεωρούμενα ως έγκυρα ψηφοδέλτια, σύμφωνα με τους αντίστοιχους κανονισμούς. Με αυτόν τον τρόπο, προκύπτουν τα τελικά συγκεντρωτικά αποτελέσματα, τα οποία οδηγούν στην κατανομή των εδρών στη βουλή με βάση το ισχύον εκλογικό σύστημα.

Αρχές που διέπουν την ψηφοφορία

1) Η αρχή της καθολικής ψηφοφορίας. Σύμφωνα με την αρχή αυτή, από το εκλογικό σώμα αποκλείονται μόνον οι πολίτες εκείνοι που δεν συγκεντρώνουν τις ελάχιστες προϋποθέσεις που θέτει, με αποκλειστικό τρόπο, το Σύνταγμα. Ο κοινός νομοθέτης δεν μπορεί να προβλέψει επί πλέον λόγους στέρησης του δικαιώματος της ψήφου.

2) Η αρχή της ισότητας της ψήφου. Η αρχή αυτή εξειδικεύεται σε δύο επί μέρους αρχές: ότι κάθε πολίτης διαθέτει μόνον μία ψήφο και ότι όλες οι ψήφοι είναι νομικά ισοδύναμες.

3) Η αρχή της άμεσης ψηφοφορίας. Κατά την αρχή αυτή, δεν μεσολαβεί κάποια άλλη βούληση μεταξύ του εκλογέα και του αποτελέσματος της εκλογής. Με άλλα λόγια, δεν είναι δυνατόν οι εκλογείς να εκλέξουν κάποιους "εκλέκτορες" οι οποίοι με τη σειρά τους θα εκλέξουν τους βουλευτές.

4) Η αρχή της μυστικότητας της ψήφου. Με την αρχή αυτή εξασφαλίζεται το ότι η εκλογική βούληση του εκλογέα δεν θα γίνει γνωστή σε τρίτους.

5) Η αρχή της υποχρεωτικής ψηφοφορίας. Σύμφωνα με την αρχή αυτή, η άσκηση του εκλογικού δικαιώματος είναι υποχρεωτική. Πρέπει πάντως να σημειωθεί ότι με τη συνταγματική αναθεώρηση του 2001 καταργήθηκε η πρόβλεψη για νόμο που θα επιβάλει ποινικές κυρώσεις στον εκλογέα που δεν θα λάβει μέρος στις εκλογές.

6) Η αρχή της ταυτόχρονης διενέργειας των εκλογών σε ολόκληρη την Επικράτεια. Με το αναθεωρημένο άρθρο 51 παρ. 4 του Συντάγματος, η αρχή αυτή μπορεί να καμφθεί για εκλογείς που βρίσκονται στο εξωτερικό, αρκεί να τηρείται η αρχή της ταυτόχρονης καταμέτρησης και ανακοίνωσης των αποτελεσμάτων.

7) Η αρχή της αυτοπρόσωπης άσκησης του εκλογικού δικαιώματος ισχύει, πλέον, μόνον για τους εκλογείς που βρίσκονται στην Επικράτεια. Το αναθεωρημένο άρθρο 51 παρ. 4 του Συντάγματος προβλέπει τη δυνατότητα στους εκλογείς που βρίσκονται στο εξωτερικό να ασκούν το εκλογικό τους δικαίωμα "με επιστολική ψήφο ή άλλο πρόσφορο μέσο".

1.2 Σκοπός της παρούσας διπλωματικής εργασίας

Σκοπός αυτής της εργασίας είναι η υλοποίηση ενός συστήματος ηλεκτρονικής ψηφοφορίας, το οποίο θα ικανοποιεί όσο το δυνατόν περισσότερες από τις απαιτήσεις των εκλογικών συστημάτων που εφαρμόζονται σήμερα, όπως:

- Ατομική εγγραφή και εξουσιοδότηση.
- Προκαθορισμένη διάρκεια της διαδικασίας ψηφοφορίας.
- Διενέργεια των εκλογών στα ειδικά διαμορφωμένα εκλογικά κέντρα.
- Διανομή ψηφοδελτίων με προκαθορισμένους υποψήφιους.
- Επιλογή της εκάστοτε προτίμησης (συνήθως κρυφά).
- Ασφαλής συγκέντρωση ψηφοδελτίων για αμερόληπτη καταμέτρηση.

Η συγκεκριμένη εργασία χρησιμοποιεί μια υλοποιημένη σε Java πλατφόρμα κινητών πρακτόρων, που έγινε στα πλαίσια προηγούμενης διπλωματικής εργασίας. Η πλατφόρμα αυτή προσφέρει τις βασικές υπηρεσίες που χρειάζονται για την ανάπτυξη εφαρμογών με χρήση κινητών πρακτόρων, όπως δυναμικό φόρτωμα πρακτόρων, υπηρεσίες κινητικότητας, επικοινωνίας και αναζήτησης. Για την επικοινωνία μεταξύ των οντοτήτων της πλατφόρμας και μεταξύ των πρακτόρων η υπάρχουσα υποδομή έκανε χρήση των Υπηρεσιών Διαδικτύου και ιδιαίτερα του πρωτοκόλλου SOAP (Simple Object Access Protocol). Στα πλαίσια της παρούσας εργασίας γίνεται, μεταξύ άλλων, χρήση μηχανισμών ασφαλείας που επιτρέπουν τη μεταφορά μηνυμάτων μεταξύ δύο οντοτήτων που χρησιμοποιούν την πλατφόρμα αυτή, προκειμένου να εξασφαλίζονται χαρακτηριστικά όπως η ασφάλεια και το απόρρητο της ψήφου.

Στα ακόλουθα κεφάλαια θα γίνει μια ανασκόπηση των μέχρι σήμερα προσπαθειών εφαρμογής συστημάτων ηλεκτρονικής ψηφοφορίας ανά τον κόσμο. Επίσης, θα γίνει ξεχωριστή αναφορά στην τεχνολογία των κινητών πρακτόρων καθώς και παρουσίαση των τελευταίων εξελίξεων πάνω σε αυτήν την τεχνολογία, ιδίως σε ό,τι αφορά στην ασφάλεια. Στη συνέχεια θα παρουσιαστούν βασικές έννοιες της κρυπτογραφίας, απαραίτητης σε ανάλογες εφαρμογές που έχουν υψηλές απαιτήσεις σε ασφάλεια, καθώς επίσης και οι τεχνολογίες που χρησιμοποιήθηκαν συγκεκριμένα για τους σκοπούς της παρούσας διπλωματικής. Τέλος, γίνεται εκτενής περιγραφή του σχεδιασμού και της λειτουργίας του συστήματος.

2 Ηλεκτρονική ψηφοφορία

2.1 Εισαγωγή

Στη σημερινή εποχή των υπολογιστών και της τεχνολογίας της πληροφορίας εμφανίζεται πλέον ως αναγκαιότητα η λεγόμενη ηλεκτρονική δημοκρατία (e-democracy) με βασικό τμήμα της την ηλεκτρονική ψηφοφορία (e-voting), η οποία αναφέρεται στη χρήση υπολογιστών ή εν γένει υπολογιστικού εξοπλισμού στη διαδικασία της ψηφοφορίας. Ως σύστημα ηλεκτρονικής ψηφοφορίας (electronic voting system) ορίζεται το σύστημα ψηφοφορίας, στο οποίο τα εκλογικά δεδομένα καταγράφονται, αποθηκεύονται και υπόκεινται σε επεξεργασία πρωτίστως ως ψηφιακή πληροφορία.

Μια τέτοια διαδικασία ωστόσο υπόκειται εγγενώς σε λάθη και, όπως έχει αποδειχθεί ιστορικά, σε διαστρέβλωση και απάτη. Απαιτεί συνεπώς εξαιρετική ακεραιότητα (ιδιαίτερα όταν μιλάμε για χρήση υπολογιστικών συστημάτων), όπως επίσης εντιμότητα και εμπειρία των εμπλεκομένων στην οργάνωση και διεξαγωγή των εκλογών ατόμων. Μπορεί επίσης να απαιτεί και ένα minimum δεξιοτήτων από την πλευρά των ψηφοφόρων. Υπό αυτό το πρίσμα λοιπόν, η ψηφοφορία αποτελεί αντιπροσωπευτικό παράδειγμα ενός από άκρο σε άκρο (end-to-end) προβλήματος ασφάλειας: από την καταχώρηση και πιστοποίηση των ψηφοφόρων έως την καθεαυτό ψηφοφορία και καταμέτρηση των αποτελεσμάτων, ενώ περικλείει ένα ευρύ φάσμα τεχνολογικών και κοινωνικών προβλημάτων, που πρέπει να ληφθούν υπόψη. Λόγω της ραγδαίας ανάπτυξης των τεχνολογιών υπολογιστών και των εξελίξεων στην κρυπτογραφία παρουσιάζεται η ηλεκτρονική ψηφοφορία ως εφαρμόσιμη εναλλακτική από τη μια, από την άλλη όμως οι υψηλές απαιτήσεις ασφαλείας εξακολουθούν να την καθιστούν ένα από τα μεγαλύτερα προβλήματα προς επίλυση, κάτι που απαιτεί τη συνεργασία ειδικών σε διάφορους τομείς, όπως λογισμικό, κρυπτογραφία, πολιτική, νομικές, οικονομικές και πολιτικές επιστήμες [22][24].

2.2 Ιστορική αναδρομή - Προηγούμενες προσπάθειες

Από νωρίς άρχισαν να αναζητούνται τρόποι ψηφοφορίας, που δε βασίζονταν στα παραδοσιακά ψηφοδέλτια (paper-ballots), αλλά συνέχιζαν να βασίζονται στο χαρτί (paper-based). Στις αρχές της δεκαετίας του ’60 άρχισε να υιοθετείται η χρήση διάτρητων καρτών (punch cards), η οποία διαδόθηκε αρκετά μέχρι τα τέλη της ίδιας δεκαετίας. Χρησιμοποιήθηκαν σε δύο μορφές: Votomatic και Datavote, εγκαταλείφθηκαν όμως ύστερα από προβλήματα που παρατηρήθηκαν στις προεδρικές εκλογές της Florida USA το 2000. Παράλληλα χρησιμοποιήθηκε από το 1962 και μέχρι το 2000 περίπου αντί για διάτρητες κάρτες και ένα είδος «οπτικών» ψηφοδελτίων (optical mark-sense ballots) τα οποία στη συνέχεια «σαρώνονταν». Πρέπει, τέλος, να αναφερθεί ότι σε όλο τον 20ό αιώνα, ιδίως μέχρι το 1960, ιδιαίτερα διαδεδομένος ήταν και ο μηχανικός τρόπος ψηφοφορίας (paperless voting), με τη βοήθεια μιας μηχανολογικής εγκατάστασης με μοχλό (mechanical-lever machines). Και αυτός όμως αποδείχτηκε μάλλον ανακριβής, ενώ είχε και το επιπρόσθετο μειονέκτημα ότι δεν διατηρούσε κανενός είδους φυσικό αρχείο ψήφων.

Η χρήση τεχνολογιών ηλεκτρονικής ψηφοφορίας επαναοργανώνει την παραδοσιακή εκλογική διαδικασία. Το χάρτινο ψηφοδέλτιο αντικαθίσταται από το ηλεκτρονικό-ψηφιακό. Τα τοπικά συστήματα ηλεκτρονικής ψηφοφορίας (direct recording electronic (DRE) voting systems) αποτελούν το ηλεκτρονικό ισοδύναμο των mechanical-lever machines και είναι τα μόνα ηλεκτρονικά εκλογικά συστήματα που περιλαμβάνουν εκλογικά κέντρα. Σε όλα τα υπόλοιπα (internet-based), όλες οι διαδικασίες λαμβάνουν χώρα από απόσταση.

Η πρώτη DRE μηχανή ψηφοφορίας (voting machine) χρησιμοποιήθηκε σε πραγματικές εκλογές το 1975 στο Streamwood και Woodstock Illinois. Ήδη όμως από το 1964 είχαν χρησιμοποιηθεί υπολογιστές σε δύο επαρχίες της Georgia στις ΗΠΑ [10][23].

Το 1990 εκδόθηκαν τα πρώτα προαιρετικά VSS (Voting System Standards) από το Federal Election Commission (FEC) Office of Elections Administration (OEA), που πλέον είναι ενσωματωμένη στην Election Assistance Commission (EAC), σύμφωνα με τα οποία στο εξής θα πιστοποιούνταν από τη National Association of State Election Directors (NASED) στα διάφορα από την ίδια εξουσιοδοτημένα ITAs (Independent Testing Authorities) τα εκάστοτε συστήματα ψηφοφορίας, τόσο ως προς το hardware όσο και ως προς το software. Τα συγκεκριμένα standards αναθεωρήθηκαν και έδωσαν τα 2002 VSS, που ισχύουν μέχρι σήμερα. Το Δεκέμβρη του 2005 η EAC ομόφωνα υιοθέτησε τα 2005 VSS, που αυξάνουν σημαντικά τις απαιτήσεις ασφάλειας και διευρύνουν την προσβασιμότητα, προσφέροντας περισσότερες δυνατότητες σε άτομα με ποικίλες «ανικανότητες». Θα τεθούν σε ισχύ το Δεκέμβρη του 2007, αντικαθιστώντας τα VSS 2002 [1].

Παράλληλα ξεκίνησε το 2001 και από την IEEE το project P1583, που αποτελεί το πρώτο standard για εξοπλισμό ψηφοφορίας που αναπτύχθηκε σε ανοιχτή συναινετική διαδικασία σύμφωνα με τους κανόνες ANSI (American National Standards Institute). Από τη στιγμή ωστόσο που δημιουργήθηκε η EAC από τη HAVA (Help America Vote Act) θα είναι αυτή υπεύθυνη για την έκδοση standards μέσω του National Institute of Standards and Technology (NIST) [1].

Η Βραζιλία το 2000 ήταν η πρώτη χώρα που χρησιμοποίησε σύστημα ψηφοφορίας πλήρως βασισμένο σε υπολογιστές. Η Georgia ήταν επίσης η πρώτη πολιτεία στις ΗΠΑ που υιοθέτησε εξολοκλήρου μια ενιαία DRE τεχνολογία για ψηφοφορία το 2002. Σε σύγκριση με το 2000, οπότε χρησιμοποιήθηκαν διάφοροι τύποι ψηφοφορίας, το ποσοστό των undervotes (των ψηφοδελτίων όπου δεν είχαν συμπληρωθεί όλα τα απαιτούμενα στοιχεία εκλογής) μειώθηκε από 4,4% σε λιγότερο από 1%, κάτι που αποτελεί ποιοτικό δείκτη ακεραιότητας ενός εκλογικού συστήματος [24][3].

Ένα πείραμα που αξίζει να αναφερθεί σ' αυτά τα πλαίσια είναι το πιλοτικό Internet-based σύστημα ψηφοφορίας SERVE (Secure Electronic Registration and Voting Experiment), που αναπτύχθηκε για το FVAP (Federal Voting Assistance Program) του Υπουργείου Άμυνας των ΗΠΑ. Σχεδιάστηκε με σκοπό να εφαρμοστεί στις εκλογές του 2004,ώστε να επιτρέψει στους εν δυνάμει ψηφοφόρους αρχικά να εγγραφούν στους εκλογικούς καταλόγους της περιοχής τους και στη συνέχεια να ψηφίσουν ηλεκτρονικά μέσω Internet απ' οπουδήποτε στον κόσμο. Στο πείραμα θα μπορούσαν να συμμετάσχουν εγγεγραμμένοι ψηφοφόροι που βρίσκονταν εκτός συνόρων, οι στρατιωτικοί και οι ψηφοφόροι 50 επαρχιών σε 7 πολιτείες. Η εφαρμογή αυτού του προγράμματος θα χρησίμευε σαν πρότυπο για γενικευμένη χρήση ηλεκτρονικής ψηφοφορίας στο μέλλον. Οι ερευνητές, βέβαια, προειδοποίησαν για ενδεχόμενα κενά του συστήματος και τελικά δεν εφαρμόστηκε με την αρχική του μορφή στις εκλογές του 2004. Αλλά και η εφαρμογή του συστήματος ηλεκτρονικής ψηφοφορίας του 2000 δεν έφερε καλύτερα αποτελέσματα. Αξιοσημείωτο είναι το γεγονός ότι στη Florida εμφανίστηκαν ακόμα μεγαλύτερα προβλήματα από τις εκλογές του 2000. Για παράδειγμα σε κάποιες περιπτώσεις το σύστημα κατέρρευσε και είτε χάθηκαν χιλιάδες ψήφων, είτε χρειάστηκε προσφυγή στα κλασικά ψηφοδέλτια (paper ballots) [7][25][27].

Μια άλλη αντίστοιχη προσπάθεια έγινε και στο Ηνωμένο Βασίλειο κατά τα έτη 2002, 2003, 2004 και 2006 στα πλαίσια ενός πιλοτικού προγράμματος με απώτερο σκοπό τη διεξαγωγή ηλεκτρονικών εκλογών μετά το 2006, κάτι που προέκυψε από την ανάγκη για εκσυγχρονισμό της διαδικασίας και μεγαλύτερη προσέλευση. Η παραδοσιακή εκλογική διαδικασία είτε συμπληρώθηκε είτε αντικαταστάθηκαν τμήματά της από ηλεκτρονικές διεργασίες. Οι τεχνολογίες που χρησιμοποιήθηκαν ήταν: Internet voting, telephone voting, Direct Recording e-voting machines (DRE) / Kiosk voting, SMS text message voting, Interactive Digital Television voting. Η προσπάθεια συντονίστηκε από κοινού από την κεντρική διοίκηση (Office of the Deputy Prime Minister) και τις τοπικές αρχές (Local Public Authorities (PAs)). Οι ανεπάρκειες του συστήματος, που θα αποτελέσουν σημαντική βοήθεια για το μέλλον, εντοπίστηκαν κυρίως σε θέματα οργανωτικά, εσωτερικής επικοινωνίας και συνεργασίας των συντελεστών-συνιστωσών της διαδικασίας, μικρής ευελιξίας και

δυνατότητας ελέγχου, εξάρτησης από τους εμπορικούς προμηθευτές. Φάνηκε επίσης ότι αν δεν ήταν διαθέσιμα και χάρτινα ψηφοδέλτια ως back-up διαδικασία, θα υπήρχε πρόβλημα. Πιθανώς αυτό αποτελεί απαραίτητο μέτρο έως ότου εξασφαλιστεί η τελειοποίηση της ηλεκτρονικής ψηφοφορίας [13].[16].

Ηλεκτρονική ψηφοφορία εφαρμόστηκε επίσης στις δημοτικές εκλογές της Εσθονίας τον Οκτώβρη του 2005,όταν για πρώτη φορά το εκλογικό σώμα μιας ολόκληρης χώρας είχε τη δυνατότητα να ψηφίσει μέσω Internet. Εξαιτίας του γεγονότος ότι για πρώτη φορά εφαρμόστηκε κάτι τέτοιο και οι «ηλεκτρονικοί» ψηφοφόροι ήταν σχετικά λίγοι, δεν μπορούμε να εξάγουμε απολύτως ασφαλή συμπεράσματα. Ωστόσο η διαδικασία κρίθηκε σε γενικές γραμμές επιτυχής, γι αυτό και χρησιμοποιήθηκε και στις βουλευτικές εκλογές του 2007 στην ίδια χώρα [18].

Στη Γαλλία χρησιμοποιήθηκε για πρώτη φορά το σύστημα ηλεκτρονικής ψηφοφορίας το 2003 από γάλλους πολίτες που ζούσαν στις ΗΠΑ για τις εκλογές της Ένωσης Γάλλων Πολιτών Εξωτερικού, με συμμετοχή πάνω από 60%. Από την άλλη, η πιλοτική χρήση e-voting τεχνολογίας στις προεδρικές εκλογές του 2007 σε 82 τοποθεσίες προκάλεσε ποικίλα προβλήματα, όπως μεγάλες ουρές, κατάρρευση συστημάτων και δυσκολία εξοικείωσης των ψηφοφόρων [27].

Κατά καιρούς επιχειρήθηκε εφαρμογή τέτοιων συστημάτων, σε πιο περιορισμένη κλίμακα βέβαια, και σε άλλες χώρες όπως Αυστραλία, Βέλγιο, Βραζιλία, Καναδά, Γερμανία., Ιρλανδία, Ιταλία, Ολλανδία, Ινδία, Νορβηγία, Ρουμανία κι Ελβετία [26].[27].

Αξίζει σε αυτό το σημείο να αναφέρουμε, ότι οι ειδικοί προβλέπουν ότι τα συστήματα ηλεκτρονικής ψηφοφορίας μέσα στα επόμενα χρόνια θα εδραιωθούν σε μικρότερης δυναμικής εκλογές, όπως οι δημοτικές και οι σχολικές. Εκλογές τέτοιου βεληνεκούς παρουσιάζουν συνήθως τα μεγαλύτερα ποσοστά αποχής και είναι σχετικά «ελαστικές» από πλευράς ασφάλειας (συνήθως οι ίδιοι οι υποψήφιοι καταμετρούν τελικά τις ψήφους), ενώ επίσης στις περισσότερες περιπτώσεις γίνεται κατασπατάληση ανθρωπίνων και χρηματικών πόρων. Έτσι, τα συστήματα ηλεκτρονικής ψηφοφορίας εισάγονται σταδιακά, με στόχο τη μείωση της αποχής και του κόστους, με παράλληλη αύξηση της ασφάλειας.

2.3 Πλεονεκτήματα και στόχοι συστημάτων ηλεκρονικής ψηφοφορίας

Τα εκλογικά συστήματα Internet και DRE έχουν κάποια έμφυτα πλεονεκτήματα σε σχέση με τα παραδοσιακά συστήματα ψηφοφορίας, συμπεριλαμβανομένης της μείωσης του λάθους του ψηφοφόρου. Αν η διεπαφή χρήστη (interface) του συστήματος είναι κατάλληλα σχεδιασμένη, για παράδειγμα, ειδοποιεί για ενδεχόμενη παράλειψη ψήφου ή και για μη έγκυρη ψηφοφορία. Από την άλλη καθίσταται δυνατή η εξυπηρέτηση ψηφοφόρων με ποικίλες αδυναμίες ή ειδικές ανάγκες, χωρίς να είναι απαραίτητη η ανθρώπινη βοήθεια και κατ' επέκταση μειώνεται η πιθανότητα ανθρώπινης παρέμβασης. Άλλοι παράγοντες που καθιστούν αναγκαία τα συστήματα ηλεκτρονικής ψηφοφορίας και πιθανά οφέλη που πρέπει να ληφθούν υπόψη είναι τα εξής:

-η αυξημένη προσέλευση-συμμετοχή ψηφοφόρων και μείωση της αποχής και η προώθηση με αυτό τον τρόπο της δημοκρατικής διαδικασίας.

-μια νέα δυναμική και νέες δυνατότητες για τις εκλογές (παρέχοντας ψηφοδέλτια σε διάφορες γλώσσες, υποστηρίζοντας μακροσκελή ψηφοδέλτια κλπ).

-το άνοιγμα μιας νέας αγοράς, με συνακόλουθες συνέπειες για το εμπόριο και την απασχόληση.

-η σαφής διευκόλυνση του ψηφοφόρου.

-ο περιορισμός της επέμβασης του ανθρώπινου παράγοντα και συνεπώς της δυνατότητας για νοθεία

-η μεγαλύτερη ταχύτητα στην έκδοση και επεξεργασία των αποτελεσμάτων.

Οι παραπάνω λόγοι έχουν οδηγήσει αρκετές χώρες στο να εκδηλώσουν ενδιαφέρον για τη χρήση τέτοιων συστημάτων σε μικρότερη ή μεγαλύτερη κλίμακα αλλά και την αισιοδοξία ότι η ηλεκτρονική ψηφοφορία σταδιακά θα κυριαρχήσει [11].[15].

2.4 Προβλήματα

Θα πρέπει εδώ να αναφέρουμε ότι υπάρχουν δύο βασικά είδη Internet voting: Polling place (εκλογικό κέντρο) Internet voting και remote (απομακρυσμένο) Internet voting, με το δεύτερο να είναι πολύ πιο ευάλωτο σε εξαπάτηση του ψηφοφόρου, να θέτει σε κίνδυνο το ιδιωτικό απόρρητο και το δικαίωμα της μυστικής ψηφοφορίας, οδηγώντας πιθανώς σε πολιτικό εξαναγκασμό, για παράδειγμα στους χώρους εργασίας ή στο στενό οικογενειακό περιβάλλον. Στο remote electronic voting, οι ψηφοφόροι ψηφίζουν μέσω internet, κυρίως μέσω ενός web browser, από το σπίτι ή από οποιαδήποτε άλλη τοποθεσία, στην οποία υπάρχει πρόσβαση στο internet.

Παρόλη τη διάχυτη αισιοδοξία και τις προοπτικές που ανοίγονται, έχει αποδειχθεί τόσο θεωρητικά όσο και στην πράξη, ότι τα συστήματα ηλεκτρονικής ψηφοφορίας που έχουν αναπτυχθεί μέχρι τώρα εμφανίζουν σημαντικά κενά ασφάλειας και κάνουν την εκλογική απάτη πρωτοφανώς απλή.

- Μια εντυπωσιακή στα αποτελέσματά της έρευνα έδειξε ότι σε μεγάλης κλίμακας εκλογές ακόμα και η αλλαγή μιας μόνο ψήφου ανά μηχάνημα είναι ικανή να ανατρέψει το αποτέλεσμα, ενώ μεγαλύτερες αλλαγές επιτυγχάνουν και τα επιθυμητά περιθώρια. Γενικά, μια απειροελάχιστη αλλαγή στο λογισμικό του μηχανήματος αρκεί. Τα πράγματα διευκολύνονται ακόμα περισσότερο, αν πολλές μηχανές τρέχουν το ίδιο λογισμικό. Επιπλέον αξίζει να σημειωθεί ότι τόσο μικρές αλλαγές μπορούν πιθανότατα να μην εντοπιστούν καθόλου, να ερμηνευθούν ως τυχαίος θόρυβος ή να καλυφθούν από λοιπούς θορύβους του συστήματος, ή ακόμα να υποτιμηθεί η σημασία τους μακροσκοπικά και να μην καταμετρηθούν ξανά οι αντίστοιχοι ψήφοι (recount) [4].
- Ένα επιπλέον σημείο είναι ότι, παρόλο που μετριάζεται η επίδραση του ανθρώπινου παράγοντα, απαιτείται προστασία πλέον όχι μόνο από λάθη και την κακόβουλη δράση ψηφοφόρων και υπευθύνων της εκλογικής διαδικασίας αλλά και από εσκεμμένα ή μη ολισθήματα των αρμόδιων προγραμματιστών, τεχνικών και system administrators [5].
- Επιπρόσθετα αναφέρουμε ότι συστήματα ψηφοφορίας όπως το ηλεκτρονικό, που δεν παράγουν κάποιου είδους φυσικό αρχείο, δημιουργούν θέμα εμπιστοσύνης στο σύστημα, εφόσον χάνεται η δυνατότητα διαφανούς επαλήθευσης. Όλα λοιπόν εξαρτώνται από το αν οι μηχανές λειτουργούν σωστά, οπότε η εμπιστοσύνη ή μη μεταπίπτει στον κατασκευαστή και στα άτομα που τις συντηρούν και τις χειρίζονται. Από την άλλη, η καταγραφή και φύλαξη των ψήφων σε κάποιου είδους φυσικό αρχείο, που είναι απαραίτητες αφενός για την πιστοποίηση και άρα για τον εντοπισμό λάθους ή απάτης, συγκρούονται αφετέρου με τη απαιτούμενη μυστικότητα της ψήφου. Για το τελευταίο βέβαια έχουν γίνει διάφορες προτάσεις, όπως θα δούμε παρακάτω.
- Περαιτέρω προβλήματα εμφανίζονται και στο νομικό πεδίο. Οι κανόνες και οι νόμοι των εκλογικών διαδικασιών βασίζονται παραδοσιακά στην υπόθεση των «υλικών» ψηφοδελτίων, οπότε απαιτείται η αμοιβαία προσαρμογή του κώδικα του νόμου και του προγραμματιστικού κώδικα, ώστε στη συγκεκριμένη περίπτωση η χρήση της τεχνολογίας να διασφαλίζει ουσιαστικά τη δημοκρατία [6].
- Παρ' όλες τις προφυλάξεις, έχουν παρατηρηθεί διάφορες επιπλοκές και παραλείψεις στην πράξη, όπως:

-ψηφοφόροι μπορούν να ψηφίσουν περισσότερες από μία φορές, χωρίς αυτό να μπορεί να ανιχνευθεί

-ένα ενδεχόμενο κύμα ψηφοφόρων, είτε τις πρώτες ώρες της ψηφοφορίας (πολλοί είναι αυτοί που θέλουν να είναι οι πρώτοι που ψηφίζουν online) είτε λίγο πριν κλείσουν οι ηλεκτρονικές κάλπες καθιστά εξαιρετικά πιθανή την κατάρρευση του συστήματος και πρέπει να έχουν προβλεφθεί εκ των προτέρων σχετικά μέτρα αντιμετώπισης

-οποιοσδήποτε με πρόσβαση στις μηχανές μπορεί να εκτελέσει διαχειριστικές λειτουργίες, όπως να δει προσωρινά αποτελέσματα ή να διακόψει νωρίτερα την ψηφοφορία

-έχουν παρουσιαστεί σοβαρά προβλήματα σε ψηφοφόρους με παλαιοτέρους browsers και σε ψηφοφόρους που χρησιμοποιούν Macintosh

-η διαδικασία διανομής εκατοντάδων χιλιάδων ψηφιακών πιστοποιητικών για τις ηλεκτρονικές υπογραφές στους εγγεγραμμένους ψηφοφόρους δεν έχει τελειοποιηθεί ακόμα, με αποτέλεσμα να δημιουργούνται επιπρόσθετα κενά ασφάλειας

-η επικοινωνία μεταξύ τερματικών και κεντρικού εξυπηρετητή δεν κρυπτογραφούνται επαρκώς, αφήνοντας «τρύπα» για ενδιάμεση κακόβουλη επέμβαση [11][28].

- Δυσκολίες στην αξιολόγηση των εκάστοτε τεχνολογιών δημιουργεί και η closed-source προσέγγιση που επικρατεί μέχρι τώρα [11].
- Την παρακολούθηση της διαδικασίας και την καταμέτρηση των ψήφων αναλαμβάνει ένας «αντικειμενικός παρατηρητής». Αυτός συνήθως είναι μία εταιρεία, όμως τόσο το ποιος την διοικεί όσο και οι πολιτικές του πεποιθήσεις σπάνια είναι γνωστά, με αποτέλεσμα αυτή η αντικειμενικότητα συχνά να αμφισβητείται.

Σε ό,τι αφορά τις πιθανές επιθέσεις (hacks) θα πρέπει γενικά να έχουμε υπ' όψη μας, ότι η παραγωγή λογισμικού απαλλαγμένου από κάθε κενό ασφάλειας είναι σημαντικά δυσκολότερη από το σκοπό του επιτιθέμενου: να βρει και να εκμεταλλευτεί ένα και μοναδικό bug. Οι πιο συνήθεις απειλές για συστήματα ηλεκτρονικής ψηφοφορίας, που έχουν εντοπιστεί μέχρι σήμερα, καταγράφονται συνοπτικά στον παρακάτω πίνακα.

Απειλή	**Επίπεδο ικανοτήτων που απαιτούνται**	**Συνέπειες**	**Πιθανότητες να συμβεί**	**Μέτρα Αντιμετώπισης**
Denial of Service attack	χαμηλό	παραβίαση πολιτικών δικαιωμάτων (πιθανώς επιλεκτικά)	σύνηθες στο Internet	όχι με απλά μέσα: απαιτεί ώρες εργασίας από μηχανικούς δικτύων
Trojan horse attack στο pc με στόχο να παρεμποδιστεί η ψηφοφορία	χαμηλό	παραβίαση πολιτικών δικαιωμάτων	υπάρχουν εκατομμύρια τρόποι για να πραγματοποιηθεί μια τέτοια σύνθετη εργασία	μπορεί να μετριασθεί ο κίνδυνος με προσεχτικό έλεγχο του λογισμικού
on-screen electioneering	χαμηλό	ενόχληση ψηφοφόρου, απογοήτευση, περισπασμός, ανάρμοστη επιρροή	υπερβολικά εύκολο με τη σημερινή web τεχνολογία	ο ψηφοφόρος δεν έχει τη δυνατότητα να το εμποδίσει- απαιτείται νέος νόμος
Spoofing (εξαπάτηση διαφόρων ειδών)	χαμηλό	κλοπή ψήφων, διακύβευση απορρήτου, παραβίαση πολιτικών δικαιωμάτων	σύνηθες και σχετικά εύκολο	δεν αντιμετωπίζεται με κάποιο τρόπο- πολύ πιθανόν να μην ανιχνευθεί
Εξαπάτηση ψηφοφόρου	χαμηλό	παραβίαση πολιτικών δικαιωμάτων	π.χ. : παραποίηση αδειών στα αρχεία cookie	δύσκολο να προβλεφθούν και κυρίως να ανιχνευθούν όλες οι πιθανές απόπειρες εξαπάτησης

Εσωτερικές επιθέσεις στους servers του συστήματος	μέσο	ολοκληρωτική διακύβευση της εκλογικής διαδικασίας	είναι οι πιο συνήθεις, επικίνδυνες και οι πιο δύσκολα ανιχνεύσιμες παραβιάσεις	επιθυμητό να επαληθεύει ο ψηφοφόρος την ψήφο του
Αγοραπωλησία ψήφων	μέσο	κατάλυση της δημοκρατίας	απόλυτα ρεαλιστικό, από τη στιγμή που συμμετέχει και ο ίδιος ο ψηφοφόρος	δεν υπάρχουν – η εκάστοτε νομοθεσία είναι πιθανόν να μην αγγίζει τους αγοραστές
Εξαναγκασμός	μέσο	κατάλυση της δημοκρατίας	πιο δύσκολο από την προηγούμενη περίπτωση αλλά κατορθωτό	δεν υπάρχουν – συνήθως δεν ανιχνεύεται
Trojan Horse Attack με στόχο την παραποίηση ψήφων ή την παρακολούθηση τους	υψηλό	κλοπή ψήφων, διακύβευση απορρήτου	ευρέως διαθέσιμο spyware είναι ένα καλό σημείο εκκίνησης	μπορεί να μετριασθεί ο κίνδυνος με προσεχτικό έλεγχο του λογισμικού – δυσκολότερα ελέγχεται σε cyber- cafés ή σε άλλα θεσμικά διαχειριζόμενα δίκτυα – δύσκολα ανιχνεύσιμο

ΠΙΝΑΚΑΣ 2.1 Ο πίνακας περιγράφει, για κάθε πιθανή απειλή των συστημάτων ηλεκτρονικής ψηφοφορίας, τι ικανότητες απαιτούνται από τον εισβολέα, ποιες είναι οι συνέπειες μιας επιτυχημένης τελικά επίθεσης, πόσο πιθανό είναι να παρουσιαστεί η καθεμία και με ποιον τρόπο μπορεί να αντιμετωπιστεί.Πηγή: [25]

Ακολουθεί, τέλος, μια κατηγοριοποίηση των επιθέσεων, ανάλογα με τα στάδια της διαδικασίας ψηφοφορίας, στα οποία αυτά υφίστανται, καθώς και η αναφορά μερικών επιπλέον:

Voting platform

Κακόβουλο φορτίο (malicious payload). Όταν φτάσει έναν voting host μπορεί να κάνει πρακτικά τα πάντα, ακόμα και να αλλάξει την ψήφο, χωρίς να γίνει αντιληπτό, ανεξαρτήτως κρυπτογραφίας ή πιστοποίησης στην πλευρά του ψηφοφόρου, κι αυτό γιατί κάνει τη ζημία πριν εφαρμοστούν οι αντίστοιχες διαδικασίες πάνω στα δεδομένα. Στη συνέχεια αυτοδιαγράφεται, ώστε να μην μπορεί να ανιχνευθεί. Συχνά χρησιπομοιούνται για τέτοιους σκοπούς τα λεγόμενα προγράμματα stealth, τα οποία δεν ανιχνεύονται ακόμα και κατά την εκτέλεσή τους.

Εγκατάσταση (Delivery mechanism). Η απευθείας εγκατάσταση προγράμματος σε έναν host είναι ένας ακόμα τρόπος, ωστόσο λιγότερο αποδοτική από την remote εγκατάσταση. Η τελευταία πραγματοποιείται π.χ. μέσω ιών που μεταφέρονται μέσω e-mail ή downloads. Επίσης τα άφθονα bugs των κοινών λειτουργικών συστημάτων (ΛΣ) επιτρέπουν την εξ

αποστάσεως εγκατάσταση κακόβουλου κώδικα. Το συνηθέστερο παράδειγμα στα σημερινά συστήματα είναι το buffer overflow, που επιτρέπει το χειρισμό της μνήμης από τον επιτιθέμενο. Οι Δούρειοι Ίπποι («κρυμμένα» προγράμματα) επίσης μπορούν να φτάσουν μέσω internet σε έναν host. Πέρα όμως από τους κινδύνους που ενυπάρχουν στα downloads και στην εγκατάσταση λογισμικού από το internet, μεγάλη δύναμη έχουν και οι εταιρίες λογισμικού με ευρέως διαδεδομένα προϊόντα, καθώς μια αλλαγή σε οποιοδήποτε configuration file για εισαγωγή συγκεκριμένου κώδικα αρκεί.

Communications Infrastructure

DOS attacks: ακινητοποιούν το σύστημα

The ping of death: ένα πακέτο χωρίζεται στα δύο, μόλις ο υπολογιστής-στόχος τα επανασυνδέει, το μήνυμα που προκύπτει είναι υπερβολικά μεγάλο για τις δυνατότητες του ΛΣ, οπότε το σύστημα καταρρέει.

Social Engineering

Είναι ο όρος που περιγράφει επιθέσεις με στόχο την εξαπάτηση του κοινού ώστε ακούσια να διακινδυνεύει την ασφάλειά του.

Οι αντίστοιχες διεπαφές μπορεί σε κάποιες περιπτώσεις, κακώς βέβαια, να μην είναι αρκετά φιλικές προς το χρήστη, ο οποίος συχνά δεν είναι ιδιαίτερα εξοικειωμένος με υπολογιστές. Ακόμα και στην αντίθετη περίπτωση όμως, ο χρήστης δεν έχει τις απαιτούμενες γνώσεις ώστε να μπορεί να διακρίνει, για παράδειγμα, ανάμεσα σε μια SSL σύνδεση (συνήθως) προς το νόμιμο εξυπηρετητή (server) και σε μια ακριβώς ίδια σελίδα ενός κακόβουλου εξυπηρετητή. Έτσι μπορούν να στηθούν εικονικά sites ψηφοφορίας, στέλνοντας π.χ. στο χρήστη το αντίστοιχο link, πιστεύοντας ο τελευταίος ότι εκεί πρέπει στην πραγματικότητα να ψηφίσει. Μπορεί επίσης ο επιτιθέμενος να στήσει μια σύνδεση που θα φτάνει μεν στο νόμιμο εξυπηρετητή, θα περνάει όμως από μια τρίτη σελίδα web, αλλάζοντας ως ενδιάμεσος τα αποτελέσματα.

Ακόμα πιο σοβαρή είναι η ενδεχόμενη επίθεση στην υπηρεσία DNS (Domain Name Service) του Internet, ώστε ο χρήστης να οδηγηθεί σε λάθος IP διεύθυνση, δηλαδή στη «στημένη» σελίδα. Τέλος, υπάρχει η δυνατότητας της «εικονικής» σύνδεσης: ο χρήστης νομίζει ότι είναι συνδεδεμένος κανονικά στον κατάλληλο εξυπηρετητή, ενώ στην πραγματικότητα δεν υπάρχει καμία σύνδεση, προς πουθενά. Δε χρειάζεται καν σύνδεση στο δίκτυο. Αυτό μπορεί, για παράδειγμα, να χρησιμοποιηθεί για τον αποκλεισμό περιοχών που θα έδιναν συγκεκριμένο εκλογικό αποτέλεσμα.

Γενικότερα, πέρα από την τεχνολογική και νομική οπτική γωνία, υπάρχει πάντα και η κοινωνική. Κυριαρχεί η άποψη, ότι τέτοιου τύπου ψηφοφορία έρχεται να εισάγει μια νέα εποχή κοινωνικών διακρίσεων. Είναι προφανές ότι η πλειοψηφία των ηλικιωμένων και χαμηλού επιπέδου μόρφωσης ψηφοφόρων θα δυσκολευτεί σε μεγάλο βαθμό να εξοικειωθεί με τη νέα τεχνολογία, ενώ ειδικά σε ότι αφορά το remote Internet voting, μεγάλο πρόβλημα θα αντιμετωπίσουν νοικοκυριά με οικονομικά προβλήματα, που ενδεχομένως δεν διαθέτουν ηλεκτρονικό υπολογιστή. Φυσικά όλα τα παραπάνω προβλήματα αντιμετωπίζουν με μεγαλύτερη ένταση οι μειονότητες μιας χώρας, π.χ. οι οικονομικοί μετανάστες. Αυτή η νέα κοινωνική διάκριση μπαίνει κάτω από το μικροσκόπιο των εκάστοτε συμφερόντων, που αποσκοπούν πλέον να ασκήσουν άμεση επιρροή στα άτομα που έχουν συνεχή πρόσβαση στο Internet, σε βάρος συνήθως των υπολοίπων [14].

2.5 Προτάσεις

Στο επίπεδο της ψηφοφορίας είναι παραπάνω από προφανής η αναγκαιότητα για καλύτερη ενημέρωση των ψηφοφόρων πάνω σε θέματα τεχνολογίας και για ευκολότερη πρόσβαση στις ηλεκτρονικές κάλπες ώστε να εξαλειφθεί κάθε φαινόμενο διακρίσεων. Στο

επίπεδο της καταγραφής των ψήφων για την εξακρίβωση της ασφάλειας της διαδικασίας έχουν προταθεί ή και εφαρμοστεί διάφοροι τρόποι.

Με βάση ονομαστικούς καταλόγους ή και αύξοντες αριθμούς ψηφοφόρων γίνονται απλοί λογιστικοί υπολογισμοί, που μπορούν να γίνουν από τον καθένα (θα πρέπει να πρόκειται για ανοιχτή διαδικασία) τόσο στο τέλος όσο και κατά τη διάρκεια της διαδικασίας, ώστε να διαγνωστούν εγκαίρως λάθη και προσπάθειες νοθείας. Τέτοιοι απλοί αριθμητικοί υπολογισμοί ωστόσο δεν καλύπτουν την περίπτωση αφαίρεσης ψήφων από έναν υποψήφιο και προσθήκης τους σε άλλον.

Μετά το τέλος της διαδικασίας, και ειδικά για ηλεκτρονικά συστήματα, μπορούν να χρησιμοποιηθούν τα αρχεία (records) των αντίστοιχων συσκευών, τα οποία περιλαμβάνουν πληροφορίες μεταξύ άλλων για τη χρονική στιγμή ρίψης κάθε ψηφοδελτίου, χωρίς αυτή βέβαια να μπορεί να συνδυαστεί με συγκεκριμένο ψηφοδέλτιο. Σύγκριση του αριθμού αυτών των χρονικών στιγμών και των ψηφοδελτίων οδηγεί σε έγκυρα συμπεράσματα για την εξέλιξη της διαδικασίας. Από κει και πέρα είναι απαραίτητη η δημιουργία «εμποδίων» και firewalls στο εσωτερικό του ίδιου του συστήματος, όπως με την κρυπτογράφηση ψήφων με διαφορετικά κλειδιά, την περιορισμένη πρόσβαση σε κρίσιμες παραμέτρους του συστήματος κλπ. Αποδοτικός φαίνεται να είναι και ο λεγόμενος παράλληλος έλεγχος, το να επιλέγονται, δηλαδή, κατά διαστήματα τυχαία κάποια μηχανήματα και να ελέγχεται η λειτουργία τους.

Σημαντική στις εκλογές είναι η δυνατότητα επανακαταμέτρησης (recount), κάτι που έρχεται σε αντίφαση με την ψηφιακή φύση της ηλεκτρονικής διαδικασίας. Μια λύση θα ήταν η χρήση διαφορετικού αλγορίθμου στη δεύτερη καταμέτρηση, κάτι που όμως έχει παρουσιάσει προβλήματα. Η επικρατέστερη πρόταση είναι να προστεθεί η ιδέα ενός χάρτινου «ίχνους» (paper audit trail), οι μηχανές ψηφοφορίας να παράγουν, δηλαδή, ένα χάρτινο αντίτυπο της ψήφου, το οποίο θα αποτελεί την υλική απόδειξη της πρόθεσης του ψηφοφόρου. Ο τελευταίος θα έχει έτσι τη δυνατότητα να επαληθεύσει ότι η ψήφος του έχει καταγραφεί σωστά και οι υπεύθυνοι να μετρήσουν τα αντίτυπα -είτε μηχανικά είτε με το χέρι- ώστε να εξασφαλίσουν ακρίβεια. Έτσι επιτυγχάνεται η ανεξαρτησία ενός κρίσιμου μέρους της διαδικασίας από την ορθότητα ή μη (bugs και δυνατότητα δολιοφθοράς) του λογισμικού (software) [5].

Σε αυτή την κατεύθυνση έχει προταθεί ένας νέος τύπος «απόδειξης» ψήφου, που χρησιμοποιεί κρυπτογραφία για ακόμα μεγαλύτερη ασφάλεια. Η ιδέα είναι η εξής:

Στο θάλαμο ο ψηφοφόρος μπορεί να δει τις επιλογές του καθαρά τυπωμένες στην απόδειξη. Αφού βγει από το θάλαμο, μπορεί να ελέγξει αν οι ψήφοι που περιέχει έχουν καταμετρηθεί σωστά στο τέλος (με βάση μόνο ένα σειριακό αριθμό (serial number)). Επειδή όμως οι επιλογές έχουν κρυπτογραφηθεί πριν βγει από το θάλαμο, η απόδειξη δεν μπορεί να χρησιμοποιηθεί για να δείξει σε κάποιον τρίτο τι έχει ψηφίσει [12].

Μια άλλη προοπτική που υπάρχει, κυρίως στο θέμα του authentication, είναι κάποιες ανθεκτικές στην παραβίαση συσκευές, όπως οι λεγόμενες «έξυπνες» κάρτες, που μπορούν να παράγουν και αποθηκεύουν κρυπτογραφικά κλειδιά και να εκτελούν υπολογισμούς, ώστε τα κατάλληλα διαπιστευτήρια να ανταλλάσσονται ανάμεσα στον host και τον εξυπηρετητή. Ωστόσο τα PC των πολιτών κατά κανόνα δε διαθέτουν το κατάλληλο λογισμικό ανάγνωσης έξυπνων καρτών και οποιαδήποτε οικονομική επιβάρυνσή τους, προκειμένου να ψηφίσουν, θα ήταν ανεπίτρεπτη. Ακόμα και σε αυτή την περίπτωση, όμως, το ζήτημα της ασφάλειας παραμένει, καθώς οι κάρτες δεν αλληλεπιδρούν απευθείας με τον εξυπηρετητή. Η επικοινωνία λαμβάνει χώρα μέσω του υπολογιστή, οπότε η λειτουργία της κάρτας μπορεί να επηρεαστεί από το κατάλληλο κακόβουλο λογισμικό [14].

Μια άλλη άποψη, που υποστηρίζεται συχνά, είναι ότι μια open-source προσέγγιση σε ότι αφορά το λογισμικό θα συμβάλλει ουσιαστικά στην αύξηση της ασφάλειας και αξιοπιστίας. Η closed-source προσέγγιση, που ακολουθείται από τα περισσότερα συστήματα (proprietary software) , εμποδίζει την ελεύθερη και δημόσια αξιολόγησή τους και καθυστερεί τη λεπτομερή εξέτασή τους, οπότε και τις απαραίτητες βελτιώσεις και προόδους [11].[26].

Άλλες προτάσεις που έχουν δημοσιευτεί σε κατά καιρούς μελέτες είναι:

-η χρήση της γλώσσας Typed MSR (strongly typed specification language για πρωτόκολλα ασφάλειας, που αποσκοπεί στον εντοπισμό λαθών στο σχεδιασμό τους) στην ανάπτυξη πρωτοκόλλων ηλεκτρονικής ψηφοφορίας και άλλων που απαιτούν προστασία του

προσωπικού απορρήτου και που βασίζονται σε ψηφιακές υπογραφές και ομομορφική κρυπτογραφία [19].

-χρήση διαφοροποιημένων πακέτων λογισμικού από διαφορετικούς κατασκευαστές ανά server ή ντετερμινιστικών διαδικασιών σε συστήματα υπογραφών όπως το RSA, παρέχουν απρόβλεπτα αποτελέσματα για τον εξωτερικό παρατηρητή. Τέτοια μέτρα κρίνονται απαραίτητα ιδίως για την περίπτωση της κλεπτογραφίας, μιας από τις πιο επικίνδυνες τεχνικές σε επίπεδο κώδικα εναντίον της ασφάλειας κρυπτογραφικών συστημάτων [20].

-πρωτόκολλα κρυπτογραφίας που ανακατεύουν τις ψήφους με δυνατότητα απόδειξης της ορθότητάς τους, συμβάλλοντας στην προστασία της ανωνυμίας [21].

-η παρουσίαση το Φεβρουάριο του 2006 της γλώσσας EML (Election Markup Language) από την τεχνική επιτροπή Εκλογών και Υπηρεσιών Ψηφοφορίας του OASIS (Organization for the Advancement of Structured Information Standards), που φιλοδοξεί να καλύψει από άκρο σε άκρο την ηλεκτρονική ψηφοφορία, από την εγγραφή σε καταλόγους μέχρι την ψηφοφορία και καταγραφή, είτε πρόκειται για ψηφοφορία μέσω Internet είτε για ψηφοφορία στο εκλογικό κέντρο [17].

2.6 Βιβλιογραφία

[1] Herb Deutsch and Stephen Berger, "Voting Systems: Standards and Certifications", COMMUNICATIONS OF THE ACM October 2004/Vol. 47, No. 10

[2] Carolyn Coggins, "Independent Testing of Voting Systems", COMMUNICATIONS OF THE ACM October 2004/Vol. 47, No. 10 34

[3] Brit J. Williams *AND* Merle S. King, "Implementing Voting Systems: the Georgia Method", COMMUNICATIONS OF THE ACM October 2004/Vol. 47, No. 10 39

[4] Anthony Di Franco, Andrew Petro, Emmett Shear**,** *AND* Vladimir Vladimirov, "Small Vote Manipulations Can Swing Elections", COMMUNICATIONS OF THE ACM October 2004/Vol. 47, No. 10 43

[5] Douglas W. Jones, "Auditing", October 2004/Vol. 47, No. 10 COMMUNICATIONS OF THE ACM 46

[6] Rebecca T. Mercuri *AND* L. Jean Camp, "The Code of Elections", COMMUNICATIONS OF THE ACM October 2004/Vol. 47, No. 10 53

[7] David Jefferson, Aviel D. Rubin, Barbara Simons *AND* David Wagner, "Analyzing Internet Voting Security", COMMUNICATIONS OF THE ACM October 2004/Vol. 47, No. 10 59

[8] Jason Kitcat, "Source Availability and e-Voting: an Advocate Recants", COMMUNICATIONS OF THE ACM October 2004/Vol. 47, No. 10 65

[9] Jeff Grove, "ACM Statement of Voting Systems", COMMUNICATIONS OF THE ACM October 2004/Vol. 47, No. 10 65

[10] David Evans and Nathanael Paul, *University of Virginia*, "Election Security: Perception and Reality", IEEE SECURITY & PRIVACY, 1540-7993/04/$20.00 © 2004 IEEE

[11] Jonnathan Bannet, Davidw.Price, Algis Rudys, Justin Singer & Dans.Wallach, *Rice University,* "Hack-a-vote: Security issues with electronic voting systems", 1540-7993/04/$20.00 © 2004 IEEE

[12] David Chaum, "Secret Ballot Receipts: True Voter-Verifiable Elections", 1540-7993/04/$20.00 © 2004 IEEE

[13] UK Electoral Commission, August 2002, "Modernising Elections --- A strategic evaluation of the 2002 electoral pilot schemes", "Securing the vote --- Report and recommendations"

[14] Aviel D. Rubin, "Security Considerations for Remote Electronic Voting", COMMUNICATIONS OF THE ACM December 2002/Vol. 45, No. 12

[15] Dimitris Gritzalis, "Secure Electronic Voting", 7th Computer Security Incidents Response Teams Workshop, Syros, Greece, September 2002

[16] Alexandros Xenakis and Prof. Ann Macintosh, "E-electoral Administration: Organizational Lessons Learned from the Deployment of E-voting in the UK"

[17] Greg Goth, "News: Cradle of Liberty Lags on E-Voting", IEEE DISTRIBUTED SYSTEMS ONLINE 1541-4922 © 2006 Published by the IEEE Computer Society, Vol. 7, No. 4; April 2006

[18] Alexander H. Trechsel, Fabian Breuer, "E-voting in the 2005 Local Elections in Estonia and the broader impact for future e-voting projects"

[19] Theodoros Balopoulos, Stefanos Gritzalis, Sokratis K. Katsikas, "Specifying Electronic Voting Protocols in Typed MSR"

[20] Przemysław Kubiak, Mirosław Kutyłowski, Filip Zagórski, “Kleptographic Attacks on a Cascade of Mix Servers”
[21] C. Andrew Neff, “A Verifiable Secret Shuffle and its Application to EVoting”, 2001, Philadelphia, Pennsylvania, USA.
[22] Orhan Cetinkaya, Deniz Cetinkaya, “Towards Secure E-Elections in Turkey: Requirements and Principles”, Second International Conference on Availability, Reliability and Security (ARES'07) © 2007, IEEE
[23] Part of the Voting and Elections web pages by Douglas W. Jones, THE UNIVERSITY OF IOWA Department of Computer Science, “A Brief Illustrated History of Voting”
[24] Peter G. Neumann, “The problems and potentials of voting systems”, COMMUNICATIONS OF THE ACM October 2004/Vol. 47, No. 10
[25] Dr. David Jefferson, Dr. Aviel D. Rubin, Dr. Barbara Simons, Dr. David Wagner, “A Security Analysis of the Secure Electronic Registration and Voting Experiment (SERVE)”, January 21, 2004
[26] Joana Matos Penha-Lopes, “Why Use an Open Source E-Voting System?”, ACM
[27] Wikipedia,the free encyclopedia, “Electronic voting”, Available: http://en.wikipedia.org/wiki/Electronic_voting
[28] Joe Mohen and Julia Glidden, “The case for Internet Voting”, January 2001/Vol. 44, No. 1 COMMUNICATIONS OF THE ACM

3 Κινητοί πράκτορες και ασφάλεια

3.1 Κινητοί πράκτορες

3.1.1 Πρώιμες τεχνολογίες-κατανεμημένα συστήματα

Η λογική, στην οποία βασίζεται η κατανεμημένη επεξεργασία, είναι η υπόθεση ότι, μια εφαρμογή λογισμικού μπορεί να χωριστεί σε θεμελιώδεις συνιστώσες λογισμικού (modules), από τις οποία αποτελείται. Οι συνιστώσες αυτές δεν είναι απαραίτητο να εκτελούνται στον ίδιο χώρο μνήμης. Σύμφωνα με τη θεώρηση αυτή, προκειμένου να εκτελεστεί η εφαρμογή, θα πρέπει να υπάρχει επικοινωνία μεταξύ των κομματιών λογισμικού. Οι όροι "πελάτης", "εξυπηρετητής" και "ομότιμη οντότητα" χρησιμοποιούνται για να περιγράψουν τους ρόλους των συνιστωσών λογισμικού κατά τη διαδικασία εκτέλεσης της εφαρμογής. Οι συνιστώσες, οι οποίες ζητούν κάποια υπηρεσία από άλλες συνιστώσες, ονομάζονται "πελάτες", ενώ οι συνιστώσες οι οποίες προσφέρουν υπηρεσίες ονομάζονται "εξυπηρετητές". Βέβαια, κατά την εκτέλεση της εφαρμογής, οι ρόλοι μπορούν να μεταβάλλονται. Για παράδειγμα, μια συνιστώσα λογισμικού μπορεί, σε κάποια περίπτωση, να λειτουργεί ως πελάτης ενώ, σε κάποια άλλη, ως εξυπηρετητής. Στην περίπτωση που μια συνιστώσα λογισμικού δρα ταυτόχρονα ως πελάτης και ως εξυπηρετητής, αυτή χαρακτηρίζεται με τον όρο "ομότιμη οντότητα" (peer entity).

Τα πρώτα κατανεμημένα συστήματα βασίζονταν σε *στατικές διεργασίες* οι οποίες εκτελούνταν σε απομακρυσμένους εξυπηρετητές και επικοινωνούσαν με σύγχρονες ή ασύγχρονες μεθόδους επικοινωνίας, π.χ. μέσω *κλήσεων απομακρυσμένων διαδικασιών* (RPC-Remote Procedure Calls). Όμως, οι συχνές κλήσεις απομακρυσμένων διαδικασιών κοστίζουν σε εύρος ζώνης δικτύου, το οποίο κόστος γίνεται μεγαλύτερο όσο αυξάνει και το πλήθος των απομακρυσμένων κλήσεων.

Για την ελάττωση αυτού του κόστους, τα πρώτα κατανεμημένα συστήματα υιοθέτησαν μια μέθοδο γνωστή ως *μετανάστευση διαδικασίας* (process migration), κατά την οποία μια διαδικασία μπορούσε να μεταφερθεί από έναν υπολογιστή σε έναν άλλο, ώστε να εκτελεστεί εκεί τοπικά, χωρίς απομακρυσμένες κλήσεις. Η μετανάστευση διαδικασιών, παρόλο που βοήθησε στην ελάττωση του κόστους επικοινωνίας, δεν επέτρεπε την εύκολη επιστροφή δεδομένων, όπως αποτελέσματα υπολογισμών, χωρίς την επιστροφή ολόκληρης της διαδικασίας.

Η δυνατότητα αυτή δόθηκε με τη χρήση της μεθόδου *απομακρυσμένης αποτίμησης* (remote evaluation), σύμφωνα με την οποία αιτήσεις εκτέλεσης αποστέλλονται με τη μορφή προγραμμάτων. Μετά τη λήψη μιας τέτοιας αίτησης, ο υπολογιστής εκτελεί το πρόγραμμα χρησιμοποιώντας τους τοπικούς πόρους του (μνήμη, επεξεργαστή κλπ.) και τελικά επιστρέφει τα αποτελέσματα στον αποστολέα ([1].[2].[3]).

3.1.2 Κινητός κώδικας

Οι τεχνολογίες *κινητού κώδικα* (mobile code), στηριζόμενες σε τεχνικές αντικειμενοστραφούς προγραμματισμού, επέκτειναν τη μέθοδο απομακρυσμένης αποτίμησης. Οι κινητοί κώδικες μπορούν να μεταναστεύσουν από έναν υπολογιστή σε έναν άλλον μεταφέροντας εκτελέσιμο κώδικα, δεδομένα με τη μορφή ιδιοτήτων αντικειμένων και πιθανώς άλλα ενσωματωμένα εκτελέσιμα αντικείμενα. Διακρίνονται δύο κατηγορίες τεχνολογιών κινητού κώδικα: *ασθενείς* και *ισχυρές* .

Οι ασθενείς τεχνολογίες κινητού κώδικα παρέχουν την υποδομή για απομακρυσμένη εκτέλεση κώδικα. Επιτρέπουν σε εφαρμογές να στέλνουν κώδικα σε έναν απομακρυσμένο

υπολογιστή ώστε να εκτελεστεί εκεί τοπικά, ή να ανακτήσουν και να συνδέσουν δυναμικά κώδικα από μία απομακρυσμένη τοποθεσία, ώστε να τον εκτελέσουν τοπικά. Ο μεταφερόμενος κώδικας μπορεί να περιλαμβάνει κάποια δεδομένα εκκίνησης αλλά όχι την κατάσταση εκτέλεσης. Παραδείγματα τέτοιων τεχνολογιών είναι τα Java applets , τα ActiveX controls και η πλατφόρμα Aglets. Από την άλλη πλευρά, οι ισχυρές τεχνολογίες κινητού κώδικα παρέχουν την υποδομή για την εκτέλεση κινητών πρακτόρων [1].

3.1.3 Κινητοί πράκτορες

3.1.3.1 Εισαγωγή

Με τον γενικό όρο «πράκτορας λογισμικού» ([2][3])αναφερόμαστε σε μια αυτόνομη οντότητα λογισμικού, η οποία αυτοματοποιεί κάποιες δραστηριότητες που ένας άνθρωπος ή οποιαδήποτε διεργασία λογισμικού μπορεί να έχει μεταβιβάσει σε αυτήν. Κάθε πράκτορας λογισμικού έχει τη δική του πληροφορία κατάστασης, τη δική του συμπεριφορά καθώς και το δικό του νήμα ελέγχου και μπορεί να αλληλεπιδρά ελεύθερα με άλλες οντότητες με σκοπό την επίλυση κάποιου συγκεκριμένου προβλήματος.

3.1.3.2 Ιστορική αναδρομή

Η έννοια του πράκτορα λογισμικού υπάρχει από τη δεκαετία του 1970, όταν οι Henry Baker και Carl Hewitt πρότειναν το μοντέλο του «ηθοποιού» (Actor model)[3]. Σε αυτό το μοντέλο, ο Hewitt πρότεινε την έννοια ενός αντικειμένου αυτόνομου και διαδραστικού που μπορεί να εκτελείται ταυτόχρονα με άλλα τέτοια όμοια αντικείμενα. Αυτό το μοντέλο αντικειμένου περιείχε ενσωματωμένη μια εσωτερική κατάσταση και μπορούσε να απαντά σε μηνύματα που προέρχονταν από άλλα όμοια αντικείμενα. Η έρευνα που ξεκίνησε τότε, και που ακόμη συνεχίζεται σήμερα, επικεντρωνόταν κυρίως στην αλληλεπίδραση και την επικοινωνία μεταξύ των πρακτόρων, τη διάσπαση και την κατανομή εργασιών, το συντονισμό και τη συνεργασία μεταξύ των πρακτόρων, την επίλυση συγκρούσεων και άλλα παρεμφερή θέματα.

Από τις αρχές της δεκαετίας του 1990 άρχισε να γίνεται μια πιο εξειδικευμένη έρευνα στα διαφορετικά πιθανά είδη των πρακτόρων λογισμικού και γενικά σε αυτό που συχνά ονομάζεται «τυπολογία» των πρακτόρων. Από αυτήν την περίοδο ξεκινάει και η κύρια εξέλιξη στην τεχνολογία των Κινητών Πρακτόρων. Με την ταυτόχρονη εμφάνιση κάποιων παρεμφερών εννοιών και τεχνολογιών, άρχισε να γίνεται διάκριση ανάμεσα σε Ευφυείς και Κινητούς Πράκτορες. Λίγο αργότερα εμφανίστηκαν και τα πρώτα συστήματα πολλαπλών πρακτόρων (Multi Agent Systems – MAS), τα οποία με τη σειρά τους χωρίζονται σε τρεις γενικές κατηγορίες: Συστήματα Κατανεμημένης Τεχνητής Νοημοσύνης (Distributed Artificial Intelligence - DAI), Συστήματα Παράλληλης Τεχνητής Νοημοσύνης (Parallel Artificial Intelligence - PAI) και Συστήματα Κατανεμημένης Επίλυσης Προβλημάτων (Distributed Problem Solving - DPS).

Την τελευταία δεκαετία έχουν γίνει αρκετές προσπάθειες για την ανάπτυξη συστημάτων κινητών πρακτόρων και τεχνολογιών που θα διευκολύνουν και θα κάνουν πιο αποδοτική τη χρήση τους. Το 1994 αναπτύχθηκε από την General Magic η γλώσσα TeleScript και στη συνέχεια άλλες διερμηνευόμενες γλώσσες, (scripting languages) όπως η Tool Command Language (TCL) και η SafeTCL που επέτρεπαν τη γρήγορη προτυποποίηση (prototyping) και παραγωγή μεταφέρσιμου κώδικα. Η TCL ήταν στην πραγματικότητα ένα πλήρες περιβάλλον εκτέλεσης κινητών πρακτόρων.

Ένα από τα προτερήματα της TeleScript ήταν ότι ενσωμάτωνε ένα πλήρες Περιβάλλον Εκτέλεσης Κινητών Πρακτόρων (Mobile Agent Execution Environment). Το περιβάλλον αυτό είχε σχεδιασθεί έτσι ώστε να επιτρέπει την υλοποίηση μεθόδων απομακρυσμένου προγραμματισμού (Remote Programming). Η βασική ιδέα ήταν ότι μικρά κομμάτια κώδικα θα μεταφέρονταν εκεί που βρισκόντουσαν μεγάλες ποσότητες δεδομένων, μειώνοντας έτσι τον φόρτο του δικτύου. Η TeleScript επέτρεπε την ύπαρξη αυτόνομων πρακτόρων, οι οποίοι μπορούσαν να επικοινωνήσουν μεταξύ τους ή με τον χρήστη και να μεταναστεύσουν σε απομακρυσμένους κόμβους προκειμένου να εκτελέσουν την εργασία που έχουν αναλάβει. Ο κάθε πράκτορας είχε τη δυνατότητα να αναστείλει τη λειτουργία του, να μεταφέρει τον

κώδικα, την κατάσταση εκτέλεσης και τα δεδομένα του και να μεταφερθεί σε κάποιον άλλο κόμβο για να συνεχίσει τη λειτουργία του[3].

Με την εμφάνιση της Java δόθηκε νέα ώθηση στην ανάπτυξη της τεχνολογίας των Κινητών Πρακτόρων (Mobile Agent Technology – MAT). Η Java παρέχει εγγενώς πολλές δυνατότητες για κινητικότητα κώδικα, εξαιτίας της πρότυπης διαδικασίας σειριοποίησης αντικειμένων (object serialization) που εφαρμόζει καθώς και το δυναμικό φόρτωμα κώδικα κατά τη διάρκεια της εκτέλεσης ενός προγράμματος. Στη συνέχεια άρχισαν να εμφανίζονται και κάποιες πλατφόρμες που παρέχουν την απαραίτητη υποδομή για τη χρήση κινητών πρακτόρων, οι σημαντικότερες από τις οποίες είναι η Odyssey της General Magic, η Aglets της IBM, η Grasshopper του ινστιτούτου IKV καθώς και η Jade της Tilab που εμφανίστηκε το τελευταίο διάστημα και από τις οποίες οι περισσότερες αναπτύχθηκαν σε Java ([3]).

3.1.3.3 Χαρακτηριστικά κινητών πρακτόρων και συστημάτων

Όπως προαναφέρθηκε, ένας πράκτορας λογισμικού ορίζεται σαν ένα αντικείμενο που έχει τη δική του συμπεριφορά, κατάσταση, καθώς και τοποθεσία. Οι κινητοί πράκτορες έχουν την ιδιότητα να μεταναστεύουν από έναν υπολογιστή σε έναν άλλο και να συνεχίζουν εκεί την εκτέλεσή τους από το σημείο που σταμάτησαν (κινητικότητα). Κατά τη μετακίνηση αυτή εκτός από τον κώδικα ο πράκτορας παίρνει μαζί του και την πληροφορία κατάστασης (τιμές τοπικών μεταβλητών, ορίσματα και τιμές τοπικών μεταβλητών ρουτινών που έχουν κληθεί στο ίδιο νήμα εκτέλεσης και δεν έχουν τερματίσει, στοίβα εκτέλεσης κτλ). Ένα από τα πλέον βασικά χαρακτηριστικά τους είναι η αυτονομία, αφού από τη στιγμή που θα ξεκινήσει η εκτέλεσή τους μπορούν αυτόνομα να αποφασίσουν σε ποιες τοποθεσίες θα μεταναστεύσουν και ποιον κώδικα θα εκτελέσουν, βασιζόμενοι σε μετα-δεδομένα μετακίνησης (mobility metadata). Οι κινητοί πράκτορες αποτελούν μορφή κινητού κώδικα. Το χαρακτηριστικό της αυτονομίας συμβάλλει και στην βελτίωση του κόστους δικτύου, εφόσον η αλληλεπίδραση ενός πράκτορα με την πηγή προέλευσής του ελαττώνεται στην απολύτως απαραίτητη. Σε αντίθεση με τα κλασσικά παραδείγματα της απομακρυσμένης αξιολόγησης (remote evaluation) και του κατ' απαίτηση κώδικα (code on demand), οι κινητοί πράκτορες είναι ενεργοί υπό την έννοια ότι μπορούν να μεταναστεύσουν μεταξύ υπολογιστών οποιαδήποτε στιγμή κατά την εκτέλεσή τους. Αυτή η ιδιότητα τους καθιστά ένα ιδιαίτερα ισχυρό εργαλείο στην ανάπτυξη κατανεμημένων εφαρμογών πάνω σε ένα δίκτυο υπολογιστών.

Εκτός από την κινητικότητα, την αυτονομία και την ενεργητικότητα, που αποτελούν τα βασικότερα χαρακτηριστικά που πρέπει να έχει ένας κινητός πράκτορας, υπάρχει και ένας αριθμός από άλλα στοιχεία που μπορεί να τον χαρακτηρίζουν [1][3]:

- *διαδραστικότητα*: ένας κινητός πράκτορας πρέπει να έχει τη δυνατότητα να επικοινωνεί και να αλληλεπιδρά με το περιβάλλον του καθώς και με άλλους πράκτορες.
- *προσαρμοστικότητα*: οι κινητοί πράκτορες μπορούν να αποκρίνονται στο περιβάλλον τους ή σε άλλους πράκτορες και να προσαρμόζονται και οι ίδιοι κατάλληλα.
- *αντιπροσωπευτικότητα*: οι κινητοί πράκτορες μπορεί να ενεργούν εκ μέρους ή και προς όφελος κάποιου (ανθρώπου ή υπολογιστικής οντότητας). Για να είναι δυνατόν να γίνει κάτι τέτοιο οι κινητοί πράκτορες πρέπει να έχουν έναν ορισμένο βαθμό αυτονομίας.
- *πρωτοβουλία*: κάθε κινητός πράκτορας θα πρέπει να είναι προσανατολισμένος προς κάποιο συγκεκριμένο στόχο και να παίρνει κατάλληλες πρωτοβουλίες κάθε φορά που ανταποκρίνεται στο περιβάλλον του.
- *νοημοσύνη*: οι κινητοί πράκτορες μπορεί να έχουν νοημοσύνη βασισμένη σε κάποια γνώση, έτσι ώστε να γίνεται αποτελεσματικότερη η αλληλεπίδραση με το περιβάλλον τους.
- *συντονισμός*: οι κινητοί πράκτορες θα πρέπει να έχουν την ικανότητα να πραγματοποιούν μεταφορά δεδομένων που μοιράζονται μεταξύ των πρακτόρων ενός περιβάλλοντος με συγκεκριμένο τρόπο.
- *μάθηση*: αναφέρεται στην ικανότητα ενός κινητού πράκτορα να παίρνει πληροφορίες σχετικά με το περιβάλλον, οι οποίες θα του επιτρέψουν να προσαρμόσει ανάλογα τη συμπεριφορά του.
- *συνεργασία*: οι κινητοί πράκτορες πρέπει να συντονίζονται και να συνεργάζονται μεταξύ τους για την επίτευξη ενός κοινού στόχου.

- *ικανότητα ανάκαμψης*: ένας κινητός πράκτορας θα πρέπει να μπορεί να αντιμετωπίζει τα όποια σφάλματα εμφανίζονται κατά τη διάρκεια ζωής τους.

3.1.3.4 Πλεονεκτήματα των κινητών πρακτόρων

Η τεχνολογία κινητών πρακτόρων λύνει, ή τουλάχιστον υπόσχεται ότι θα λύσει άμεσα, διάφορα προβλήματα σε ποικίλα μέτωπα. Συγκεκριμένα παρουσιάζει τα παρακάτω πολύ σημαντικά πλεονεκτήματα:

- *Ασύγχρονη και αυτόνομη εκτέλεση διεργασιών*: Μετά από την είσοδο ενός πράκτορα σε ένα κατανεμημένο υπολογιστικό σύστημα, ο χρήστης μπορεί να εκτελέσει άλλες δραστηριότητες χωρίς να απαιτείται να αλληλεπιδρά με τον πράκτορα αυτό, ακόμα και να αποσυνδεθεί από το δίκτυο μετά την αποστολή του πράκτορα στον απομακρυσμένο υπολογιστή, όταν πρόκειται για πρόσβαση σε απομακρυσμένες εφαρμογές. Για την υποστήριξη αυτής της συμπεριφοράς χρησιμοποιείται ένας μηχανισμός αποθήκευσης και προώθησης του πράκτορα.
- *Μείωση της δικτυακής κυκλοφορίας και της απαιτούμενης υπολογιστικής ισχύος στην μεριά του πελάτη*: Καθώς ανταλλαγές δεδομένων πολύ μεγάλου όγκου γίνονται πλέον τοπικά από τους κόμβους που περιέχουν αυτήν την πληροφορία, οι υπολογιστές στην μεριά του πελάτη μπορούν να επικεντρωθούν στην εκτέλεση σχετικά απλών λειτουργιών. Αυτό το γεγονός επιτρέπει την υλοποίηση "λεπτών πελατών" (thin clients). Με αυτόν τον τρόπο, οι κινητοί πράκτορες είναι σε θέση να μειώσουν το κόστος στο εύρος ζώνης δικτύου, συγκρινόμενοι με παλαιότερες μεθόδους κατανεμημένης επικοινωνίας.
- *Μείωση των χρονικών καθυστερήσεων λόγω δικτύου*: Οι κινητοί πράκτορες, λόγω των κινητών και προσαρμοστικών ιδιοτήτων τους, προσφέρουν μεγάλες δυνατότητες για απόκριση σε πραγματικού χρόνου συστήματα, όπου η χρονική καθυστέρηση δεν μπορεί να γίνεται αποδεκτή και δημιουργεί προβλήματα.
- *Αυξημένη ευρωστία*: Η μείωση της εξάρτησης μιας κατανεμημένης εφαρμογής από την διαθεσιμότητα του δικτύου είναι ευπρόσδεκτη και λύνει ένα από τα βασικά προβλήματα που αντιμετωπίζουν οι αρχιτεκτονικές πελάτη-εξυπηρετητή. Στα συστήματα κινητών πρακτόρων επέρχεται αυτή η μείωση της εξάρτησης, καθώς από την στιγμή που ο κώδικας του πελάτη βρίσκεται στον ίδιο κόμβο με αυτόν του εξυπηρετητή, η επικοινωνία είναι τοπική.
- *Αυτοματοποίηση της επεξεργασίας κατανεμημένων υπολογισμών*: Οι πράκτορες έχουν ενσωματωμένα δρομολόγια τα οποία καθορίζουν τι είδους δραστηριότητες πρέπει να εκτελέσουν και σε ποιους κόμβους με αποτέλεσμα να μην απαιτείται συνεχής αλληλεπίδραση με τον χρήστη.
- *Αποκεντρωμένη και κατά τόπους επεξεργασία (έλεγχος και διαχείριση):* Η κλωνοποίηση των πρακτόρων επιτρέπει την (αυτοματοποιημένη) κατανομή των υπολογιστικών συνιστωσών προηγούμενα συγκεντρωτικών προγραμμάτων.
- *Ικανότητα εκτέλεσης σε ετερογενή περιβάλλοντα*: οι κινητοί πράκτορες έχουν τη δυνατότητα να εκτελούνται σε περιβάλλοντα με διαφορετικές υποδομές. Έτσι επεκτείνεται η διαλειτουργικότητα σε κατανεμημένα συστήματα. Εφαρμογές μπορούν να εκτελούνται σε διαφορετικές πλατφόρμες και λειτουργικά συστήματα και να επικοινωνούν μέσω κινητών πρακτόρων.
- *Ευελιξία:* Η κατ' απαίτηση διανομή λογισμικού ή παροχή υπηρεσιών γίνεται εφικτή καθώς κινητοί πράκτορες μπορούν να "κατεβούν" στιγμιαία σε οποιοδήποτε κόμβο πελάτη ή εξυπηρετητή.
- *Κλιμάκωση εφαρμογών*: Λόγω της δυναμικής εκτέλεσης των προγραμμάτων πρακτόρων διευκολύνεται η ανάπτυξη περισσότερο κλιμακούμενων εφαρμογών.
- *Ανασχεδιασμό του συστήματος και μετά την κατασκευή του δικτύου:* κατά το σχεδιασμό μιας παραδοσιακής αρχιτεκτονικής πελάτη / εξυπηρετητή, η αρχιτεκτονική ορίζει αυστηρά κατά το σχεδιασμό του συστήματος τους ρόλους του πελάτη και του εξυπηρετητή, παίρνοντας εξαρχής αποφάσεις σχετικά με τους περιορισμούς εύρους ζώνης, την κίνηση του

δικτύου, τον αριθμό πελατών και εξυπηρετητών κλπ. Αν οι αρχικές προδιαγραφές είναι λανθασμένες, η εφαρμογή δεν θα ανταποκρίνεται σωστά. Η τεχνολογία κινητών πρακτόρων επιτρέπει να λαμβάνονται λιγότερες αποφάσεις κατά τη σχεδίαση του συστήματος, το οποίο πλέον μπορεί εύκολα να ανασχεδιασθεί [3].

3.1.3.5 Εφαρμογές των κινητών πρακτόρων

Όπως αναφέρθηκε, η τεχνολογία κινητών πρακτόρων προσφέρει σημαντικά πλεονεκτήματα σ' ένα εκτεταμένο εύρος εφαρμογών, οι οποίες απαιτούν κατανεμημένους υπολογισμούς και ασύγχρονη επικοινωνία. Ο κατανεμημένος χαρακτήρας των κινητών πρακτόρων μπορεί να οδηγήσει σε βελτιστοποιημένη χρήση υπολογιστικών πόρων, όπως για παράδειγμα σε προβλήματα Υπολογιστικού Ηλεκτρομαγνητισμού. Η δυνατότητα ασύγχρονης επικοινωνίας ελαττώνει την ανάγκη για επικοινωνία σε πραγματικό χρόνο για τον τελικό χρήστη. Η δυνατότητα αυτόνομης μεταφοράς και εκτέλεσης κώδικα σε διαφορετικά περιβάλλοντα συμβάλλει στην ελάττωση του κόστους εύρους του δικτύου και στην ανάπτυξη εφαρμογών με αυξημένη διαλειτουργικότητα.

Αρχικά, λόγω των χαρακτηριστικών της μεταφοράς, της αυτονομίας και της ασύγχρονης επικοινωνίας, οι κινητοί πράκτορες μπορούν να χρησιμοποιηθούν για να παρέχουν ανεκτικότητα σε λάθη δικτύου (network fault tolerance). Επίσης η προσαρμοστικότητα που παρουσιάζουν σε αλλαγές τόσο της κατάστασης του δικτύου όσο και του περιβάλλοντος του δικτύου, όπως είναι η κατάτμηση του δικτύου και οι αποσυνδεδεμένοι υπολογιστές, κάνει την τεχνολογία των κινητών πρακτόρων κατάλληλη για την προστασία κρίσιμων εφαρμογών από σφάλματα που προέρχονται από αναξιόπιστα δίκτυα.

Μια πολύ σημαντική εφαρμογή της τεχνολογίας κινητών πρακτόρων έγκειται στη συλλογή δεδομένων πάνω σ' ένα δίκτυο, π.χ. το διαδίκτυο. Για παράδειγμα, ένας κινητός πράκτορας μπορεί να σταλεί σε έναν εξυπηρετητή όπου υπάρχει μια μεγάλη βάση δεδομένων, να εκτελέσει τοπικά μια αναζήτηση και να επιστρέψει με το αποτέλεσμα στον υπολογιστή που ξεκίνησε η εκτέλεσή του.

Κινητοί πράκτορες μπορούν να χρησιμοποιηθούν για την αποδοτική ανανέωση απομακρυσμένων υπηρεσιών σε κατανεμημένες εφαρμογές. Για παράδειγμα, ένας κινητός πράκτορας μπορεί να σταλεί από ένα κεντρικό σημείο ελέγχου για την ανανέωση των λειτουργιών των κόμβων του δικτύου. Επίσης, οι κινητοί πράκτορες είναι κατάλληλοι για εφαρμογές που απαιτούν συμπαγή επικοινωνία (robustness) μέσα από μη αξιόπιστα δίκτυα και συνδέσεις χαμηλού εύρους ζώνης δικτύου. Παραδείγματα εφαρμογών τους περιλαμβάνουν μεταξύ άλλων, διαχείριση δικτύων, εφαρμογές ηλεκτρονικού εμπορίου και ηλεκτρονικές δημοπρασίες .

Συγκεκριμένα σε ό,τι αφορά το ηλεκτρονικό εμπόριο, ανοίγονται νέες προοπτικές. Για παράδειγμα, κάποιος πράκτορας μπορεί να αναλαμβάνει να ενημερώνει διαφορετικές ιστοσελίδες για τα νέα προϊόντα, χωρίς να είναι απαραίτητη η ύπαρξη κοινής βάσης δεδομένων ή άλλης υποδομής επικοινωνίας ανάμεσα στις ιστοσελίδες. Από την πλευρά του χρήστη, κάποιος πράκτορας μπορεί να αναλαμβάνει να αναζητήσει προϊόντα τα οποία ταιριάζουν με κάποια κριτήρια αναζήτησης, και να του παρουσιάσει τα αποτελέσματα, πραγματοποιώντας και κάποια σύγκριση μεταξύ τους. Οι πράκτορες που θα χρησιμοποιηθούν σε τέτοιου είδους εφαρμογές πρέπει να είναι αξιόπιστοι, τόσο από άποψη ασφάλειας, όσο και ανοχής σε σφάλματα. Επιπλέον, η ίδια η τεχνολογία των κινητών πρακτόρων οδηγεί σε ένα νέο είδος ηλεκτρονικού εμπορίου, το λεγόμενο κινητό εμπόριο (mobile commerce, m – commerce).

Αποδοτική εφαρμογή των κινητών πρακτόρων μπορεί να γίνει και στην περίπτωση των κινητών χρηστών ενός δικτύου. Αυτό που χαρακτηρίζει αυτήν την κατηγορία χρηστών είναι η δαπανηρή και ευαίσθητη σύνδεσή τους με το δίκτυο, καθώς και ο κατά κανόνα ελλιπής σε πόρους υπολογιστής που χρησιμοποιούν για τη σύνδεση. Για την υποστήριξη αυτών των κινητών χρηστών το παράδειγμα του κινητού πράκτορα μπορεί να χρησιμοποιηθεί, αφού ταιριάζει ιδανικά σε περιπτώσεις όπου δεν είναι δυνατή η συνεχόμενη και σταθερή σύνδεση.

Προς αυτήν την κατεύθυνση έχουν γίνει προσπάθειες για την εκμετάλλευση του μοντέλου των κινητών πρακτόρων στην διαχείριση δικτύων κινητών συσκευών.

Πολύ σημαντικός είναι ο ρόλος των κινητών πρακτόρων και στην ταχεία ανάπτυξη των λεγόμενων Ευφυών Δικτύων(ΕΔ), των οποίων η αρχιτεκτονική γίνεται περισσότερο ισχυρή και δυναμική με τη χρήση νέων τεχνολογιών, όπως η πλατφόρμα κατανεμημένης επικοινωνίας CORBA (Common Object Request Broker) και οι τεχνολογίες κινητών πρακτόρων. Με τη χρήση της CORBA γίνεται καλύτερη εκμετάλλευση της κατανεμημένης φύσης των ΕΔ. Η χρήση κινητών πρακτόρων επιτρέπει μια εύρωστη και δυναμικά οργανωμένη εφαρμογή με μειωμένο κόστος επικοινωνίας.

Τέλος, σε κάποιες πρόσφατες ερευνητικές προσπάθειες έχει γίνει προσπάθεια για τη χρησιμοποίηση κινητών πρακτόρων στην ανάπτυξη πλεγματικών υποδομών [1].[2].[3].

3.1.3.6 Τελευταίες εξελίξεις

Παρά το γεγονός ότι η τεχνολογία κινητών πρακτόρων υπάρχει πολλά χρόνια, μέχρι σήμερα δεν χαίρει ευρείας αποδοχής από την κοινότητα της πληροφορικής, ενώ και η διάδοση της γίνεται με σχετικά αργούς ρυθμούς. Σε μεγάλο βαθμό αυτό οφείλεται στην απουσία βασικών θεμάτων από τη συγκεκριμένη τεχνολογία, ώστε να μπορεί αυτή να θεωρηθεί εκμεταλλεύσιμη και εφαρμόσιμη. Ακόμα και σήμερα διεξάγονται μεγάλες ερευνητικές προσπάθειες για την ολοκλήρωση και διάδοση της τεχνολογίας των κινητών πρακτόρων.

Ένα από τα πιο κρίσιμα εμπόδια στην ευρεία διάδοση των κινητών πρακτόρων θεωρείται η ασφάλεια. Τα τελευταία χρόνια μεγάλο μέρος της έρευνας έχει αφιερωθεί στο συγκεκριμένο αντικείμενο.

Επίσης, η έλλειψη επαρκούς υποστήριξης για την ανάπτυξη εφαρμογών αποτελεί τροχοπέδη στην εξάπλωση της χρήσης της τεχνολογίας κινητών πρακτόρων. Γι΄ αυτό το λόγο γίνεται μια μεταφορά της έρευνας από το πεδίο της κινητικότητας και γενικά των θεμάτων που αφορούν τη λειτουργία των κινητών πρακτόρων σε θέματα που σχετίζονται με την απλοποίηση και την υποστήριξη χρήσης των κινητών πρακτόρων για υλοποίηση πραγματικών εφαρμογών. Κάποιες προσπάθειες έχουν επικεντρωθεί στο να γίνουν οι εφαρμογές που θα χρησιμοποιούν τους πράκτορες το επίκεντρο της σχεδίασης και όχι οι ίδιοι οι πράκτορες. Με αυτήν την προσέγγιση η εφαρμογή και οι πράκτορες είναι δυνατόν να αλληλεπιδρούν απευθείας.

Σοβαρές προσπάθειες γίνονται και για τη δημιουργία νέων μοντέλων και γλωσσών προδιαγραφών για κινητούς πράκτορες, ώστε να αντιμετωπιστούν προβλήματα σχετικά με την ασφάλεια και την πολύπλοκη συμπεριφορά των πρακτόρων (όπως π.χ. τη μετανάστευση). Απαραίτητη είναι η ύπαρξη συγκεκριμένων χαρακτηριστικών στα προαναφερθέντα μοντέλα και γλώσσες, όπως παραμετροποίηση (customizability) και επαρκής περιγραφική δύναμη. Μια λύση για τα παραπάνω προβλήματα που διαθέτει αυτά τα χαρακτηριστικά είναι η FDT (Formal Description Technique).

Παρά την συνεχή αύξηση που παρατηρείται στην ταχύτητα των CPU, οι σύγχρονοι υπολογιστές δεν μπορούν να καλύψουν τις τεράστιες ανάγκες που υπάρχουν σε τομείς όπως η βιοπληροφορική. Τα υπολογιστικά μοντέλα που υπάρχουν σήμερα σε όλους τους τομείς της επιστήμης παρουσιάζουν μια εκθετική χρονική πολυπλοκότητα, ενώ και τα δεδομένα προς επεξεργασία αυξάνονται ραγδαία. Γι' αυτούς τους λόγους η ζήτηση για υπολογιστική ισχύ υπερσκελίζει την όποια αύξησή της. Ενώ όμως συμβαίνει αυτό, τεράστια ποσά υπολογιστικής ισχύος υπάρχουν αχρησιμοποίητα σε υπολογιστές σε όλο τον κόσμο. Ένας τρόπος για να γίνει εκμετάλλευση αυτής της αχρησιμοποίητης υπολογιστικής ισχύος είναι η σύνδεση των υπολογιστών αυτών σε ένα κατανεμημένο δίκτυο Peer-to-Peer . Στο P2P Μοντέλο κάθε κόμβος μπορεί είτε να ξεκινήσει επικοινωνία, είτε να δεχτεί αιτήσεις και να διαθέσει υπηρεσίες, χωρίς να γίνεται κάποια διάκριση ρόλων. Πλέον δεν είναι υπεύθυνος για τις εφαρμογές κάποιος εξυπηρετητής, αλλά όλοι οι κόμβοι του δικτύου, οπότε μιλάμε για ένα πλήρως κατανεμημένο σύστημα.

Το ιδανικό εργαλείο για την υλοποίηση κατανεμημένων υπηρεσιών, όπως είναι αυτές που προσφέρει το μοντέλο P2P, είναι οι κινητοί πράκτορες, αφού μπορούν να αντιδρούν με βάση

τη διαθεσιμότητα πηγών ενώ μπορούν να εκτελούν διάφορες εργασίες χωρίς να χρειάζεται να υπάρχει εγκατεστημένο κάθε φορά κάποιο συγκεκριμένο για την εργασία αυτή λογισμικό.

Τρία συστήματα που κάνουν χρήση κινητών πρακτόρων πάνω σε δίκτυα P2P είναι το POPCORN , το οποίο προορίζεται για εμπορικές συναλλαγές, το YACA, που προορίζεται για κατανεμημένη επίλυση υπολογιστικά απαιτητικών προβλημάτων και το PaCMan (Parallel Computing with Java Mobile Agents**)**. Το σημαντικότερο ίσως σύστημα κινητών πρακτόρων που χρησιμοποιεί την peer-to-peer αρχιτεκτονική επικοινωνίας είναι το JADE (Java Agent Development Framework) της TILAB. Είναι βασισμένο στη Java και δεν απαιτεί κάποια συγκεκριμένη έκδοση για να εκτελεστεί, ενώ επιπλέον συγκεντρώνει τα περισσότερα από τα χαρακτηριστικά που προαναφέρθηκαν: διαλειτουργικότητα, ομοιομορφία, ευκολία στη χρήση καθώς και μια φιλοσοφία pay-as-you-go, που επιτρέπει στον προγραμματιστή να χρησιμοποιεί μόνο τα χαρακτηριστικά που χρειάζεται.

Τέλος, οι συνεχείς εξελίξεις στα ασύρματα δίκτυα (Wavelan, Bluetooth) καθώς και η ευρεία διάδοση κινητών υπολογιστών διαφόρων ειδών (όπως φορητοί υπολογιστές, κινητά τηλέφωνα, PDAs), καθιστούν αναγκαία την εφαρμογή της συγκεκριμένης τεχνολογίας σε κινητούς υπολογιστές, κατά την οποία παρουσιάζονται πολυσύνθετα προβλήματα που σχετίζονται με το εξαιρετικά ανομοιογενές και συνεχώς μεταβαλλόμενο περιβάλλον ως προς το λογισμικό, το υλικό και τα επίπεδα δικτύου, αφού οι συσκευές που εμπλέκονται περιέχουν εφαρμογές με μεγάλες διαφορές στο υλικό και τα συστήματα μεσισμικού και μπορεί να χρησιμοποιούν διαφορετικές δικτυακές υποδομές[3].

3.2 Ασφάλεια κινητών πρακτόρων

3.2.1 Εισαγωγή

Η ασφάλεια είναι ένα από τα βασικότερα και ανοιχτά ακόμα θέματα σχετικά με τους κινητούς πράκτορες. Κι αυτό γιατί οι ίδιοι «παραβιάζουν» μια σειρά υποθέσεων, στις οποίες βασίζονται, κατά ένα μεγάλο μέρος, τα υπάρχοντα μέτρα ασφαλείας των υπολογιστικών συστημάτων, όπως:

-τα διάφορα προγράμματα ενεργούν εκ μέρους ενός προσώπου που μπορεί εύκολα να προσδιοριστεί και που σκόπιμα προβαίνει στην εκτέλεσή τους

-προέρχονται από εύκολα προσδιορίσιμες και γενικά έμπιστες πηγές (έτσι ώστε οι Δούρειοι Ίπποι (Trojan horses), προγράμματα που επιτίθενται σε ένα σύστημα ενεργώντας αντίθετα από τις προθέσεις του χρήστη, να είναι σπάνιοι)

-οι απειλές ασφάλειας προέρχονται από επιτιθέμενους που τρέχουν κακόβουλο λογισμικό, οπότε τα μέτρα ασφαλείας θα πρέπει να προσανατολίζονται στην αυθεντικότητα-ταυτοποίηση του χρήστη, ώστε να εξασφαλίζουν ότι τα προγράμματα που τρέχει ενεργούν μόνο όπως του έχει επιτραπεί

-τέλος, τα προγράμματα δεν μετακινούνται από σύστημα σε σύστημα, παρά μόνο αν οι χρήστες σκόπιμα τα μεταφέρουν.

Ιδίως η τελευταία υπόθεση είναι έντονα αμφισβητήσιμη στα σημερινά υπολογιστικά περιβάλλοντα. Χωρίς αυτό να δηλώνεται πάντα ρητά, τα σημερινά συστήματα ενσωματώνουν πολλά χαρακτηριστικά κινητικότητας: e-mails μπορεί να περιέχουν κώδικα που εκτελείται αυτόματα όταν ανοίξουν, ιστοσελίδες μπορεί να περιέχουν Java applets που εκτελούνται κατά την προβολή της σελίδας, προγράμματα άγνωστης προέλευσης εκτελούνται από χρήστες ανεξέλεγκτα κλπ. Ήδη, λοιπόν, έχουν αρχίσει να προκύπτουν σχετικά ζητήματα ασφάλειας, τα οποία ενισχύονται ακόμα περισσότερο όταν πρόκειται για πλήρη συστήματα κινητών πρακτόρων. Διάφορες πλευρές του θέματος έχουν αναγνωριστεί και μελετηθεί στη βιβλιογραφία[4]. Οι απειλές ασφάλειας μπορούν να ομαδοποιηθούν σε τρεις κατηγορίες: απειλές λόγω μη ασφαλούς δικτύωσης κατά τη διάρκεια της μετανάστευσης, απειλές από κακόβουλους πράκτορες και απειλές από κακόβουλους υπολογιστές εκτέλεσης.

3.2.2 Απειλές ασφάλειας

Μη ασφαλή δίκτυα-Απειλές κατά τη μεταφορά

Τα θέματα ασφάλειας δικτύων αφορούν προφανώς και τους κινητούς πράκτορες, αφού πάνω από αυτά αναπτύσσονται. Τόσο η μεταφορά τους από έναν υπολογιστή επεξεργασίας σε άλλον όσο και η μεταξύ τους επικοινωνία πρέπει να προστατεύονται. Σε περίπτωση που δεν εξασφαλίζεται η ακεραιότητα και η εμπιστευτικότητά τους, είναι ευάλωτοι σε επιθέσεις παρακολούθησης και τροποποίησης από τρίτους (man-in-the-middle attacks) [4].

Κακόβουλοι πράκτορες

Οι κινητοί πράκτορες μεταφέρουν δεδομένα και εκτελέσιμο κώδικα. Συνεπώς κακόβουλοι πράκτορες μπορεί να χρησιμοποιηθούν για να προκαλέσουν διάφορες επιθέσεις στους υπολογιστές που εκτελούνται, π.χ. άρνηση εξυπηρέτησης, μη εξουσιοδοτημένη προσπέλαση/ τροποποίηση δεδομένων και μεταφορά ιών και Δούρειων Ίππων. Αναλυτικότερα, οι επιθέσεις αυτές περιλαμβάνουν:

- *Πλαστοπροσωπία (impersonation)* και *μεταμφίεση (masquerading)*. Σε περίπτωση έλλειψης μεθόδων αυθεντικοποίησης, κακόβουλοι πράκτορες μπορούν να προσποιηθούν ότι προέρχονται από έμπιστες πηγές, ώστε να αποκτήσουν πρόσβαση στο περιβάλλον εκτέλεσης. Επίσης, σε περίπτωση έλλειψης μεθόδων ελέγχου προσπέλασης, κακόβουλοι πράκτορες μπορεί να εκτελεστούν με περισσότερα από τα αναμενόμενα δικαιώματα, ώστε να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση σε εμπιστευτικά δεδομένα, προγράμματα ή άλλους πόρους του συστήματος, ή να προκαλέσουν μη εξουσιοδοτημένη τροποποίησή τους.
- *Άρνηση εξυπηρέτησης (denial of service)*. Τέτοιες επιθέσεις μπορούν να προκληθούν με διάφορους τρόπους, όπως για παράδειγμα με τη δέσμευση μεγάλου αριθμού υπολογιστικών πόρων. Ένας κακόβουλος πράκτορας μπορεί να εκκινήσει μεγάλο αριθμό TCP/IP συνδέσεων με άλλους απομακρυσμένους υπολογιστές, ώστε να παρεμποδίσει τη δημιουργία άλλων απομακρυσμένων συνδέσεων. Το ίδιο αποτέλεσμα μπορεί να προκληθεί από κακόβουλους πράκτορες που καταναλώνουν μεγάλη ποσότητα της μνήμης του υπολογιστή στον οποίο εκτελούνται, ή από επιθέσεις μέσω ιών.
- *Παρακολούθηση (eavesdropping)*. Κακόβουλοι πράκτορες ενεργώντας ως Δούρειοι Ίπποι, μπορεί να λειτουργήσουν ως παθητικοί ωτακουστές, είτε παρακολουθώντας την επικοινωνία του υπολογιστή εκτέλεσης με άλλους υπολογιστές και προγράμματα, είτε προσπαθώντας να προσπελάσουν εμπιστευτική πληροφορία, αποθηκευμένη στον υπολογιστή εκτέλεσης. Επίσης μπορεί να λειτουργήσουν ως ενεργοί ωτακουστές, προσπαθώντας να αποστείλουν αυτή την πληροφορία σε έναν απομακρυσμένο προορισμό [1].

Κακόβουλοι υπολογιστές εκτέλεσης

Σαν πρόβλημα είναι ακόμα δυσκολότερο στην επίλυσή του από αυτό των κακόβουλων πρακτόρων. Κι αυτό γιατί ο πράκτορας βρίσκεται κάτω από τον πλήρη έλεγχο του διακομιστή, ένας κακόβουλος διακομιστής συνεπώς μπορεί να κάνει πρακτικά οτιδήποτε. Κάποιες από τις πιθανές απειλές είναι:

- *Πλαστοπροσωπία* (impersonation) και *μεταμφίεση* (masquerading). Σε περίπτωση έλλειψης μεθόδων αυθεντικοποίησης, πριν τη μετανάστευση ενός πράκτορα σε έναν υπολογιστή, ένας εχθρικός υπολογιστής μπορεί να προσποιηθεί ότι αποτελεί κάποιον έμπιστο προορισμό.
- *Άρνηση εξυπηρέτησης* (denial-of-service). Είναι φανερό ότι ένας υπολογιστής εκτέλεσης μπορεί να αρνηθεί την παροχή πόρων για την εκτέλεση ενός πράκτορα ή να αρνηθεί τη μετανάστευση ενός πράκτορα σε κάποιο άλλο υπολογιστή και να τερματίσει την εκτέλεσή του.
- *Παρακολούθηση* (eavesdropping). Ο υπολογιστής εκτέλεσης έχει πρόσβαση στον κώδικα, τα δεδομένα και τη ροή εκτέλεσης ενός πράκτορα. Συνεπώς μπορεί να προσπελάσει οποιαδήποτε πληροφορία μεταφέρει ο πράκτορας, τουλάχιστον κατά τη στιγμή που θα πρέπει να χρησιμοποιήσει την πληροφορία αυτή και ο ίδιος ο πράκτορας.

- *Τροποποίηση κώδικα* (code-tampering). Ο υπολογιστής που εκτελεί έναν πράκτορα μπορεί να τροποποιήσει τον κώδικά του, για παράδειγμα εισάγοντας ιούς ή άλλου είδους επιβλαβή κώδικα. Επίσης, μπορεί να αλλάξει τμήματα του εκτελέσιμου κώδικα ή των δεδομένων του πράκτορα με άλλα της επιλογής του.
- *Αυθαίρετος καταλογισμός ευθύνης* (arbitrary non-repudiation). Ο υπολογιστής εκτέλεσης ενός πράκτορα μπορεί να υποκλέψει το μυστικό κλειδί υπογραφής ενός κινητού πράκτορα τη στιγμή που το χρησιμοποιεί ο ίδιος ο πράκτορας για κάποιο υπολογισμό και να το χρησιμοποιήσει για να δεσμεύσει τον πράκτορα (ή τον αποστολέα του) σε αυθαίρετες συναλλαγές της επιλογής του [1].

3.2.3 Υπάρχουσες λύσεις

3.2.3.1 Προστασία κατά τη μετακίνηση

Οι δίαυλοι επικοινωνίας θα πρέπει να είναι κρυπτογραφικά ασφαλισμένοι, μέσω αυθεντικότητας και εμπιστευτικότητας οντοτήτων και δεδομένων. Κάποιοι μηχανισμοί γι αυτό το σκοπό είναι οι SSL(Secure Socket Layer)/TLS(Transport Layer Security) στο στρώμα μεταφοράς, ή IPsec(IP security) στο στρώμα δικτύου και μπορούν είτε να παρέχονται από την υποκείμενη πλατφόρμα είτε από τους ίδιους τους πράκτορες.

Σε αυτά τα πλαίσια η προστασία των κρυπτογραφικών κλειδιών των πρακτόρων από κακόβουλους διακομιστές και αντίστροφα είναι ιδιαίτερα σημαντική. Η αυθεντικοποίηση πρακτόρων και διακομιστών δε σχετίζεται μόνο με την προστασία της επικοινωνίας αλλά και με το θέμα των κακόβουλων πρακτόρων/ διακομιστών. Είναι το πρώτο βήμα για τον καθορισμό του αν ένας πράκτορας/ διακομιστής είναι αξιόπιστος και, ανάλογα, αν ο πράκτορας μπορεί με ασφάλεια να μεταναστεύσει στο διακομιστή ή ο διακομιστής να εκτελέσει τον πράκτορα [4].

3.2.3.2 Προστασία από κακόβουλους πράκτορες

Η προστασία του υπολογιστή από κακόβουλους πράκτορες σε ό,τι αφορά την αυθεντικοποίηση είναι δυνατή κατ' αρχήν με τη βοήθεια sandbox μηχανισμών (π.χ. τα sandbox στοιχεία ασφάλειας της γλώσσας Java). Πιο ισχυρή ακόμα προστασία μπορεί να αναπτυχθεί πάνω από ένα περιβάλλον java. Οι τεχνικές ασφαλούς προγραμματισμού (language safety), δηλαδή γλώσσες προγραμματισμού που παρέχουν ασφαλή εκτέλεση, μπορούν επίσης να παίξουν σημαντικό ρόλο, όπως π.χ. η γλώσσα Safe-Tcl. Εξάλλου πολιτικές ασφάλειας μπορούν να βασιστούν και σε γλώσσες εξουσιοδότησης (authorization languages), καθορίζοντας ποιος πράκτορας μπορεί να κάνει τι, όπως επίσης και στον έλεγχο χρήσης πόρων (resource usage control) ή στην ανάθεση (allocation).

Πέρα από την αυθεντικοποίηση, ο πράκτορας μπορεί να φέρει μια ένδειξη, βάση της οποίας ο διακομιστής θα αποφασίζει κατά πόσο ο κώδικας είναι ασφαλής για εγκατάσταση και εκτέλεση. Από την άλλη, όχι μόνο ο κώδικας αλλά και η κατάσταση του πράκτορα μπορεί να χρησιμεύσει στην ασφάλεια. Με την εκτίμηση κατάστασης (state appraisal), το σύνολο των προνομίων ενός πράκτορα υπολογίζονται με βάση την κατάστασή του. Επίσης, όχι μόνο μεμονωμένοι διακομιστές αλλά και ομάδες διακομιστών απαιτούν προστασία από κακόβουλους πράκτορες. Έτσι αντιμετωπίζονται επιθέσεις που δεν εντοπίζονται από μεμονωμένους hosts (π.χ. κατανάλωση πόρων) [4].

Ιδιαίτερη αναφορά θα πρέπει να γίνει στους μηχανισμούς ελέγχου πρόσβασης (access control), που είναι πολύ σημαντικοί για την προστασία κάθε κατανεμημένης εφαρμογής.

Η *πολιτική πρόσβασης* καθορίζει κανόνες για την αυθεντικοποίηση (authentication) και την εξουσιοδότηση (authorization) οντοτήτων που ζητούν πρόσβαση σε συγκεκριμένους πόρους ενός συστήματος. Στην περίπτωση συστημάτων κινητών πρακτόρων η πολιτική πρόσβασης καθορίζει κανόνες για την αυθεντικοποίηση των κινητών πρακτόρων που ζητούν πρόσβαση σε κάποιο διακομιστή. Επίσης, περιλαμβάνει κανόνες που προσδιορίζουν την εξουσιοδότηση που έχει κάποιος πράκτορας για κατανάλωση συγκεκριμένων πόρων, πρόσβαση σε συγκεκριμένα αρχεία ή εφαρμογές και άλλα δικαιώματα, όπως εξουσιοδότηση

για την εκκίνηση σύνδεσης σε κάποιο απομακρυσμένο υπολογιστή. Πολλά προτεινόμενα τέτοια σχήματα υπάρχουν στη διεθνή βιβλιογραφία.

Το επικρατέστερο σχετικό μοντέλο είναι το μοντέλο Ελέγχου Πρόσβασης Βασισμένου σε Ρόλους (Role Based Access Control-RBAC) και βασίζεται σε τρία σύνολα σημασιολογικών δομών, στους *χρήστες,* τους *ρόλους* και τα *προνόμια (*ή *δικαιώματα πρόσβασης)*. Υπάρχουν επίσης οι έννοιες της *συνόδου* (session), των *περιορισμών* (constraints), του *διαχωρισμού καθηκόντων* (separation of duties) και της ιδιαίτερα σημαντικής *μεταβίβασης ρόλου* (role delegation). Σημαντικό χαρακτηριστικό του μοντέλου είναι και ο δυναμικός του χαρακτήρας. Για περισσότερα στοιχεία σχετικά με τα παραπάνω βλέπε και [1].

Τα κυριότερα σχήματα ελέγχου πρόσβασης για συστήματα κινητών πρακτόρων που προτείνονται στη βιβλιογραφία είναι: το σχήμα ασφάλειας των Karjoth et al, το σχήμα αυθεντικοποίησης και εξουσιοδότησης των Berkovits et al και το σχήμα διαχείρισης δικαιωμάτων πρόσβασης του Jansen. Για μια ουσιαστική σύγκριση των παραπάνω σχημάτων, βλέπε [1].

Εκτός από την εφαρμογή RBAC πολιτικών πρόσβασης σε συστήματα πρακτόρων, οι κινητοί πράκτορες έχουν χρησιμοποιηθεί ως υποστηρικτική τεχνολογία για την υλοποίηση πολιτικών RBAC σε κατανεμημένες εφαρμογές. Αρκετά μοντέλα προτείνουν οι Demurjian et al, ενώ οι Fayad et al πρότειναν την έννοια των *κινητών πολιτικών* (mobile policies) [1].

3.2.3.3 Προστασία από κακόβουλους υπολογιστές εκτέλεσης

Η προστασία του πράκτορα από εχθρικούς υπολογιστές εκτέλεσης αποτελεί ένα πολύ δυσκολότερο και ανοιχτό ερευνητικά πρόβλημα. Κατά τη διάρκεια της εκτέλεσης ενός κινητού πράκτορα, ο τελευταίος βρίσκεται σε μια πολύ ασύμμετρη σχέση με τον υπολογιστή, αφού ο υπολογιστής έχει πρόσβαση στα δεδομένα, τον κώδικα και την κατάσταση εκτέλεσής του, με τα δύο τελευταία να σχετίζονται στενά με την εκτέλεση του ίδιου του πράκτορα στο εκάστοτε περιβάλλον. Χρειαζόμαστε, λοιπόν, μηχανισμούς που να προστατεύουν αφενός το τμήμα των δεδομένων και αφετέρου την ακεραιότητα και το απόρρητο της εκτέλεσης. Οι ερευνητικές προσπάθειες για την αντιμετώπιση αυτών των προβλημάτων διαχωρίζονται σε δύο κατηγορίες: ανίχνευση και παρεμπόδιση επιθέσεων κατά του πράκτορα.

3.2.3.3.1 Ανίχνευση επιθέσεων κατά του πράκτορα

Περιλαμβάνει λύσεις που στοχεύουν στην εκ των υστέρων (a posteriori) ανίχνευση της ταυτότητας και στην απόδειξη της συμπεριφοράς ενός κακόβουλου υπολογιστή εκτέλεσης.

Σε ότι αφορά την προστασία των δεδομένων ενός πράκτορα, υπάρχουν διάφοροι τρόποι ανίχνευσης μιας παραποίησής τους. Για παράδειγμα, μπορούν να εισαχθούν «ψεύτικα» δεδομένα, τα οποία κανονικά (από έναν νομιμοποιημένο διακομιστή) δεν είναι προς τροποποίηση, οπότε αν έχουν τροποποιηθεί σημαίνει ότι τα δεδομένα έχουν «πειραχτεί».

Σχετικά με την ακεραιότητα της εκτέλεσης ιδιαίτερα σημαντική είναι, όπως είπαμε και παραπάνω, η προστασία του κώδικα και της κατάστασης του πράκτορα, τα οποία μπορούν για το σκοπό αυτό να υπογραφούν ψηφιακά και μετά να κρυπτογραφηθούν με το δημόσιο κλειδί του παραλήπτη, ώστε ο τελευταίος να λαμβάνει σίγουρα το σωστό κώδικα και κατάσταση. Με τις προτάσεις που ακολουθούν, θα είναι δυνατή και η διαπίστωση της σωστής ή όχι εκτέλεσης του πράκτορα από τον παραλήπτη.

Με την εκτίμηση κατάστασης (state appraisal) καθορίζονται σταθερές που πρέπει να ικανοποιούνται από την κατάσταση ενός πράκτορα. Ο Vigna [5] πρότεινε ένα μηχανισμό ανίχνευσης, ο οποίος καταγράφει την εκτέλεση ενός πράκτορα και την αλληλεπίδρασή του με το περιβάλλον εκτέλεσης. Ο μηχανισμός αυτός μπορεί να χρησιμοποιηθεί αργότερα από το δημιουργό του πράκτορα προκειμένου να διαπιστωθεί η ορθότητα της εκτέλεσης και να εντοπιστούν πιθανοί κακόβουλοι υπολογιστές. Ο μηχανισμός αυτός όμως απαιτεί συνεχή σύνδεση του αποστολέα στο δίκτυο και αλληλεπιδραστική επικοινωνία του με τον πράκτορα ή και επανεκτέλεσή του από τον ίδιο, το οποίο συνεπάγεται μεγάλο κόστος επικοινωνίας.

Συντομότερες αποδείξεις ορθότητας (proofs of correctness), βασισμένες σε ολογραφικές αποδείξεις προτάθηκαν από τον Yee [6], σύμφωνα με τις οποίες ελέγχονται μόνο κάποια τυχαία επιλεγμένα bits. Οι Biehl *et al.* [7] εφαρμόζουν προς βελτίωση του παραπάνω

σχήματος τεχνικές απόρρητης ανάκτησης πληροφοριών (private information retrieval), ώστε να μην αποκαλύπτεται στον υπολογιστή εκτέλεσης ποια bits ελέγχονται.

Για την αποφυγή της on-line σύνδεσης του αποστολέα και της αλληλεπίδρασής του με τον πράκτορα, οι Yi et al. [8] πρότειναν τη χρήση ενός Κέντρου Υπηρεσιών Πρακτόρων (Agent Service Center). Το κέντρο αυτό λειτουργεί ως έμπιστη οντότητα και είναι υπεύθυνο για την αποστολή πρακτόρων σε ένα δίκτυο εκ μέρους διαφόρων χρηστών. Το κέντρο παρακολουθεί το δρομολόγιο κάθε πράκτορα και συλλέγει τις απαραίτητες πληροφορίες για την αποκάλυψη κακόβουλων υπολογιστών εκτέλεσης.

Μία διαφορετική προσέγγιση που προκύπτει από την ανάγκη για ανοχή σε λάθη, είναι η αντιγραφή πρακτόρων. Πολλαπλοί πράκτορες με ίδια λειτουργία αποστέλλονται σε διαφορετικούς διακομιστές, οπότε εκείνοι που παραποιούνται σημαίνει ότι πέρασαν από κακόβουλους [9]. Ένα πολυπρακτορικό σύστημα παρουσιάζεται επίσης και στο [1], το οποίο χρησιμοποιεί έναν στατικό και πολλούς κινητούς πράκτορες για κάθε χρήστη. Οι κινητοί πράκτορες μπορεί να εκτελούνται ασύγχρονα σε διαφορετικούς υπολογιστές, ενώ ο στατικός πράκτορας μπορεί να εντοπίσει πιθανούς κακόβουλους υπολογιστές. Το σύστημα αυτό δεν απαιτεί on-line σύνδεση του αποστολέα κατά τη διάρκεια εκτέλεσης των πρακτόρων ή της ύπαρξης έμπιστων κέντρων.

Τέλος, ο Hohl [10] γενικεύει αυτές τις λύσεις χρησιμοποιώντας την έννοια των καταστάσεων αναφοράς (reference states).

Τα συστήματα αυτά παρέχουν ικανοποιητικές λύσεις για συγκεκριμένα προβλήματα. Παρόλα αυτά, υπάρχουν περιπτώσεις όπου η εκ των υστέρων ανίχνευση δεν παρέχει αρκετή προστασία. Για παράδειγμα, σε εφαρμογές όπου ένας κινητός πράκτορας μπορεί να μεταφέρει ηλεκτρονικά νομίσματα ή/ και μυστικά κλειδιά του αποστολέα του, η ανίχνευση δεν είναι αρκετή. Σε αυτές τις περιπτώσεις το χρονικό διάστημα μεταξύ της επίθεσης κατά του πράκτορα και της ανίχνευσης της επίθεσης μπορεί να είναι πολύ κρίσιμο για την ασφάλεια του αποστολέα του πράκτορα [4].

3.2.3.3.2 Παρεμπόδιση επιθέσεων κατά του πράκτορα

Η φιλοσοφία αυτής της μεθόδου είναι η εκ των προτέρων (a priori) παρεμπόδιση επιθέσεων του περιβάλλοντος εκτέλεσης κατά του πράκτορα. Διακρίνονται δύο περιπτώσεις: παθητική και ενεργητική παρεμπόδιση.

Οι μηχανισμοί παθητικής παρεμπόδισης προστατεύουν τους κινητούς πράκτορες χρησιμοποιώντας οργανωτικές ή αρχιτεκτονικές λύσεις. Οι Farmer et al. πρότειναν ένα σχήμα στο οποίο οι πράκτορες κυκλοφορούν μόνο μέσα σε έμπιστο περιβάλλον εκτέλεσης. Οι Merwe και Sholms παρουσίασαν ένα σύστημα για ηλεκτρονικές αγορές μέσω πρακτόρων, όπου οι πράκτορες υλοποιούνται ως κατανεμημένα αντικείμενα που επικοινωνούν απομακρυσμένα. Μερικοί μηχανισμοί ανίχνευσης επίσης χρησιμοποιούν παθητικούς μηχανισμούς παρεμπόδισης. Αυτές οι προσεγγίσεις είτε κάνουν ισχυρές υποθέσεις για την εμπιστοσύνη του περιβάλλοντος εκτέλεσης, είτε κάνουν συμβιβασμούς σε πλεονεκτήματα της τεχνολογίας των κινητών πρακτόρων, όπως είναι η αυτονομία και η μετανάστευση [1].

Οι Rasmusson and Jansson υποστήριξαν ότι η επιλογή των διακομιστών βάση της «φήμης» τους στο δίκτυο συνιστά μια πιο ευέλικτη λύση, προϋποθέτει βέβαια την αναγνώρισή τους με βάση μηχανισμούς αυθεντικοποίησης. Και στις δύο περιπτώσεις πάντως προκύπτει ένα κλειστό σύνολο από έμπιστες πλατφόρμες εκτέλεσης, ενώ για την ανάγκη επικοινωνίας πρακτόρων χωρίς κοινό έμπιστο διακομιστή έχει εισαχθεί και ο όρος του εικονικού έμπιστου τρίτου (virtual trusted third party) [4].

Η ενεργητική παρεμπόδιση επικεντρώνεται στην ανάπτυξη λύσεων που παρέχουν προστασία σε κινητούς πράκτορες από επιθέσεις του περιβάλλοντος εκτέλεσης, χωρίς να συμβιβάζουν τα πρακτικά πλεονεκτήματα του υποδείγματος των κινητών πρακτόρων. Μια κατηγορία τέτοιων λύσεων περιλαμβάνει τη χρήση ασφαλούς υλικού (tamper-resistant hardware). Παραδείγματα έμπιστου υλικού είναι οι ασφαλείς συν-επεξεργαστές (secure co-processors) και οι έξυπνες κάρτες (smart cards), ενώ οι Wilhelm *et al*. [11] παρουσιάζουν ένα σύστημα ασφαλούς περιβάλλοντος. Όμως η εφαρμογή τέτοιων λύσεων δεν είναι μεγάλη,

κυρίως λόγω του υψηλού κόστους που συνεπάγονται. Η χρήση ασφαλούς υλικού συνεπάγεται επίσης υποθέσεις για την εμπιστοσύνη του περιβάλλοντος εκτέλεσης, σε μικρότερο βέβαια βαθμό από ότι οι λύσεις της προηγούμενης κατηγορίας [4].

Η αναζήτηση λύσεων για ενεργητική παρεμπόδιση επιθέσεων εναντίον κινητών πρακτόρων που βασίζονται σε λογισμικό (software-based), είναι σχετικά πρόσφατο ερευνητικό πεδίο.

Σε σχέση με την προστασία των δεδομένων, οι Karjoth et al. ορίζουν ένα σύνολο απαιτήσεων ασφαλείας που θα πρέπει να ικανοποιούνται σε σχέση με επιθέσεις σε πράκτορες που φέρουν *αλυσίδα ενθυλακωμένων πληροφοριών* (chain of encapsulated information):

-*εμπιστευτικότητα δεδομένων* (data confidentiality), ώστε μόνο ο δημιουργός τους να έχει πρόσβαση στην πληροφορία

-*μη-δυνατότητα άρνησης* (non-repudiability) πληροφορίας από την πλευρά του διακομιστή

-*το προς τα εμπρός απόρρητο* (forward privacy), ώστε να μην αποκαλύπτονται οι ταυτότητες των προηγούμενων διακομιστών

-*ισχυρή προς τα εμπρός ακεραιότητα* (forward integrity), ώστε οι ήδη συγκεντρωμένες πληροφορίες να μην τροποποιούνται

-*δημόσια επαληθεύσιμη ακεραιότητα* (publicly verifiable integrity), ώστε οποιοσδήποτε να έχει τη δυνατότητα να επαληθεύει την πληροφορία

-αντίσταση στην χωρίς εξουσιοδότηση εισαγωγή δεδομένων

-αντίσταση στην περικοπή δεδομένων [1].

Διάφορες λύσεις έχουν προταθεί, που ικανοποιούν τις παραπάνω απαιτήσεις ή υποσύνολό τους και οι οποίες βασίζονται σε κοινές κρυπτογραφικές αρχές: οι διακομιστές υπογράφουν ψηφιακά τα δεδομένα που δίνουν στον agent και στη συνέχεια αυτά κρυπτογραφούνται με το δημόσιο κλειδί του δημιουργού του πράκτορα. Η ιδιότητα «αλυσίδας» που προαναφέραμε εξασφαλίζεται συμπεριλαμβάνοντας τα μέχρι πρότινος συγκεντρωμένα δεδομένα και την ταυτότητα του επόμενου διακομιστή στα συνολικά δεδομένα πριν την ψηφιακή τους υπογραφή. Σε αρκετά αντίστοιχα προτεινόμενα πρωτόκολλα πάντως έχουν εντοπιστεί ελαττώματα.

Σε ό,τι αφορά τώρα το απόρρητο της εκτέλεσης, τα *κλειδιά περιβάλλοντος* (environmental key generation) [12], κρυπτογραφικά κλειδιά, δηλαδή, που παράγονται με βάση δεδομένα περιβάλλοντος, συμβάλλουν, όχι ακριβώς στο απόρρητο, αλλά στην εκτέλεση του πράκτορα που κρυπτογραφούν μόνο υπό ορισμένες περιβαλλοντικές συνθήκες και όχι σε πιθανά κακόβουλους διακομιστές.

Μια άλλη προσέγγιση που προτάθηκε από τον Hohl[13] είναι η χρήση *τεχνικών «σύγχυσης» και ανασχηματισμού* (code obfuscation) του κώδικα και της ροής εκτέλεσης ενός πράκτορα. Σκοπός αυτής της μεθόδου είναι να κάνει όσο το δυνατόν δυσκολότερη και χρονοβόρα την παρακολούθηση και ανακάλυψη του πραγματικού αποτελέσματος της εκτέλεσης ενός πράκτορα. Η προσέγγιση αυτή μπορεί να είναι χρήσιμη για περιπτώσεις όπου ο χρόνος κατά τον οποίο απαιτείται προστασία του πράκτορα είναι σχετικά μικρός. Όμως το χρονικό διάστημα ασφάλειας που προσφέρεται δεν μπορεί να αποδειχθεί ή να προσδιοριστεί, αφού εξαρτάται από τις υπολογιστικές δυνατότητες του αντιπάλου. Αυτή η προσέγγιση μπορεί να θεωρηθεί ότι ανήκει στην κατηγορία που είναι γνωστή στην κρυπτογραφία ως «ασφάλεια μέσω ασάφειας» (security through obscurity).

Μια άλλη προσέγγιση για την προστασία κινητών πρακτόρων στηρίζεται στον Ασαφή Υπολογισμό Συναρτήσεων (Secure Function Evaluations ή Function hiding), μια κρυπτογραφική θεώρηση που αρχικά προτάθηκε από τους Goldreigh et al.. Σύμφωνα με αυτή τη θεώρηση, δύο οντότητες, η Alice (πράκτορας) και ο Bob (υπολογιστής) θέλουν να υπολογίσουν το αποτέλεσμα μιας συνάρτησης f για μια είσοδο x. Η Alice γνωρίζει τη συνάρτηση f ενώ ο Bob την είσοδο x. Οι δύο συμμετέχοντες θέλουν να υπολογίσουν το αποτέλεσμα $f(x)$ με τέτοιο τρόπο, ώστε κανείς από τους δύο να μην αποκτήσει περισσότερη πληροφορία από όση ήδη κατέχει, πέραν του αποτελέσματος $f(x)$ [4].

Μια ειδική περίπτωση Ασφαλούς Υπολογισμού Συναρτήσεων που προτάθηκε από τους Sander και Tschudin [14] για την προστασία των κινητών πρακτόρων, είναι ο *Υπολογισμός με Κρυπτογραφημένες Συναρτήσεις-ΥΚΣ* (Computing with Encrypted Functions-CEF). Σύμφωνα με αυτή τη μέθοδο, για την απόκρυψη μιας συνάρτησης s, η συνάρτηση αυτή

κρυπτογραφείται μέσω της σύνθεσής της με μία άλλη συνάρτηση *f*. Ο υπολογιστής εκτελεί την κρυπτογραφημένη συνάρτηση *s* ο *f* χωρίς να έχει πρόσβαση στην *s*. Η ασφάλεια της μεθόδου έγκειται στη δυσκολία «αποσύνθεσης» της κρυπτογραφημένης συνάρτησης στις αρχικές συναρτήσεις. Επειδή το πνεύμα της τεχνολογίας των κινητών πρακτόρων είναι η αυτονομία τους, οι Sander και Tschudin πρότειναν την υλοποίηση *μη-αλληλεπιδραστικών ΥΚΣ* (non-interactive CEF) για την προστασία των κινητών πρακτόρων. Οι ίδιοι ερευνητές παρατήρησαν ότι μη-αλληλεπιδραστικοί ΥΚΣ μπορούν να υλοποιηθούν με τη χρήση αλγεβρικά ομομορφικών κρυπτογραφικών συστημάτων δημόσιου κλειδιού. Δυστυχώς όμως μέχρι στιγμής δεν υπάρχουν αποδεδειγμένα ασφαλή κρυπτοσυστήματα που να είναι αλγεβρικά ομομορφικά (δηλαδή προσθετικά και πολλαπλασιαστικά ομομορφικά). Υπάρχοντα κρυπτοσυστήματα που είναι προσθετικά ομομορφικά και θεωρούνται ασφαλή (π.χ. τα κρυπτοσυστήματα των Naccache και Stern [15])δεν μπορούν να χρησιμοποιηθούν γιατί για τον έλεγχο της ορθότητας των αποτελεσμάτων οι παράμετροι εισόδου θα πρέπει να είναι εκθετικά μεγάλοι.

Οι Cachin et al [16] πρότειναν μια λύση για την προστασία των κινητών πρακτόρων που βασίζεται σε κρυπτογραφημένα κυκλώματα (encrypted circuits) και στη μέθοδο επιλήσμονος μεταφοράς (oblivious transfer). Αν και η μέθοδος αυτή είναι πολύ ενδιαφέρουσα αφού λύνει το πρόβλημα για την προστασία οποιασδήποτε συνάρτησης πολυωνυμικού χρόνου, έχει μόνο θεωρητική αξία αφού δεν είναι πρακτικά εφαρμόσιμη.

Οι Loureiro και Molva [17] πρότειναν μια μέθοδο που επιτρέπει σε έναν κινητό πράκτορα να υπολογίσει μια Boolean συνάρτηση, χωρίς να αποκαλύπτει την ίδια τη συνάρτηση στο περιβάλλον εκτέλεσής του. Η ασφάλεια της μεθόδου ανάγεται στην ασφάλεια του κρυπτογραφικού συστήματος δημόσιου κλειδιού της McEliece. Η μεθοδολογία αυτή αποδεικνύει ότι υπάρχουν περιπτώσεις όπου είναι δυνατόν για έναν κινητό πράκτορα να υπολογίζει συναρτήσεις χωρίς να τις αποκαλύπτει στο περιβάλλον εκτέλεσης. Πρέπει να σημειωθεί όμως ότι ο πράκτορας δεν μπορεί να χρησιμοποιήσει το αποτέλεσμα του υπολογισμού αυτού πριν επιστρέψει στον αρχικό αποστολέα του, αφού το αντίστοιχο μυστικό κλειδί του αποστολέα είναι απαραίτητο για την αποκρυπτογράφηση του αποτελέσματος [1].

Οι Sander και Tschudin [14] πρότειναν μια εφαρμογή της μεθόδου μη-αλληλεπιδραστικών ΥΚΣ για την προστασία συναρτήσεων υπογραφής για κινητούς πράκτορες, γνωστή και ως *μη-αποσπώμενες υπογραφές* (undetachable signatures). Με αυτές τις υπογραφές, ένας κινητός πράκτορας μπορεί με ασφάλεια να υπογράψει για μία συναλλαγή, χρησιμοποιώντας μια συνάρτηση υπογραφής του αποστολέα του σε κρυπτογραφημένη μορφή. Αν και ο πράκτορας εκτελείται σε κάποιον πιθανώς εχθρικό υπολογιστή, η συνάρτηση υπογραφής είναι κρυπτογραφημένη με τέτοιον τρόπο ώστε είναι υπολογιστικά αδύνατο για τον υπολογιστή να υποκλέψει τη συνάρτηση. Για την υλοποίηση των μη-αποσπώμενων υπογραφών, οι Sander και Tschudin πρότειναν τη χρήση ρητών συναρτήσεων, που όμως αποδείχθηκαν μη ασφαλείς. Ένα κρυπτογραφικό πρωτόκολλο ασφαλούς μη-αποσπώμενης υπογραφής παρουσιάζεται και στο [1]. Το πρωτόκολλο αυτό συνδυάζει το σχήμα υπογραφής RSA με εκθετικές συναρτήσεις. Η ασφάλεια του πρωτοκόλλου ανάγεται στην ασφάλεια της υπογραφής RSA στο μοντέλο random oracles υπό ορισμένες κρυπτογραφικές υποθέσεις.

Οι μη-αποσπώμενες υπογραφές μπορούν να χρησιμοποιηθούν από κινητούς πράκτορες για μία μόνο εκτέλεση σε κάποιον υπολογιστή, και όχι για πολλαπλές εκτελέσεις. Αρκετά σχήματα υπογραφής για κινητούς πράκτορες βασίστηκαν ή επέκτειναν τις δυνατότητες του σχήματος μη-αποσπώμενης υπογραφής RSA. Το σχήμα για μη-αποσπώμενες πολυ-υπογραφές που παρουσιάζεται επίσης στο [1] ανήκει σε αυτή την κατηγορία. Βασίζεται στο σχήμα πολυ-υπογραφών των Mitomi και Miyaji [18] και επιτρέπει την εκτέλεση του πράκτορα σε πολλούς υπολογιστές. Επίσης, προσφέρει καταλογισμό ευθύνης τόσο για τον πράκτορα όσο και για το περιβάλλον εκτέλεσης.

Οι Borselius et al [19] επέκτειναν τις μη-αποσπώμενες υπογραφές σε threshold υπογραφές, όπου για τη δημιουργία μιας έγκυρης υπογραφής απαιτείται η συνεργασία *n* από *k* πράκτορες ($n < k$). Το σχήμα αυτό χρησιμοποιήθηκε από τους ίδιους ερευνητές για ηλεκτρονικές συναλλαγές με κινητούς πράκτορες.

Οι μη-αποσπώμενες υπογραφές έχουν επεκταθεί επίσης σε υπογραφές πληρεξουσιότητας (proxy-signatures). Αναλυτικότερα, οι Kim et al. [20] πρότειναν ένα σχήμα για υπογραφές πληρεξουσιότητας μιας χρήσης (one-time proxy signatures), το οποίο επιτρέπει σε έναν κινητό πράκτορα να υπογράφει με ασφάλεια ένα μήνυμα. Επίσης οι Lee et al [21] πρότειναν ένα σχήμα για ισχυρές υπογραφές πληρεξουσιότητας (strong proxy signatures) που προσφέρουν διπλό καταλογισμό ευθύνης.

Πολλές ακόμα προσπάθειες μπορούν να αναφερθούν: οι Kotzanikolaou *et al.* προτείνουν ένα σύστημα με έναν «αφέντη» (master) και πολλούς «δούλους» (slave) πράκτορες, προκειμένου να περιορίζεται η χρήση του ιδιωτικού κλειδιού. Οι Rom˜ao *et al.* προτείνουν πιστοποιητικά πληρεξουσιότητας (proxy certificates), τα οποία λήγουν μετά από παρέλευση ορισμένου χρόνου. Οι Ferreira and Dahab παρουσιάζουν την ιδέα ενός ιδιωτικού κλειδιού που έχει γίνει blinded. Οι Yi et al προτείνουν ζεύγη κλειδιών μιας χρήσης, δηλαδή ένα για κάθε μήνυμα (one-time key pair), ενώ οι Bellare and Miner και Krawczyk την ανανέωση (update) του ιδιωτικού κλειδιού σε τακτά χρονικά διαστήματα. Οι Goldreich *et al.* εισάγουν μια διαφορετική ιδέα: ο χρήστης έχει ένα κύριο κλειδί, το οποίο χωρίζει σε δευτερεύοντα, που όταν συνενώνονται δίνουν πάλι το αρχικό κλειδί, προκειμένου να μην αναθέτονται σε κάποιον τα συνολικά προνόμια με τα οποία συνδέεται. Τέλος, αξίζει να αναφερθεί και μια άλλη κατηγορία λύσεων, σύμφωνα με την οποία ο κινητός πράκτορας συνεργάζεται με κάποιον εξυπηρετητή. Οι S^3 (Server Supported Signatures) παρουσιάστηκαν από τον Asokan *et al.* Ο Server-Aided Secret Computation είναι μια γενική μέθοδος, κατά την οποία ένας εξυπηρετητής εκτελεί συγκεκριμένους υπολογισμούς. Παράδειγμα αυτής της μεθόδου αποτελούν τα πρωτόκολλα Server-aided RSA (Merkle and Werchner 1998) [4].

3.2.4 Ασφάλεια κινητών πρακτόρων και κρυπτογραφία

Σε αυτό το σημείο αναφέρουμε διάφορους κρυπτογραφικούς μηχανισμούς και τεχνικές που μπορούν να χρησιμοποιηθούν για την προστασία εφαρμογών κινητού πράκτορα. Η ταξινόμηση των μηχανισμών έγινε με κριτήριο τις υπηρεσίες ασφάλειας που παρέχουν, δηλαδή εμπιστευτικότητα, ακεραιότητα, αυθεντικοποίηση, εξουσιοδότηση και καταλογισμό ευθύνης.

Με κριτήριο την **εμπιστευτικότητα** διακρίνουμε τους εξής κρυπτογραφικούς μηχανισμούς:

1) *συμμετρικούς κρυπτογραφικούς αλγόριθμους*, π.χ. για να επιτρέπεται η εκτέλεση ενός πράκτορα μόνο σε ορισμένους υπολογιστές. Ο πράκτορας κρυπτογραφείται πριν τη μετανάστευσή του με ένα συμμετρικό κλειδί, το οποίο διανέμεται σε εξουσιοδοτημένους υπολογιστές. Μόνο οι συγκεκριμένοι υπολογιστές που γνωρίζουν αυτό το κλειδί μπορούν να εκτελέσουν τον πράκτορα.

2) *υβριδική κρυπτογραφία*, η οποία επιτυγχάνει καλύτερη αποδοτικότητα από τον προηγούμενο μηχανισμό, ή κρυπτογράφηση ορισμένων μόνο τμημάτων ενός πράκτορα. Όταν ο πράκτορας εκτελείται σειριακά, είναι δυνατόν να κρυπτογραφούνται τα μερικά αποτελέσματα εκτέλεσής του, ώστε να προστατεύονται από τους υπολογιστές.

3) *κρυπτογραφία ολίσθησης (sliding encryption):* Η συγκεκριμένη τεχνική επιτρέπει σε μικρές ποσότητες δεδομένων να κρυπτογραφούνται και να σχηματίζουν αποτελέσματα αποδοτικού μήκους. Ο κινητός πράκτορας, χρησιμοποιώντας το δημόσιο κλειδί το οποίο φέρει, κρυπτογραφεί την πληροφορία που συγκεντρώνει από κάθε πλατφόρμα που επισκέπτεται και στη συνέχεια, όταν ο πράκτορας επιστρέφει στο ίδιο σημείο, η πληροφορία αποκρυπτογραφείται με το αντίστοιχο μυστικό κλειδί που διατηρείται εκεί.

Σε ό,τι αφορά την **ακεραιότητα**, **αυθεντικότητα** και τον **καταλογισμό ευθύνης**, διακρίνονται οι ακόλουθοι μηχανισμοί:

1) *άθροισμα ελέγχου ακεραιότητας και κώδικας αυθεντικοποίησης*: Μιλώντας για άθροισμα ελέγχου ακεραιότητας εννοούμε, στην περίπτωση κινητών πρακτόρων, το αποτέλεσμα μιας συνάρτησης κατακερματισμού (hash), με είσοδο τον εκτελέσιμο κώδικα, τα δεδομένα και την κατάσταση του πράκτορα. Αυτό αποστέλλεται στον παραλήπτη μαζί με τον πράκτορα, ο παραλήπτης υπολογίζει και πάλι τα αποτελέσματα της συνάρτησης κατακερματισμού και αν το αποτέλεσμα είναι ίδιο με το άθροισμα ακεραιότητας που συνοδεύει τον πράκτορα, τότε ο πράκτορας δεν έχει τροποποιηθεί. Το άθροισμα ελέγχου ακεραιότητας μπορεί να συνδυαστεί με κρυπτογράφηση για έλεγχο αυθεντικοποίησης. Κρυπτογραφείται με μυστικό συμμετρικό κλειδί, γνωστό στον αποστολέα και τον παραλήπτη του πράκτορα, και το αποτέλεσμα είναι ο λεγόμενος κώδικας αυθεντικοποίησης. Με αυτόν τον τρόπο προστατεύεται το άθροισμα ελέγχου ακεραιότητας ενώ ταυτόχρονα, γίνεται και ασθενής αυθεντικοποίηση.

2) *ψηφιακές υπογραφές και υπογεγραμμένος κώδικας:* Όπως είδαμε με τη αμέσως προηγούμενη τεχνική επιτυγχάνεται ασθενής μόνο αυθεντικοποίηση, γιατί δεν παρέχει δυνατότητα για μονοσήμαντη σύνδεση ενός πράκτορα με κάποιον αποστολέα. Αντιθέτως, οι ψηφιακές υπογραφές συνδυάζουν αλγορίθμους δημοσίου κλειδιού με συναρτήσεις κατακερματισμού και με αυτόν τον τρόπο, παρέχουν ακεραιότητα και ισχυρή αυθεντικοποίηση. Στην περίπτωση κινητού πράκτορα, ο συγγραφέας του μπορεί να υπογράφει τον πράκτορα με το μυστικό του κλειδί για έλεγχο ακεραιότητας, ο αποστολέας με τη σειρά του μπορεί να τον υπογράφει με το δικό του μυστικό κλειδί για την αυθεντικότητα του αποστολέα και τέλος, ο παραλήπτης μπορεί να ελέγχει τόσο την ακεραιότητα του πράκτορα, όσο και την αυθεντικότητα του αποστολέα, χρησιμοποιώντας τα αντίστοιχα κλειδιά. Τα παραπάνω καθιστούν αναγκαία την ύπαρξη μιας Υποδομής Δημοσίου Κλειδιού (Public Key Infrastructure), ώστε να συνδεθούν τα δημόσια κλειδιά με τις διάφορες οντότητες.

3) *Μη αποσπώμενες υπογραφές:* Για τον καταλογισμό ευθύνης γίνεται χρήση ψηφιακών υπογραφών: ο χρήστης υπογράφει ψηφιακά ένα μήνυμα χρησιμοποιώντας το μυστικό του κλειδί και στη συνέχεια η υπογραφή επαληθεύεται με τη χρήση του αντίστοιχου δημοσίου κλειδιού. Όταν όμως πρόκειται για κινητούς πράκτορες, οι συμβατικές ψηφιακές υπογραφές δεν επαρκούν για καταλογισμό ευθύνης, καθώς δεν πρέπει να αποκαλύπτεται το μυστικό κλειδί στο περιβάλλον εκτέλεσης του πράκτορα. Με τις μη αποσπώμενες υπογραφές RSA είναι υπολογιστικά αδύνατο για έναν υπολογιστή να αποσπάσει το κλειδί υπογραφής από τα στατικά δεδομένα του πράκτορα. Αυτού του τύπου οι υπογραφές δεν απαιτούν αλληλεπίδραση του πράκτορα με τον αποστολέα του για τον υπολογισμό μιας υπογραφής.

4) *Υπογραφές πληρεξουσιότητας μιας χρήσης (Proxy signatures):* επιτρέπουν σε κάποιο χρήστη να μεταβιβάσει το δικαίωμα της υπογραφής του σε κάποιον πληρεξούσιό του, ο οποίος μπορεί πλέον να υπογράφει για λογαριασμό του αρχικού χρήστη. Ο μηχανισμός αυτός μπορεί να χρησιμοποιηθεί για τον υπολογισμό μιας υπογραφής από έναν πράκτορα, ενώ προσφέρει επίσης καταλογισμό ευθύνης τόσο για τον πράκτορα όσο και για τον υπολογιστή εκτέλεσης.

5) *Δυναμικές πολύ - υπογραφές:* αντιμετωπίζουν τις αδυναμίες, τόσο των μη αποσπώμενων υπογραφών, όσο και των υπογραφών πληρεξουσιότητας μιας χρήσης, σε ό,τι αφορά τη μεταφερσιμότητα του πράκτορα , επειδή για λόγους ασφαλείας δεν μπορούν να χρησιμοποιηθούν για περισσότερες από μία υπογραφές. Αυτό που τελικά πετυχαίνουν οι δυναμικές πολύ - υπογραφές, είναι αυτόνομοι πράκτορες, που μπορούν να μεταναστεύουν και να εκτελούνται σειριακά σε πολλαπλούς υπολογιστές οι οποίοι δεν είναι γνωστοί εκ των προτέρων, ενώ παράλληλα το σύστημα αυτό είναι ανθεκτικό σε επιθέσεις απομάκρυνσης, στις οποίες μια συνομωσία από κακόβουλους υπολογιστές εκτέλεσης προσπαθούν να απομακρύνουν από την πολύ – υπογραφή τη μερική υπογραφή ενός ή περισσοτέρων προηγούμενων υπολογιστών εκτέλεσης. Επιτυγχάνεται τελικά έτσι καταλογισμός ευθύνης τόσο για τον πράκτορα όσο και για τους συμμετέχοντες υπολογιστές.

6) Υπογραφές πληρεξουσιότητας χωρίς προκαθορισμένο αποστολέα: στηρίζονται στις υπογραφές πληρεξουσιότητας μιας χρήσης. Είναι κατάλληλες για συστήματα αυτόνομων πρακτόρων χωρίς προκαθορισμένο δρομολόγιο, παρέχοντας καταλογισμό ευθύνης μόνο για

τον πράκτορα, μπορώντας όμως να χρησιμοποιηθούν από τον πράκτορα για πολλαπλές εκτελέσεις και υπογραφές.

7) Αλυσωτές υπογραφές και πολύ – υπογραφές: Χρησιμοποιούνται για τον έλεγχο της ακεραιότητας τόσο του στατικού όσο και του δυναμικού μέρους ενός πράκτορα, ο οποίος πρόκειται να εκτελέσει τον κώδικά του σε πολλούς υπολογιστές. Η υπογραφή περιλαμβάνει ολόκληρη την αλυσίδα όλων των υπογραφών από προηγούμενες εκτελέσεις του πράκτορα στους διάφορους υπολογιστές . Όμως η ασφάλεια των αλυσωτών υπογραφών δεν μπορεί να αποδειχθεί με χρήση συμβατικών κρυπτογραφικών υποθέσεων, ενώ επίσης οι συγκεκριμένες υπογραφές δεν είναι ιδιαίτερα αποδοτικές, ούτε και ανθεκτικές σε επιθέσεις απομάκρυνσης προηγούμενων υπογραφών από κακόβουλους υπολογιστές εκτέλεσης. Οι δυναμικές πολύ – υπογραφές που αναφέρθηκαν προηγουμένως είναι αποδοτικότερες, η ασφάλεια τους ελέγχεται ευκολότερα και μπορούν να χρησιμοποιηθούν και για τον έλεγχο ακεραιότητας του πράκτορα πέραν του καταλογισμού ευθύνης.

8) Ισχυρή χρονική ασφάλεια(strong forward security) : προτείνεται ως μέθοδος για την ελαχιστοποίηση των συνεπειών της μη εξουσιοδοτημένης αποκάλυψης ενός ιδιωτικού κλειδιού. Μπορούν να προστατευτούν δεδομένα τα οποία έχουν υποστεί κρυπτογραφική επεξεργασία πριν την αποκάλυψη του μυστικού κλειδιού, ειδικά σε περιπτώσεις όπου είναι δύσκολο να εντοπισθούν εισβολές, με τον εισβολέα να μη χρησιμοποιεί τα υποκλεμμένα κλειδιά μέχρις ότου αυτό να είναι πρόσφορο ή κερδοφόρο. Ανά τακτά χρονικά διαστήματα γίνεται ανανέωση του ζεύγους ιδιωτικού / δημοσίου κλειδιού και εξασφαλίζεται πως πιθανή αποκάλυψη του μυστικού κλειδιού κατά τη διάρκεια μιας περιόδου δεν θα προσβάλει την ασφάλεια του συστήματος για τις περιόδους που προηγήθηκαν ή τις περιόδους που θα ακολουθήσουν την περίοδο κατά την οποία έγινε η αποκάλυψη. Αρχικά κάθε χρήστης επιλέγει τυχαία ένα ζεύγος μυστικού / δημοσίου κλειδιού και το πιστοποιεί μέσω της CA, αφού αρχικά αυθεντικοποιηθεί στην Αρχή με φυσικό τρόπο. Στη συνέχεια, στο τέλος κάθε περιόδου επιλέγει ένα νέο τυχαίο ζεύγος κλειδιών. Το μυστικό κλειδί της τρέχουσας περιόδου χρησιμοποιείται για να υπογραφεί ψηφιακά το δημόσιο κλειδί της νέας περιόδου. Αυτή η υπογραφή, μαζί με το δημόσιο κλειδί της νέας περιόδου, αποστέλλονται μαζί στην CA ώστε να πιστοποιηθεί το νέο δημόσιο κλειδί. Η CA επαληθεύει την υπογραφή καθώς και το πιστοποιητικό της τρέχουσας περιόδου και σε περίπτωση που είναι έγκυρα, εκδίδει ψηφιακό πιστοποιητικό για το δημόσιο κλειδί της νέας περιόδου. Με τον τρόπο αυτό παρέχεται ισχυρή χρονική ασφάλεια, γιατί σε περίπτωση που το μυστικό κλειδί αποκαλυφθεί σε κάποια χρονική περίοδο χωρίς να το αντιληφθεί ο νόμιμος κάτοχος του κλειδιού, τότε για την επόμενη χρονική περίοδο δύο δημόσια κλειδιά θα υποβληθούν στην Αρχή για πιστοποίηση: ένα από τον αντίπαλο και ένα από το νόμιμο χρήστη. Και τα δύο κλειδιά θα φαίνονται σωστά υπογεγραμμένα, αφού ο αντίπαλος μπορεί να πλαστογραφεί τις υπογραφές του νόμιμου χρήστη. Αυτό θα υποδείξει στην CA ότι το κλειδί του χρήστη έχει αποκαλυφθεί και κανένα από τα δημόσια κλειδιά δεν θα πιστοποιηθεί. Με βάση το μηχανισμό αυτό υλοποιείται πολιτική πρόσβασης βασισμένη σε μεταβίβαση ρόλων.

Τέλος με γνώμονα την **εξουσιοδότηση** βλέπουμε τα εξής:

1) *Αποτίμηση κατάστασης:* προτάθηκε για έλεγχο της κατάστασης του πράκτορα, πριν του παραχωρηθούν οποιαδήποτε δικαιώματα πρόσβασης, επιβεβαιώνοντας έτσι ότι η κατάσταση του δεν έχει αλλοιωθεί κατά τη διάρκεια της μετάδοσής του. Ανάλογα με το βαθμό αλλοίωσης δίνονται περιορισμένα ή και καθόλου δικαιώματα στον πράκτορα. Έτσι ο μηχανισμός αυτός ελέγχει την ακεραιότητα των πρακτόρων. Οι συναρτήσεις αποτίμησης κατάστασης γίνονται μέρος του κώδικα του πράκτορα και μπορεί να δημιουργούνται είτε από το συγγραφέα είτε από τον ιδιοκτήτη του πράκτορα, ενώ παράλληλα δρουν και ως μηχανισμός εξουσιοδότησης, εφόσον χρησιμοποιούνται για την παραγωγή διαφορετικών επιπέδων πρόσβασης ανάλογα με το αποτέλεσμά τους. Ένα πρόβλημα που εμφανίζει η μέθοδος είναι ότι αν και μπορεί να σχεδιασθεί εύκολα για να αντιμετωπίσει περισσότερο προφανείς επιθέσεις , δύσκολα αντιμετωπίζει επιθέσεις που δεν έχουν προβλεφθεί ή εντοπιστεί.

2) *Πιστοποιητικά χαρακτηριστικών (attribute certificates) :* χρησιμοποιούνται για τον έλεγχο της συμπεριφοράς κινητού κώδικα και κινητών πρακτόρων. Τα χαρακτηριστικά αναφέρονται σε κανόνες πολιτικής, που ελέγχουν την πρόσβαση σε υπολογιστικούς πόρους και υπηρεσίες. Μέσω των πιστοποιητικών αυτών γίνεται δυναμική αντιστοίχηση δικαιωμάτων πρόσβασης για κινητό κώδικα και κινητούς πράκτορες, τα οποία δικαιώματα παραχωρούνται σε κάποια οντότητα μαζί με κάποια άλλη πληροφορία, όπως π.χ. τον εκδότη του πιστοποιητικού, τον ιδιοκτήτη, την ημερομηνία και ώρα λήξης, καθώς και μια ψηφιακή υπογραφή που αυθεντικοποιεί το πιστοποιητικό. Για την πιστοποίηση χρησιμοποιείται συνήθως μια Υποδομή Δημοσίου Κλειδιού. Πιστοποιητικά χαρακτηριστικών μπορούν επίσης να εκδοθούν για υπολογιστές εκτέλεσης.

3) *Μεταβίβαση ρόλου (role delegation):* είναι η ιδιότητα μιας ενεργής οντότητας, για παράδειγμα ενός χρήστη του συστήματος που είναι μέλος ενός ρόλου με συγκεκριμένα δικαιώματα πρόσβασης, να εξουσιοδοτήσει μια άλλη οντότητα να γίνει μέλος του ρόλου αυτού για μια συγκεκριμένη χρονική περίοδο. Η βασική χρήση των κινητών πρακτόρων είναι η μεταβίβαση εργασίας, δηλαδή η ανάθεση σε ένα πράκτορα μιας εργασίας προς διεκπεραίωση εκ μέρους μιας άλλης οντότητας. Έτσι γίνεται μεταβίβαση δικαιωμάτων μεταξύ οντοτήτων, με προστασία από μη εξουσιοδοτημένη μεταβίβαση και με ελεγχόμενη διαδικασία ανάκλησης της μεταβίβασης ρόλων. Ο κρυπτογραφικός αυτός μηχανισμός σχεδιάζεται με τη βοήθεια άλλων μηχανισμών, όπως οι μη αποσπώμενες υπογραφές, η ισχυρή χρονική ασφάλεια και τα πιστοποιητικά χαρακτηριστικών. Επειδή η μεταβίβαση ρόλου μεταξύ πρακτόρων μπορεί να πραγματοποιηθεί σε εχθρικά (μη έμπιστα) περιβάλλοντα εκτέλεσης μπορεί να οδηγήσει σε ανεξέλεγκτη ροή προνομίων πρόσβασης. Συνεπώς απαιτείται η λήψη κατάλληλων μέτρων ασφαλείας, τα οποία να προστατεύουν την αυθεντικότητα και γνησιότητα της μεταβίβασης ρόλου κινητών πρακτόρων.

4) *Threshold μη αποσπώμενες υπογραφές:* ο συγκεκριμένος κρυπτογραφικός μηχανισμός επιτρέπει τη διαμοίραση μιας συνάρτησης μεταξύ ορισμένων οντοτήτων, ώστε η συνάρτηση να μπορεί να υπολογιστεί για κάποια τιμή εισόδου (για παράδειγμα για κάποιο μήνυμα που πρέπει να υπογραφεί), μόνο αν συνεργαστεί ένα πλήθος οντοτήτων μεγαλύτερο από ένα προκαθορισμένο όριο (threshold). Ακόμα κι αν ορισμένες από τις εμπλεκόμενες οντότητες συμπεριφέρονται λανθασμένα ή είναι υπό τον έλεγχο κακόβουλων χρηστών, η κρυπτογραφική συνάρτηση θα υπολογιστεί σωστά, υπό την προϋπόθεση ότι ο αριθμός των οντοτήτων που λειτουργούν όπως απαιτεί το πρωτόκολλο είναι μεγαλύτερος από το προκαθορισμένο όριο [1].

3.3 Βιβλιογραφία

[1] Κοτζανικολάου Παναγιώτης, "Ασφαλή συστήματα κινητών πρακτόρων", Διδακτορική Διατριβή,Πανεπιστήμιο Πειραιώς,Τμήμα Πληροφορικής,Πειραιάς, Ιανουάριος 2003.

[2] Ιωάννης Ε. Φουκαράκης, "Ανάπτυξη υποδομής κινητών πρακτόρων με χρήση τεχνολογιών διαδικτύου", Διπλωματική Εργασία, ΕΜΠ, Αθήνα, Νοέμβριος 2003.

[3] Θεόδωρος Ε. Αθαναηλέας, "Επέκταση πλατφόρμας κινητών αντιπροσώπων με χρήση δικτυακών υποδοχών", Διπλωματική Εργασία, ΕΜΠ, Αθήνα, Ιούνιος 2005.

[4] Joris Claessens, Bart Preneel and Joos Vandewalle, "(How) can mobile agents do secure electronic transactions on untrusted hosts?-A survey of the security issues and the current solutions", COmputer Security and Industrial Cryptography(COSIC) Dept. of Electrical Engineering-ESAT, K.U.Leuven, Belgium, September 2002.

[5] Vigna, G. "Protecting Mobile Agents through Tracing". In Proceedings of the Third ECOOP Workshop on Mobile Object Systems: Operating System support for Mobile Object Systems, 1997.

[6] Yee, B. S. "A Sanctuary for Mobile Agents". In Secure Internet Programming: Security Issues for Mobile and Distributed Objects, J. Vitek and C. Jensen, Eds. Lecture Notes in Computer Science, LNCS 1603. Springer-Verlag, 1999, pp. 261–274.

[7] Biehl, I., Meyer, B., and Wetzel, S. "Ensuring the Integrity of Agent-Based Computations by Short Proofs". In *Proceedings of the Second International Workshop on Mobile Agents,1998*.

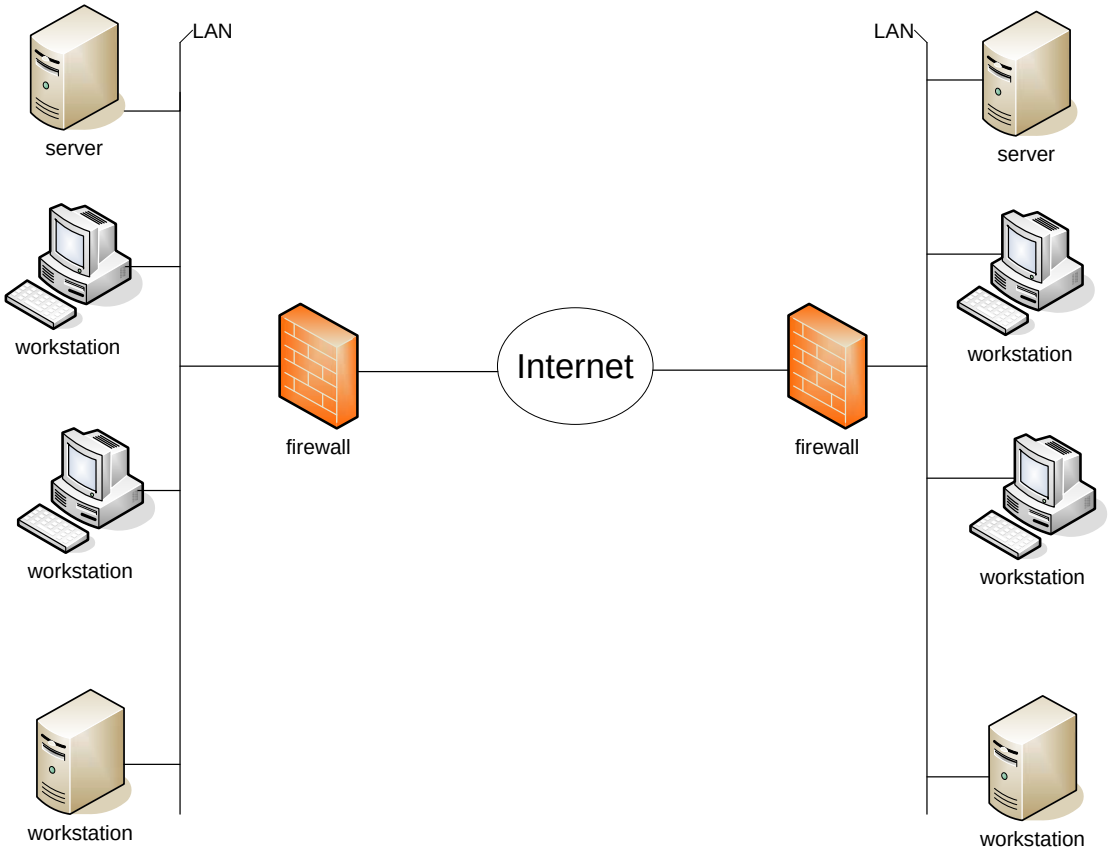
[8] Yi, X., Siew, C. K., and Syed, M. R. “Digital signature with one-time pair of keys”. Electronics Letters 36, 2, January 2000, pp. 130–131.
[9] Minsky, Y., van Renesse, R., Schneider, F. B., and Stoller, S. D. “Cryptographic Support for Fault-Tolerant Distributed Computing”. In Proceedings of the Seventh ACM SIGOPS European Workshop, 1996, pp. 109–114.
[10] Hohl, F. “A Framework to Protect Mobile Agents by Using Reference States”. In Proceedings of the 20th International Conference on Distributed Computing Systems, 2000.
[11] Wilhelm, U. G., Staamann, S., and Butty_an, L. “On the Problem of Trust in Mobile Agent Systems”. In Proceedings of the 1998 Network and Distributed System Security (NDSS’98) Symposium, 1998.
[12] Riordan, J. and Schneier, B. “Environmental Key Generation Towards Clueless Agents”. In Mobile Agents and Security, G. Vigna, Ed. Lecture Notes in Computer Science, LNCS 1419. Springer-Verlag, 1998, pp. 15–24.
[13] F. Hohl, “Time limited blackbox security: Protecting mobile agents from malicious hosts”, Lecture Notes in Computer Science Vol. 1419, 1998, pp. 92.113.
[14] T. Sander and C. Tschudin, “Protecting mobile agents against malicious hosts”, Lecture Notes in Computer Science Vol. 1419, Springer, 1998, pp. 44.60.
[15] D. Naccache and J. Stern, “A new public-key cryptosystem”, Procceedings of Eurocrypt'97, Lecture Notes in Computer Science Vol. 1223, Springer, 1997, pp. 27.36.
[16] C. Cachin, J. Camenisch, J. Kilian, and J. Muller, “The one-round secure computation and secure autonomous mobile agents”, Proceedings of 27th ICALP, Lecture Notes in Computer Science Vol. 1853, Springer, 2000, pp. 512.523.
[17] S. Loureiro and R. Molva, “Privacy for mobile code”. In proceedings of Distributed Object Security Workshop OOPSLA'99 (Denver), Springer, November 1999.
[18] S. Mitomi and A. Miyaji, “A multisignature scheme with message flexibility, order flexibility and order verifiability”, Procceedings of 5th ACISP, Lecture Notes in Computer Science Vol.1841, Springer, July 2000, pp. 298.312.
[19] “Undetachable threshold signatures”, Proceedings of the 8th IMA Conference on Cryptography and Coding, Lecture Notes in Computer Science Vol. 2260, Springer, December 2001, pp. 239.244.
[20] H. Kim, J. Baek, B. Lee, and K. Kim, “Secret computation with secrets for mobile agent using one-time proxy signature”, Symposium on Cryptography and Information Security Vol 2/2, The Institute of Electronics, Information and Communication Engineers, January 2001, pp. 845.850.
[21] B. Lee, H. Kim, and K. Kim, “Secure mobile agent using strong non-designated proxy signature”, Proceedings of ACISP 2001, Lecture Notes in Computer Science Vol. 2119, Springer, 2001, pp. 474.486.

4 Μηχανισμοί και τεχνικές κρυπτογραφίας

4.1 Εισαγωγή

Στα σημερινά υπολογιστικά και δικτυακά περιβάλλοντα η ασφάλεια καλείται να καλύψει τόσο φυσικούς πόρους, λειτουργικά συστήματα και λογισμικά εφαρμογών, όσο και τα δεδομένα τα οποία αυτά αποθηκεύουν και επεξεργάζονται. Επιπλέον, ο έλεγχος της πρόσβασης σε δίκτυα, τα τείχη προστασίας (firewalls) και άλλες τεχνολογίες «φιλτραρίσματος» θεωρούνται πλέον απαραίτητες ώστε να προστατευθούν τα όρια των εκάστοτε δικτυακών υποδομών.

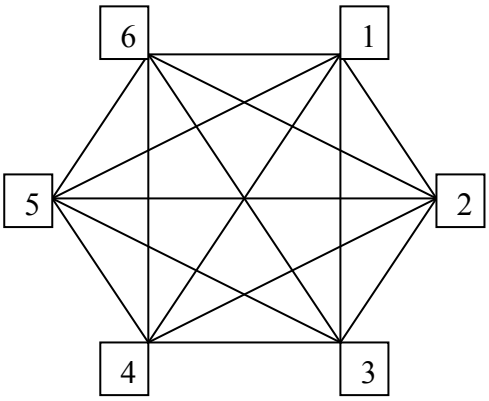
Έξω από αυτά τα όρια τα μη προστατευμένα δεδομένα που στέλνονται και λαμβάνονται από τον αντίστοιχο οργανισμό πάνω από τα διάφορα δίκτυα είναι ευάλωτα σε αποκάλυψη, τροποποίηση, διαγραφή και άλλων ειδών επιθέσεις. Προκειμένου να προστατεύονται τα μεταφερόμενα δεδομένα πάνω από μη αξιόπιστα δίκτυα, η μόνη εφαρμόσιμη και οικονομικά συμφέρουσα τεχνολογία είναι η κρυπτογραφία, η οποία βρίσκεται στο επίκεντρο σε ό,τι αφορά τόσο τα εικονικά ιδιωτικά δίκτυα (virtual private networks, VPNs) όσο και τις τεχνολογίες δημόσιου κλειδιού (public key infrastructures).



Σχήμα 4.1 Το σημερινό υπολογιστικό και επικοινωνιακό περιβάλλον.

4.2 Κρυπτογραφία μυστικού κλειδιού

Η συμμετρική κρυπτογραφία μυστικού κλειδιού (Secret Key Cryptography) χρησιμοποιεί το ίδιο κλειδί για την κρυπτογράφηση και κατόπιν αποκρυπτογράφηση ενός κειμένου. Προκειμένου *n* οντότητες να επικοινωνούν με ασφάλεια μεταξύ τους, ένα σύστημα κρυπτογράφησης μυστικού κλειδιού θα απαιτεί *n*(n-1)/2* κλειδιά (Σχήμα 4.2), παρόλο που κάθε οντότητα χρειάζεται να γνωρίζει μόνο *n* κλειδιά. Για ένα σύνολο έξι χρηστών, αυτό σημαίνει απλά 15 κλειδιά. Αλλά για ένα σύνολο 1000 χρηστών, για παράδειγμα, η διαχείριση και ασφαλής ανταλλαγή σχεδόν μισού εκατομμυρίου μυστικών κλειδιών είναι πρακτικά μη εφαρμόσιμη[2].



Σχήμα 4.2 Μυστικά κλειδιά στην περίπτωση 6 χρηστών.

4.3 Κρυπτογραφικά συστήματα δημόσιου κλειδιού

Σε αντίθεση με τα παραπάνω, τα κρυπτογραφικά συστήματα δημόσιου κλειδιού (public key cryptosystems, PKCs) χρησιμοποιούν ζεύγη συσχετιζόμενων κλειδιών που δημιουργούνται μαζί. Το κρυπτογραφημένο κείμενο που παράγεται από το ένα κλειδί μπορεί να αποκρυπτογραφηθεί μόνο με το άλλο μέλος του ίδιου ζεύγους κλειδιών. Το ένα από τα κλειδιά αυτά παραμένει μυστικό (*ιδιωτικό κλειδί, private key)* και το άλλο δημοσιοποιείται και μπορεί να χρησιμοποιηθεί από όλους (*δημόσιο κλειδί, public key)*.

Για την απόκρυψη ενός μηνύματος κατά τη μεταφορά, ώστε μόνο ο επιθυμητός παραλήπτης να μπορεί να το διαβάσει, το αρχικό κείμενο κρυπτογραφείται με το δημόσιο κλειδί του παραλήπτη. Μόνο το μυστικό κλειδί του παραλήπτη μπορεί να αποκρυπτογραφήσει το μεταφερόμενο κρυπτογραφημένο κείμενο. Παρόμοια, για την εξακρίβωση της ακεραιότητας και αυθεντικότητας ενός μηνύματος, είναι πιθανό να κρυπτογραφούνται πληροφορίες με το ιδιωτικό κλειδί του αποστολέα. Αυτό επιτρέπει σε όλους όσους έχουν πρόσβαση στο δημόσιο κλειδί αυτού του αποστολέα να επικυρώνουν το μήνυμα αποκρυπτογραφώντας το επιτυχώς.

4.3.1 Πλεονεκτήματα συστημάτων δημόσιου έναντι κρυφού κλειδιού

Για τη διασφάλιση της μεταφοράς δεδομένων, οι τεχνολογίες δημόσιου κλειδιού προτιμούνται από τις τεχνολογίες μυστικού κλειδιού για τους ακόλουθους λόγους:

- Απαιτούν λιγότερα κλειδιά προς διαχείριση: κάθε οντότητα (*n*) έχει ένα ζεύγος κλειδιών, έτσι ώστε ο συνολικός αριθμός κλειδιών είναι *2n* και όχι ανάλογο του n^2 , όπως στην περίπτωση των συστημάτων μυστικού κλειδιού.
- Εφόσον τα ιδιωτικά κλειδιά δε χρήζουν διανομής ή άλλης διαχείρισης, τα συστήματα δημόσιου κλειδιού απαιτούν την επίδειξη ακεραιότητας και αυθεντικότητας μόνο του δημόσιου κλειδιού. Οι χρήστες πρέπει να έχουν τη διαβεβαίωση ότι τα δημόσια κλειδιά τους όντως ανήκουν στους εκδότες τους.
- Επειδή δε μεταφέρονται κάποιου είδους κρυφά κλειδιά πάνω από οποιοδήποτε δίκτυο, τα PKCs δεν εκτίθενται εύκολα σε κίνδυνο, ακόμα και αν τα δημόσια κλειδιά πρέπει να

αλλάξουν. Τα PKCs μπορούν να χρησιμοποιηθούν για την κρυπτογράφηση προσωρινών κλειδιών (*session keys)* που μπορούν με τη σειρά τους να χρησιμοποιηθούν στην κρυπτογραφία μυστικού κλειδιού προς αποφυγή του μεγαλύτερου υπολογιστικού φορτίου του PKC.

- Για την κρυπτογράφηση ενός μηνύματος, έτσι ώστε οι πολλαπλοί χρήστες του PKC να λαμβάνουν και στη συνέχεια να αποκρυπτογραφούν το μήνυμα με ασφάλεια, το PKC λογισμικό μπορεί να δημιουργήσει ένα *κλειδί συνόδου (session key)*. Αυτό το μυστικό κλειδί κρυπτογραφείται έπειτα με το δημόσιο κλειδί κάθε παραλήπτη και τέλος αποστέλλεται μαζί με το κρυπτογραφημένο κείμενο σε όλους τους παραλήπτες, χωρίς να διακινδυνεύει το απόρρητο.
- Οι ψηφιακές υπογραφές δημόσιου κλειδιού παρέχουν τη βάση για τον καταλογισμό ευθύνης (non-repudiation) στην περίπτωση διένεξης. Μόνο ο κάτοχος ενός ιδιωτικού κλειδιού θα μπορούσε να έχει στείλει ένα μήνυμα αποκρυπτογραφημένο με το δημόσιο κλειδί του. Αντίθετα, λόγω του ότι τα μυστικά κλειδιά πρέπει να είναι γνωστά τουλάχιστον σε δύο κάθε φορά οντότητες, δεν μπορούν να υποστηρίξουν καταλογισμό ευθύνης[2].

4.3.2 Η ανάγκη για υποδομή δημόσιου κλειδιού

Τα συστήματα δημόσιου κλειδιού βασίζουν την αποτελεσματικότητά τους στην ακεραιότητα κάθε δημόσιου κλειδιού και στη διασύνδεσή του με μια συγκεκριμένη οντότητα, όπως ένα άτομο, έναν οργανισμό ή ένα τμήμα δικτύου. Χωρίς μηχανισμούς διασφάλισης της ακεραιότητας και αυθεντικότητας, οι χρήστες είναι ευάλωτοι σε επιθέσεις «μεταμφίεσης» μέσω αντικατάστασης του δημόσιου κλειδιού.

Ας υποθέσουμε, για παράδειγμα, ότι ο οργανισμός Α θέλει να στείλει ένα απόρρητο μήνυμα στον Β, χωρίς κανείς άλλος να μπορεί να το διαβάσει. Ο Α θα μπορούσε να χρησιμοποιήσει το δημόσιο κλειδί του Β για να κρυπτογραφήσει το μήνυμα, ωστόσο για καλύτερη επίδοση ο Α θα χρησιμοποιούσε μάλλον συμμετρικό αλγόριθμο για την κρυπτογράφηση και αλγόριθμο δημόσιου κλειδιού για να κρυπτογραφήσει το συμμετρικό κλειδί. Αν όμως ο Α «ξεγελαστεί» και αντ' αυτού χρησιμοποιήσει το δημόσιο κλειδί ενός τρίτου οργανισμού Γ σα να ήταν του Β, τότε ο Γ θα μπορούσε να αποκρυπτογραφήσει το μήνυμα. Η τεχνική αυτή είναι γνωστή ως *απάτη δημόσιου κλειδιού.*

Μια τέτοια απάτη από πλευράς του Γ θα εμπόδιζε από την άλλη τον Β να διαβάσει το μήνυμα του Α, όπως έπρεπε αρχικά. Έτσι μια τέτοια επίθεση θα συνεχιζόταν με τον Γ να επανακρυπτογραφεί τα μηνύματα του Α, χρησιμοποιώντας το πραγματικό δημόσιο κλειδί του Β, και να τα στέλνει τελικά στον Β. Μια τέτοια παρεμπόδιση και επανακρυπτογράφηση αποτελεί παράδειγμα επίθεσης ενδιάμεσου *(man-in-the-middle attack).*

Ένα άλλο παράδειγμα ρήξης της σύνδεσης ανάμεσα σε ένα δημόσιο κλειδί και τον ιδιοκτήτη του περιλαμβάνει την εξακρίβωση ηλεκτρονικών υπογραφών. Ας υποθέσουμε ότι ο Α θέλει να επαληθεύσει την υπογραφή του Β. Αν ο Γ μπορεί να ξεγελάσει τον Α ώστε ο τελευταίος να χρησιμοποιήσει το δικό του δημόσιο κλειδί σα να ήταν του Β, τότε ο Γ θα μπορούσε να υπογράφει μηνύματα μεταμφιεζόμενος ως Β χρησιμοποιώντας το δικό του (του Γ) ιδιωτικό κλειδί. Ο Α θα χρησιμοποιούσε ακούσια το δημόσιο κλειδί του Γ και θα νόμιζε εσφαλμένα ότι το μήνυμα όντως έχει υπογραφεί από τον Β.

Συμπερασματικά, τόσο για ψηφιακές υπογραφές όσο και για υπηρεσίες κρυπτογράφησης, κάθε χρήστης πρέπει να χρησιμοποιεί το δημόσιο κλειδί του σωστού «συνομιλητή», προκειμένου να εξασφαλίζεται η ασφάλεια. Υπάρχουν πολλοί φυσικοί, ηλεκτρονικοί και υβριδικοί μηχανισμοί για τη διανομή δημόσιων κλειδιών με αξιόπιστο τρόπο, ώστε κάθε χρήστης να είναι σίγουρος ότι έχει το σωστό δημόσιο κλειδί για κάθε άλλο χρήστη με τον οποίο συνδιαλέγεται. Αυτοί οι μηχανισμοί για τη διανομή και δέσμευση δημόσιων κλειδιών είναι γνωστοί ως υποδομή δημόσιου κλειδιού (Public Key Infrastructure, PKI)[2].

4.3.3 Πιστοποιητικό δημόσιου κλειδιού. Το πιστοποιητικό X.509

Η αντίστοιχη τεχνική χρησιμοποιεί ένα πιστοποιητικό δημόσιου κλειδιού, το οποίο έχει εκδοθεί από μια αξιόπιστη οντότητα, την *CA* (certification authority, CA). Μια CA εκδίδει πιστοποιητικά δημόσιου κλειδιού στους διάφορους συνδρομητές συγκεντρώνοντας τα

στοιχεία τους και υπογράφοντας τα στοιχεία αυτά με το ιδιωτικό της κλειδί. Το γενικά αποδεκτό πρότυπο για τα ψηφιακά πιστοποιητικά δημόσιου κλειδιού είναι το X.509 version 3. Το πιστοποιητικό κάθε CA περιέχει τις ακόλουθες πληροφορίες κλειδιού:

- Αριθμός έκδοσης του χρησιμοποιούμενου προτύπου.
- Σειριακός αριθμός (serial number) του πιστοποιητικού (μοναδικός για κάθε πιστοποιητικό που εκδίδει η CA)
- Ο αλγόριθμος και οι σχετικές παράμετροι που χρησιμοποιήθηκαν από τη CA για την υπογραφή του πιστοποιητικού.
- Το όνομα της CA.
- Η χρονική περίοδος για την οποία ισχύει το πιστοποιητικό.
- Το όνομα του συνδρομητή.
- Το δημόσιο κλειδί του συνδρομητή, ο αλγόριθμος δημόσιου κλειδιού και οι σχετικές παράμετροι.
- Ο μοναδικός αναγνωριστικός αριθμός της CA (προαιρετικό).
- Ο μοναδικός αναγνωριστικός αριθμός του συνδρομητή (προαιρετικό).
- Επεκτάσεις (προαιρετικό).
- Η ψηφιακή υπογραφή της CA.

Κάθε χρήστης πρέπει να γνωρίζει το δημόσιο κλειδί της CA, ώστε να μπορεί να επαληθεύει την ψηφιακή υπογραφή στα πιστοποιητικά που εκδίδονται από αυτή, και να το εμπιστεύεται. Αυτό το δημόσιο κλειδί λαμβάνεται κατά τη διαδικασία εγγραφής. Μόλις οι υπογραφές επαληθευτούν, οι χρήστες μπορούν να χρησιμοποιήσουν το όνομα και το δημόσιο κλειδί του συνδρομητή στο πιστοποιητικό, με τόση εμπιστοσύνη στην ακρίβεια της πληροφορίας, όση έχουν και στην αξιοπιστία της CA.

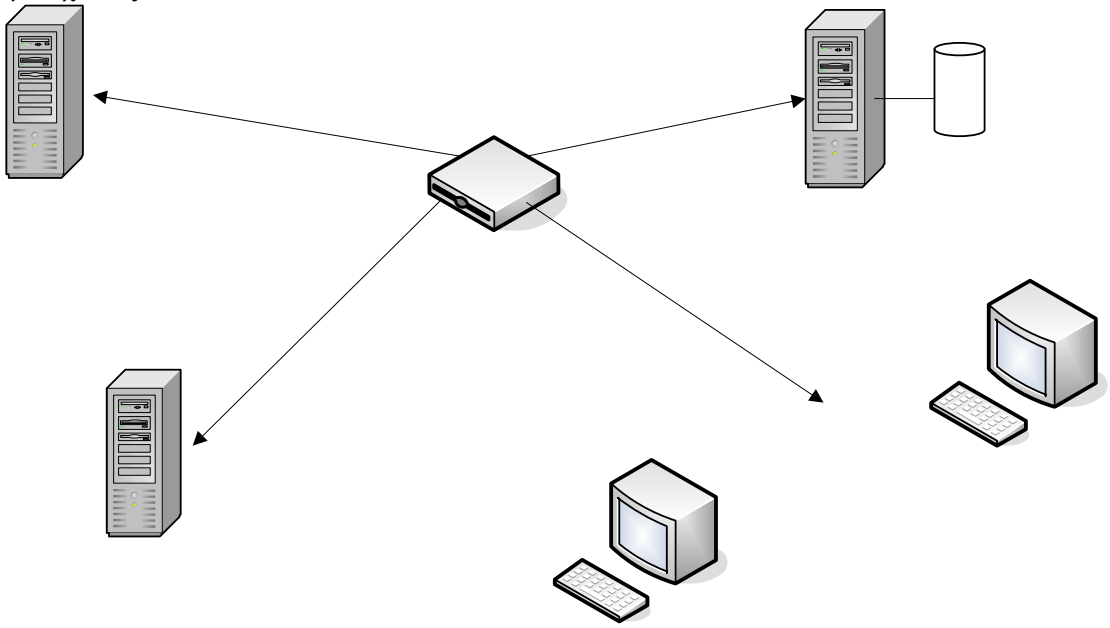
Σε κάποιες περιπτώσεις η CA ίσως χρειαστεί να ακυρώσει τη σύνδεση ενός συνδρομητή με το δημόσιο κλειδί του. Το ιδιωτικό κλειδί του συνδρομητή, για παράδειγμα, μπορεί να έχει υποκλαπεί. Εφόσον ένα πιστοποιητικό δημόσιου κλειδιού είναι ένα ηλεκτρονικό αντικείμενο και μπορεί να βρίσκεται σε διάφορα μέρη την ίδια στιγμή, δεν είναι ούτε εφαρμόσιμο ούτε πιθανό να ανακληθούν ή διαγραφούν ή σβηστούν όλα τα αντίγραφα ενός τέτοιου πιστοποιητικού σε ένα κατανεμημένο περιβάλλον. Έτσι προκειμένου να ακυρώσει ένα πιστοποιητικό παύοντας τη συσχέτιση ανάμεσα στο συνδρομητή και στο δημόσιο κλειδί του, η CA δημιουργεί μια λίστα από άκυρα πιστοποιητικά, τη *λίστα ανακληθέντων πιστοποιητικών* (certificate revocation list, CRL). Οι χρήστες πρέπει να ελέγχουν ότι ένα πιστοποιητικό δεν ανήκει στη λίστα προτού χρησιμοποιήσουν το δημόσιο κλειδί στο πιστοποιητικό. Αν το πιστοποιητικό είναι στη λίστα, ο χρήστης δεν πρέπει να το χρησιμοποιήσει. Η CA υπογράφει τη λίστα, ώστε οι χρήστες να επαληθεύουν την ακεραιότητα και αυθεντικότητά της. Οι πληροφορίες κλειδιού στην CRL X.509 version 2 είναι οι εξής:

- Αριθμός έκδοσης του χρησιμοποιούμενου προτύπου CRL.
- Ο αλγόριθμος και οι σχετικές παράμετροι που χρησιμοποιήθηκαν από τη CA για την υπογραφή του πιστοποιητικού.
- Η χρονική στιγμή έκδοσης αυτής της CRL.
- Η χρονική στιγμή έκδοσης της επόμενης CRL (προαιρετικό).
- Η λίστα με τα ανακληθέντα πιστοποιητικά (για καθένα δίνονται τα ακόλουθα):
    - Σειριακός αριθμός (serial number) του πιστοποιητικού
    - Η στιγμή κατά της οποία η CA ενημερώθηκε για την ανάκληση
    - Επεκτάσεις που σχετίζονται με το ανακληθέν πιστοποιητικό (προαιρετικό).
- Επεκτάσεις σχετικές με το CRL (προαιρετικό).
- Η ψηφιακή υπογραφή της CA. [2]

4.3.4 Υποδομή Δημοσίου Κλειδιού

Κάθε ομάδα χρηστών, η οποία προστατεύεται από μία CA, ονομάζεται *πεδίο* (*domain*). Στους συνδρομητές κάθε πεδίου διανέμονται πιστοποιητικά δημοσίου κλειδιού από την αρμόδια CA. Το Σχήμα 4.3 παρουσιάζει τους συμμετέχοντες σε μια Υποδομή Δημοσίου Κλειδιού για ένα πεδίο. Η CA είναι υπεύθυνη για την παραγωγή πιστοποιητικών και για την

παραγωγή της CRL. Στέλνει αυτά τα υπογεγραμμένα αντικείμενα σ' έναν καταχωρητή, μέσω του οποίου μπορούν οι έμπιστοι συμμετέχοντες να τα ανασύρουν. Επίσης αρχειοθετεί τα πιστοποιητικά και τις λίστες ανακληθέντων πιστοποιητικών, σε περίπτωση που ζητηθούν στον μέλλον για την επίλυση διαφωνιών ανάμεσα στους συνδρομητές και στους έμπιστους συμμετέχοντες.



Σχήμα 4.3 Συμμετέχοντες στην Υποδομή Δημόσιου Κλειδιού για ένα πεδίο.

Η Αρχή Εγγραφής (Registration Authority RA) είναι ο έμπιστος πράκτορας της CA και είναι υπεύθυνη για την επιβεβαίωση της ταυτότητας του συνδρομητή. Ουσιαστικά πραγματοποιεί τις ακόλουθες εργασίες:

- Εξακριβώνει την ταυτότητα που ο συνδρομητής ισχυρίζεται ότι κατέχει. Για παράδειγμα, η RA θα μπορούσε να ζητά από τον συνδρομητή μια σε ισχύ ταυτότητα με φωτογραφία, όπως είναι η άδεια οδήγησης και το διαβατήριο.
- Μέσω του συνδρομητή αποκτά το δημόσιο κλειδί του.
- Προμηθεύει το συνδρομητή με το CA δημόσιο κλειδί, το οποίο ο συνδρομητής εμπιστεύεται. Η εμπιστοσύνη αυτή, γενικά, επιτυγχάνεται μέσω της λήψης του δημοσίου κλειδιού από μία έμπιστη πηγή, όπως π.χ. «χέρι με χέρι» ή μέσω Secure Sockets Layer (SSL) από έναν έμπιστο ή γνωστό δικτυακό τόπο.
- Στέλνει το αίτημα για τη δημιουργία του πιστοποιητικού στην CA. Τυπικά, η RA δημιουργεί ένα μήνυμα ηλεκτρονικού ταχυδρομείου, το οποίο περιέχει το όνομα του συνδρομητή και το δημόσιο κλειδί του, το υπογράφει ψηφιακά και το στέλνει στην CA. Άλλα μέσα μεταφοράς (π.χ. Διαδίκτυο) είναι επίσης κατάλληλα, με την προϋπόθεση να παραμένουν η ταυτότητα του συνδρομητή και το δημόσιο κλειδί του αναλλοίωτα [2].

4.3.5 Πολιτική του ψηφιακού πιστοποιητικού

Για να διασφαλιστεί η ασφάλεια της Υποδομής Δημοσίου Κλειδιού, οι συνιστώσες της είναι απαραίτητο να λειτουργούν με υψηλό βαθμό ασφάλειας. Για παράδειγμα:

- Τα ιδιωτικά κλειδιά πρέπει να παραμένουν εμπιστευτικά.
- Τα ιδιωτικά κλειδιά πρέπει να χρησιμοποιούνται μόνο από τους ιδιοκτήτες των κλειδιών.
- Πρέπει να διασφαλίζεται η ακεραιότητα του δημοσίου κλειδιού.
- Η αρχική ταυτοποίηση του συνδρομητή (δηλ. η κατοχή του ιδιωτικού κλειδιού και το θέμα του πιστοποιητικού δημοσίου κλειδιού) πρέπει να εξασφαλίζει ότι δεν πραγματοποιήθηκε υποκλοπή ταυτότητας κατά την δημιουργία του πιστοποιητικού.

Τα συστήματα υπολογιστών των CA και RA πρέπει να προστατεύονται από τυχόν αλλοιώσεις. Επιπρόσθετα με τις απαιτήσεις ασφαλείας, με σκοπό να διευκολυνθεί το

ηλεκτρονικό εμπόριο, η Υποδομή Δημοσίου Κλειδιού πρέπει να απευθύνει υποχρεώσεις προς όλες τις μονάδες συνεργασίας και ευθύνες σε περίπτωση διαφωνίας. Αυτά τα θέματα ασφάλειας, ευθυνών και υποχρεώσεων υλοποιούνται μέσω μιας *πολιτικής πιστοποιητικών* (certificate policy, CP). Σύμφωνα με το στάνταρτ X.509 του American National Standards Institute (ANSI), ένα πιστοποιητικό είναι ένα «σετ κανόνων» που καταδεικνύει την «προσαρμοστικότητα» ενός πιστοποιητικού στα πλαίσια μιας συγκεκριμένης κοινότητας ή/ και μιας κλάσης εφαρμογών με κοινές απαιτήσεις ασφαλείας. Ο χρήστης του πιστοποιητικού μπορεί να χρησιμοποιήσει μια πολιτική πιστοποιητικών για να αποφασίσει εάν ένα πιστοποιητικό, ή η σύνδεση που πραγματοποιείται ανάμεσα στο πιστοποιητικό και τον ιδιοκτήτη του, είναι επαρκώς άξιο εμπιστοσύνης για μια συγκεκριμένη εφαρμογή. Η CP απευθύνει απαιτήσεις ασφαλείας και υποχρεώσεις προς όλες τις συνιστώσες της Υποδομής Δημοσίου Κλειδιού, και όχι μόνο προς την CA: αυτές περιλαμβάνουν τις CA και RA, καταχωρητές, συνδρομητές και έμπιστους συμμετέχοντες. Μια πιο λεπτομερής περιγραφή των πρακτικών που ακολουθεί η CA κατά την έκδοση και τη διαχείριση των πιστοποιητικών, περιέχεται σε μια *δήλωση πρακτικών πιστοποίησης*(certification practice statement, CPS), η οποία εκδίδεται από την CA. Αν και μια CP και μια CPS καλύπτουν τα ίδια πεδία, η Πολιτική Πιστοποιητικών προσδιορίζει τις απαιτήσεις ασφαλείας και τις υποχρεώσεις για μια Υποδομή Δημόσιου Κλειδιού και η CPS περιγράφει πώς αυτές οι απαιτήσεις ικανοποιούνται από την Πολιτική Δημοσίου Κλειδιού. Επίσης όμως, χρησιμοποιούνται διαφορετικά. Η CP αποτελεί τη βάση για την πιστοποίηση μέσα στα όρια της Υποδομής Δημοσίου Κλειδιού, με σκοπό να προωθήσει το ασφαλές ηλεκτρονικό εμπόριο. Ένα *αντικείμενο αναγνώρισης* (*Object Identifier, OID*) που αναπαριστά την CP και χρησιμοποιείται για να δημιουργεί ένα πιστοποιητικό, μπορεί να συμπεριληφθεί στις «πολιτικές πιστοποιητικών» των πιστοποιητικών X.509. Το OID, έτσι, επιτρέπει στους έμπιστους συμμετέχοντες να ενημερώνονται για την προσοχή που λήφθηκε κατά την δημιουργία των πιστοποιητικών, την προτεινόμενη χρήση και τις υποχρεώσεις των διαφόρων συμμετεχόντων.

Η CPS επιτρέπει στην Υποδομή Δημοσίου Κλειδιού να χρησιμοποιεί και να διαχειρίζεται τις διάφορες συνιστώσες της. Ακόμα αποτελεί τη βάση όπου πραγματοποιούνται οι έλεγχοι για το αν οι συνιστώσες της Υποδομής Δημοσίου Κλειδιού λειτουργούν σύμφωνα με τις προδιαγραφές της CPS [2].

4.3.6 Παγκόσμια υποδομή δημόσιου κλειδιού

Οι αρχές μιας Υποδομής Δημοσίου Κλειδιού με μία CA μπορούν να επεκταθούν ώστε να υποστηρίζουν, για παράδειγμα παγκόσμιο, ασφαλές ηλεκτρονικό εμπόριο, μέσω πολλαπλών CA ή/ και CA οι οποίες πιστοποιούν άλλες CA και η μία την άλλη. Ο τρόπος με τον οποίο οι CA πιστοποιούν η μία την άλλη ονομάζεται *μοντέλο εμπιστοσύνης, γράφος εμπιστοσύνης ή αρχιτεκτονική της Υποδομής Δημοσίου Κλειδιού*. Για να επικοινωνήσει ένα άτομο με ασφάλεια με κάποιο άλλο άτομο, πρέπει να υπάρχει ένα μονοπάτι εμπιστοσύνης, το οποίο να ξεκινά από τον έμπιστο συμμετέχοντα (trust anchor of the relying party) και να καταλήγει στον συνδρομητή του οποίου η υπογραφή απαιτείται να διασταυρωθεί ή σε εκείνον που πρόκειται να σταλεί κάποιο κρυπτογραφημένο μήνυμα. Ασφαλείς επικοινωνίες σε παγκόσμιο επίπεδο απαιτούν την ύπαρξη ενός μονοπατιού εμπιστοσύνης από κάθε συνδρομητή προς κάθε άλλο συνδρομητή.

Η έμπιστη τρίτη πλευρά ελέγχει την ισχύ των πιστοποιητικών. Μετά από αυτό, τα δημόσια κλειδιά είναι άξια εμπιστοσύνης και χρησιμοποιούνται για να ελεγχθούν τα πιστοποιητικά που εκδίδονται από την CA. Στη συνέχεια το δημόσιο κλειδί μπορεί να χρησιμοποιηθεί για την διασταύρωση ψηφιακών υπογραφών και για κρυπτογράφηση.

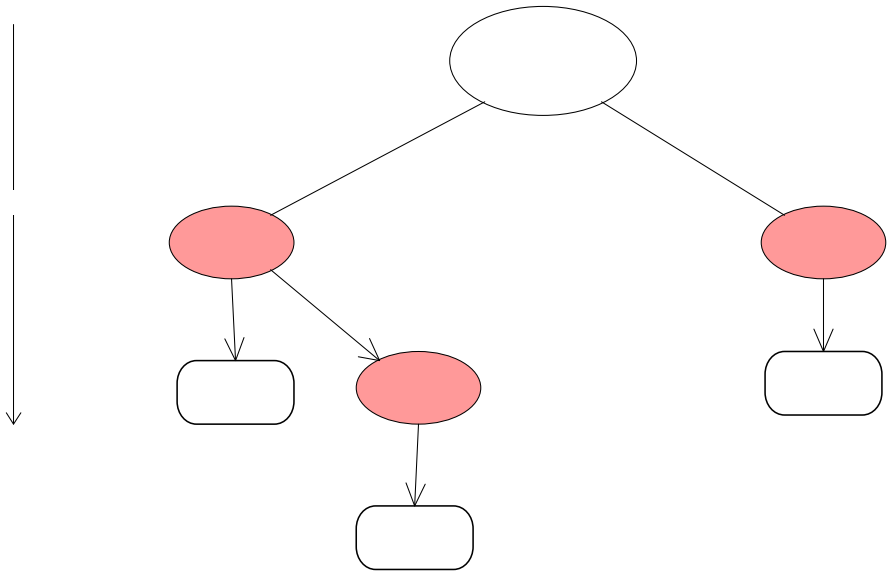
Έχουν αναπτυχθεί τρεις διαφορετικές αρχιτεκτονικές, η υποδομή δημοσίου κλειδιού (PKI) με μία CA, το ιεραρχικό μοντέλο CA και το μικτό μοντέλο CA.

4.3.6.1 Υποδομή Δημοσίου Κλειδιού με μόνο μία CA

Σε αυτήν την αρχιτεκτονική θεωρούμε ότι μόνο μία CA έχει εκδώσει τα πιστοποιητικά δύο οντοτήτων A και B. Συνεπώς για να επικοινωνήσει με ασφάλεια ο A με τον B, αρκεί να έχει το δημόσιο κλειδί της CA, την οποία εμπιστεύεται. Κατ' αυτόν τον τρόπο μεταβατικά η εμπιστοσύνη αυτή μεταφέρεται από την CA στον B. Αυτή η θεώρηση είναι εφαρμόσιμη για μικρό αριθμό χρηστών, διότι όσο μεγαλύτερη είναι η ομάδα χρηστών που εξυπηρετείται από την CA, τόσο δυσκολότερο είναι γι' αυτήν να υποστηρίξει τεχνικά τις λειτουργίες της PKI. Φυσικό επακόλουθο των παραπάνω είναι η δημιουργία πολλών CA, οι οποίες εξυπηρετούν πολλούς χρήστες και η ανάπτυξη σχέσεων εμπιστοσύνης ανάμεσά τους. Ο συσχετισμός των CAs μπορεί να γίνει με δύο βασικούς τρόπους: είτε αναπτύσσοντας σχέση προϊσταμένου-υφισταμένου, είτε αναπτύσσοντας σχέση ομότιμων. Κάθε μέθοδος παρουσιάζει μειονεκτήματα και πλεονεκτήματα, στην πραγματικότητα όμως και οι δύο παραπάνω συνδυάζονται για την κατασκευή σύνθετων δομών PKI.

4.3.6.2 Ιεραρχικό μοντέλο Υποδομής Δημοσίου Κλειδιού

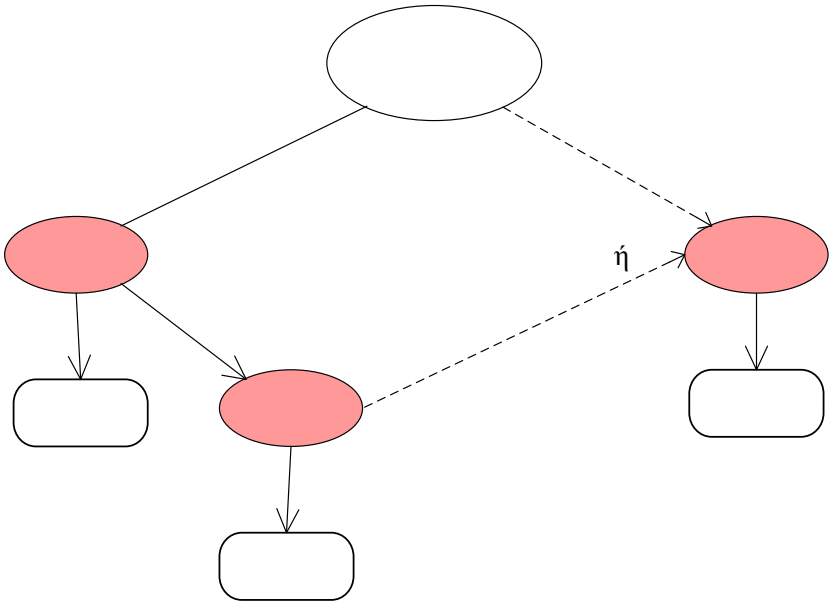
Πρόκειται για μία PKI με σχέση προϊσταμένου- υφισταμένου ανάμεσα στις CAs της. Σ' αυτήν τη μεθοδολογία, όλοι οι χρήστες εμπιστεύονται την ίδια ρίζα CA. Κατ' αυτόν τον τρόπο όλοι οι χρήστες ξεκινούν τη διαδρομή των πιστοποιητικών τους με το δημόσιο κλειδί της ρίζας CA. Γενικά η ρίζα CA δεν εκδίδει πιστοποιητικά σε μεμονωμένους χρήστες, αλλά σε υφιστάμενες CAs, οι οποίες μπορούν να εκδίδουν πιστοποιητικά σε χρήστες ή άλλες CAs, που είναι υφιστάμενες αυτών. Συνεπώς στο ιεραρχικό μοντέλο PKI η σχέση εμπιστοσύνης είναι προς μία κατεύθυνση μόνο, από την ρίζα CA προς τις υφιστάμενές της, όπως παρουσιάζεται και στο παρακάτω σχήμα.



Σχήμα 4.4 Ιεραρχικό μοντέλο PKI

Τα βασικά πλεονεκτήματα του ιεραρχικού μοντέλου είναι η εύκολη επεκτασιμότητά του, το ότι οι διαδρομές από το πιστοποιητικό έως τη ρίζα είναι εύκολα αναγνωρίσιμες και μικρές και τα πιστοποιητικά είναι μικρότερα κι απλούστερα.

Τα μειονεκτήματα από την άλλη εντοπίζονται στο ότι όλα εξαρτώνται από ένα σημείο αναφοράς (ρίζα CA), ο προσδιορισμός του οποίου, στο σημερινό ανταγωνιστικό εταιρικό περιβάλλον, μπορεί να αποτελέσει αντικείμενο διαφωνίας. Η μετάβαση από ένα μοντέλο απομονωμένης CA σε ένα μοντέλο ιεραρχικής αρχιτεκτονικής PKI μπορεί να είναι, τέλος, πρακτικά ή οικονομικά ασύμφορη.



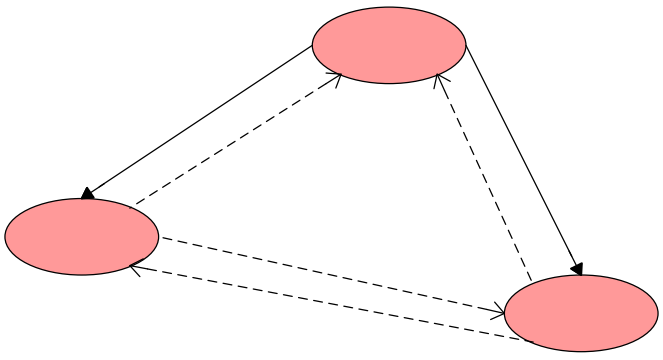
Σχήμα 4.5 Προσθήκη νέας CA σε ιεραρχικό μοντέλο PKI.

4.3.6.3 Μικτό μοντέλο Υποδομής Δημοσίου Κλειδιού

Είναι δομημένο με σχέση ομότιμη προς ομότιμη ανάμεσα στις CAs και λέγεται δίκτυο PKI ή δίκτυο εμπιστοσύνης PKI. Σε αυτό το μοντέλο, όλες οι CAs θεωρούνται σημείο εμπιστοσύνης, εκδίδουν πιστοποιητικά για τους χρήστες τους, αλλά και μεταξύ τους, στα οποία περιγράφουν την αμφίδρομη σχέση εμπιστοσύνης ανάμεσά τους.

Πλεονέκτημα αυτής της δομής αποτελεί το ότι εύκολα προστίθενται νέες CAs, ενώ τα πολλά σημεία εμπιστοσύνης διασφαλίζουν την ακεραιότητά της, ακόμα και σε περίπτωση που μία CA κριθεί αναξιόπιστη.

Ωστόσο, η αμφίδρομη δόμηση των σχέσεων εμπιστοσύνης έχει ως συνέπεια το ότι οι διαδρομές δεν είναι μοναδικές, καθώς είναι δυνατή η παρακολούθηση περισσότερων από μίας διαδρομών, που κάποιες από αυτές οδηγούν σε έγκυρες διαδρομές, ενώ άλλες οδηγούν σε αδιέξοδα. Τέλος η επεκτασιμότητα, εκτός από μεγάλο πλεονέκτημα, αποτελεί και τεράστιο μειονέκτημα, διότι το μέγιστο μήκος μιας διαδρομής είναι θεωρητικά ίσο με το πλήθος των CAs που υπάρχουν στο μοντέλο [1].



Σχήμα 4.6 Διάγραμμα των σχέσεων ανάμεσα στις CA ενός δικτύου PKI.

4.4 Αλγόριθμοι κρυπτογράφησης

4.4.1 Γενική επισκόπηση

Στην παράγραφο αυτή παρουσιάζονται συνοπτικά οι κυριότεροι αλγόριθμοι που χρησιμοποιούνται σήμερα στην κρυπτογραφία, με έμφαση στις δύο βασικές οικογένειες αλγορίθμων: τους *συμμετρικούς αλγόριθμους,* που χρησιμοποιούν το ίδιο κλειδί για κρυπτογράφηση και αποκρυπτογράφηση και είναι πιο γρήγοροι από τους ασύμμετρους και τους *αλγόριθμους ψηφιακών υπογραφών,* για τη δημιουργία και επαλήθευση των οποίων χρησιμοποιούνται ασύμμετροι αλγόριθμοι σε συνδυασμό με αλγόριθμους κατακερματισμού.

4.4.1.1 Συμμετρικοί αλγόριθμοι

Το πρότυπο DES (Data Encryption Standard) [3], το Τριπλό DES (Triple DES) [4] και ο αλγόριθμος IDEA(International Data Encryption Algorithm) [5] είναι συμμετρικά συστήματα τμηματικής (block) κρυπτογράφησης, που χρησιμοποιούν «κομμάτια» δεδομένων των 64 bits. Ο αλγόριθμος DES επιλέχθηκε ως επίσημο Ομοσπονδιακό Πρότυπο Επεξεργασίας Πληροφορίας (Federal Information Processing Standard, FIPS) των ΗΠΑ το 1976 και συνεπώς υιοθετήθηκε διεθνώς. Καθώς το μικρό κλειδί του (56 bits) καθιστά τον DES μη ασφαλή, ο Τριπλός DES θεωρήθηκε ως ένας απλός τρόπος αύξησης του μεγέθους κλειδιού χωρίς να απαιτείται η μετάβαση σε νέο αλγόριθμο. Ο τελευταίος διενεργεί, στην ουσία, τρεις διαδοχικές εκτελέσεις του DES, με αποτέλεσμα ένα κλειδί μήκους 192 bits (συμπεριλαμβανομένων των bits ισοτιμίας). Ο IDEA ήταν από τις πρώτες προτάσεις αντικατάστασης του DES με κλειδί μήκους 128 bits. Χρησιμοποιήθηκε στο πρόγραμμα PGP (Pretty Good Privacy) [6] σαν εναλλακτική και θεωρείται αρκετά ασφαλής.

Ο αλγόριθμος AES (Advanced Encryption Standard) [7] είναι από τους δημοφιλέστερους αλγόριθμους στη συμμετρική κρυπτογραφία. Θεωρείται πολύ ασφαλής και η Επιτροπή Εθνικής Ασφάλειας (National Security Agency, NSA) των ΗΠΑ τον συμπεριλαμβάνει στην «Κατηγορία Β» των κρυπτογραφικών αλγορίθμων που προτείνονται ως η βάση για την προστασία τόσο μη απόρρητων όσο και πολύ απόρρητων πληροφοριών [8]. Ο AES έχει σταθερό μέγεθος block 128 bits και μέγεθος κλειδιού 128, 192 ή 256 bits.

Ο Blowfish [9] αποτελεί έναν ακόμα αλγόριθμο με σταθερό μέγεθος block 64 bits. Χρησιμοποιείται σε μεγάλο αριθμό εφαρμογών και προϊόντων κρυπτογραφίας και θεωρείται αρκετά ασφαλής. Το μέγεθος κλειδιού του μπορεί να μεταβάλλεται και φτάνει τα 448 bits. Στην πράξη το συνηθέστερο μήκος κλειδιού είναι 128 bits.

Οι αλγόριθμοι Rivest Ciphers 2 (RC2) [10] και 5 (RC5) [11] σχεδιάστηκαν από τον Ronald Rivest. Ο RC2 είναι ένας αλγόριθμος τμηματικής κρυπτογράφησης με μήκος κλειδιού που μεταβάλλεται από 8 έως 128 bits και μήκος block 64 bits, ενώ ο RC5 έχει μεταβλητό μήκος κλειδιού μέχρι τα 2048 bits αλλά επιπλέον, σε αντίθεση με την πλειοψηφία των σχημάτων, μεταβλητό μέγεθος block.

Ο Rivest Cipher 4 (RC4) [12] του ίδιου σχεδιαστή είναι ένας συμμετρικός αλγόριθμος με κυμαινόμενο μέγεθος κλειδιού. Σε αντίθεση με τους δύο προηγούμενους είναι κρυπτογράφησης ροής (stream cipher) και χρησιμοποιείται σε δημοφιλή πρωτόκολλα, όπως το SSL (Secure Socket Layer) [13].

4.4.1.2 Ψηφιακές υπογραφές

Οι αλγόριθμοι σύνοψης μηνύματος Message Digest Algorithms 2 (MD2) [14] και 5 (MD5) [15] συνιστούν κρυπτογραφικές συναρτήσεις κατακερματισμού (hash functions) και αναπτύχθηκαν από τον Ronald Rivest. Παράγουν και οι δύο μια τιμή κατακερματισμού μήκους 128 bits.Κατά μία έννοια, ο MD5 αντικατέστησε τον MD2, παρόλο που και ο τελευταίος χρησιμοποιείται ακόμα σε μεγάλο βαθμό. Εφαρμόζονται κυρίως σε Υποδομές Δημόσιου Κλειδιού ως μέρος των ψηφιακών πιστοποιητικών, συνήθως με τον ασύμμετρο κρυπτογραφικό αλγόριθμο RSA.

Ο αλγόριθμος RSA (Ronald Rivest, Adi Shamir, Leonard Adleman) [16] αναπτύχθηκε το 1977 και είναι ένα από τα πρώτα κρυπτογραφικά σχήματα δημόσιου κλειδιού, ενώ σήμερα

αποτελεί το ευρύτερα αποδεκτό και υλοποιούμενο. Τα κλειδιά RSA έχουν συνηθέστερα μήκος 1024-2048 bits.

Ο αλγόριθμος ασφαλούς κατακερματισμού Secure Hash Algorithm 1 (SHA-1) [17] θεωρείται ο διάδοχος του MD5. Παράγει τιμή κατακερματισμού μήκους 160 bits, οπότε θεωρείται ασφαλέστερος των MD2 και MD5.

Ο αλγόριθμος DSA (Digital Signature Algorithm) [18] προτάθηκε από το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (National Institute of Standards and Technology, NIST) τον Αύγουστο του 1991 προκειμένου να χρησιμοποιηθεί στο Πρότυπο Ψηφιακής Υπογραφής (Digital Signature Standard, DSS) [18]. Σύμφωνα με το DSS, ο DSA χρησιμοποιείται σε συνδυασμό με τον SHA-1 για τη δημιουργία και επαλήθευση ψηφιακών υπογραφών. Ο DSA χρησιμοποιεί για τη δημιουργία ιδιωτικών και δημόσιων κλειδιών έναν πρώτο αριθμό (κλειδί), συνήθως μήκους 512-1024 bits.

Ο αλγόριθμος ECDSA (Elliptic Curve Digital Signature Algorithm) είναι το ανάλογο ελλειπτικής καμπύλης του DSA και παρέχει μικρότερα μεγέθη κλειδιών για το ίδιο επίπεδο ασφάλειας. Περιλαμβάνεται και αυτός στην «Κατηγορία Β» της NSA [8], που σημαίνει ότι ο ECDSA, χρησιμοποιώντας την πρωταρχική ελλειπτική καμπύλη που προκύπτει από 256 bits με SHA-256, ενδείκνυται για την προστασία απόρρητων πληροφοριών.

Η οικογένεια SHA περιλαμβάνει, εκτός του SHA-1, τους αλγόριθμους SHA-224, SHA-256, SHA-384 και SHA-512 [17], που δίνουν τιμές κατακερματισμού μήκους 224, 256, 384 και 512 bits αντίστοιχα, και αναφέρονται συχνά αθροιστικά ως SHA-2.

4.4.2 Μελέτη απόδοσης

Α. Δημιουργία κλειδιού/ών

Ο Πίνακας 4.1 παρουσιάζει τα συγκεντρωμένα από προσομοιώσεις αποτελέσματα, αναφορικά με τους χρόνους εκτέλεσης των διαδικασιών δημιουργίας κλειδιού/ών. Όπου ήταν δυνατόν, οι αλγόριθμοι δοκιμάστηκαν με διαφορετικά μεγέθη κλειδιών. Για την δημιουργία του πίνακα, λήφθηκαν υπ' όψιν οι εκτελέσεις κάθε αλγορίθμου. Για παράδειγμα, από τη στιγμή που ο αλγόριθμος RSA με μέγεθος κλειδιού 2048 bits δοκιμάστηκε σε συνδυασμό με διάφορες συναρτήσεις κατακερματισμού, συμπεριλήφθηκαν όλες αυτές οι εκτελέσεις. Έτσι, ο Πίνακας 4.1 παρέχει τον αριθμό των εκτελέσεων.

Οι μετρήσεις στον Πίνακα 4.1 αφορούν την ελάχιστο, την μέγιστη και τη μέση παρατηρούμενη χρονική διάρκεια όλων των εκτελέσεων [19].

Αλγόριθμος	Μέγεθος Κλειδιού(bits)	Αριθμός εκτελέσεων	Ελάχιστος χρόνος(msecs)	Μέγιστος χρόνος (msecs)	Μέσος χρόνος(msecs)
DES	56	100	343	1125	398
T - DES	192	100	343	453	376
AES	128	100	359	562	411
AES	192	100	359	469	389
AES	256	100	359	750	394
Blowfish	56	100	360	500	398
Blowfish	112	100	359	1047	415
Blowfish	224	100	359	547	389
Blowfish	336	100	359	422	389
Blowfish	448	100	344	406	377
IDEA	128	100	562	797	611
RC2	64	100	359	687	390

RC2	96	100	359	594	388
RC2	128	100	343	421	371
RC4	1024	100	328	453	358
RC4	128	100	328	657	356
RC4	512	100	328	437	355
RC5	128	100	563	625	597
RC5	512	100	578	750	620
RC5	1024	100	563	703	627
RC5	2048	100	547	719	613
DSA	512	100	78	485	99
DSA	768	100	78	141	98
DSA	1024	100	78	156	99
RSA	1024	600	250	2328	588
RSA	1536	600	453	8609	1805
RSA	2048	600	672	23157	4770
ECDSA	192	400	203	813	247
ECDSA	239	400	218	516	265
ECDSA	256	400	218	485	266

Πίνακας 4.1 Χρόνοι καθυστέρησης κατά την δημιουργία κλειδιών

Β. Κρυπτογράφηση και Αποκρυπτογράφηση

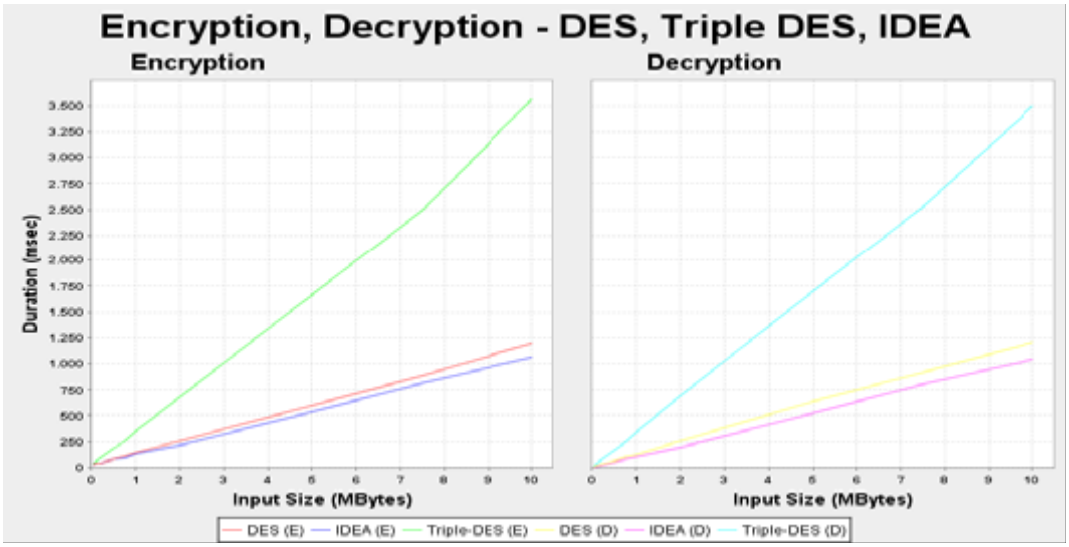
Τα Σχήματα 4.7 – 4.11 απεικονίζουν τα αποτελέσματα εκτέλεσης κρυπτογράφησης και αποκρυπτογράφησης χρησιμοποιώντας συμμετρικούς αλγόριθμους, Όπως δείχνει το Σχήμα 4.7, οι αλγόριθμοι DES, Triple DES and IDEA μπορούν να χαρακτηριστούν σχετικά αργοί. Ειδικά οι αλγόριθμοι DES και Triple DES είναι οι αλγόριθμοι με τη μεγαλύτερη καθυστέρηση εκτέλεσης από τους αλγόριθμους που δοκιμάστηκαν, χωρίς να υπάρχει κάποιο κέρδος σε ασφάλεια, το οποίο να δικαιολογεί αυτήν την καθυστέρηση

Στο Σχήμα 4.8 παρατηρείται ότι ο αλγόριθμος AES είναι σημαντικά ταχύτερος στη διενέργεια κρυπτογράφησης και αποκρυπτογράφησης σε σχέση με τους αλγόριθμους DES, Triple DES και IDEA. Συγκεκριμένα, η αύξηση του μεγέθους του κλειδιού προκαλεί μόνο μια μικρή- αν και υπολογίσιμη- καθυστέρηση εκτέλεσης, ακόμα και για μεγάλα αρχεία, ενώ οι χρόνοι δημιουργίας κλειδιών παραμένει περίπου οι ίδιοι.

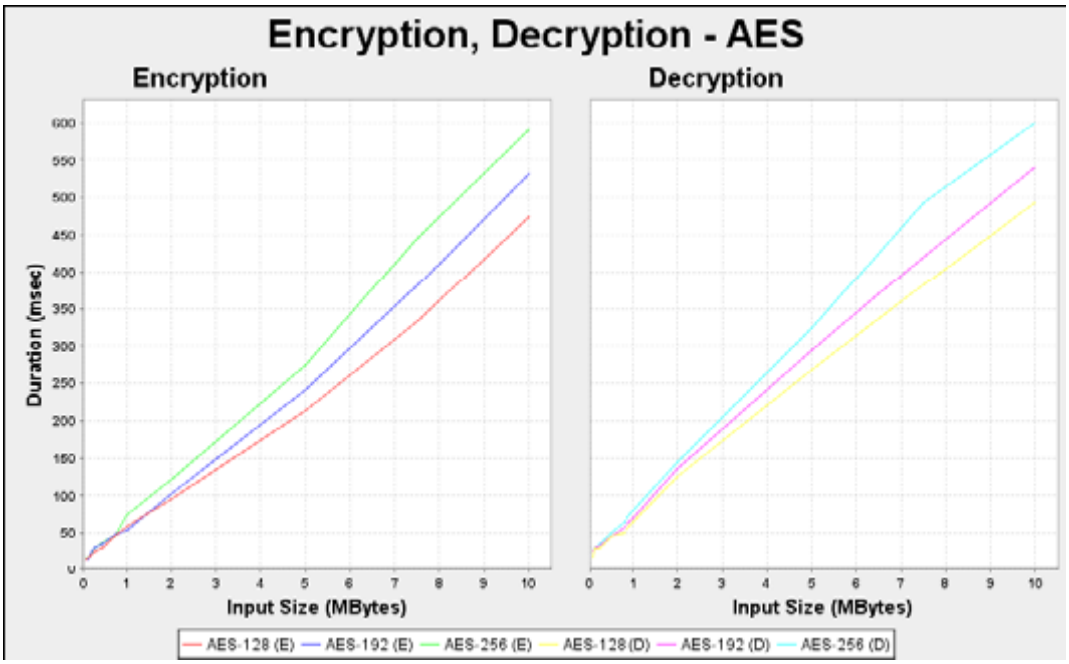
Ο αλγόριθμος Blowfish (Σχήμα 4.9) είναι ένας σχετικά ταχύς αλγόριθμος, με το μέγεθος του κλειδιού να μην παίζει σημαντικό ρόλο κατά την εκτέλεσή του. Οι χρόνοι δημιουργίας των κλειδιών είναι συγκρίσιμοι με εκείνους του AES αλγορίθμου.

Όπως απεικονίζεται στο Σχήμα 4.10, ο RC5 αποτελεί πράγματι μια βελτιωμένη έκδοση του RC2, σε ό,τι αφορά την καθυστέρηση εκτέλεσης. Το μέγεθος του κλειδιού δείχνει να έχει ασήμαντη επίδραση πάνω στην εκτέλεση και των δύο αλγορίθμων, ενώ τα κλειδιά του RC5 χρειάζονται σημαντικά περισσότερο χρόνο για να δημιουργηθούν.

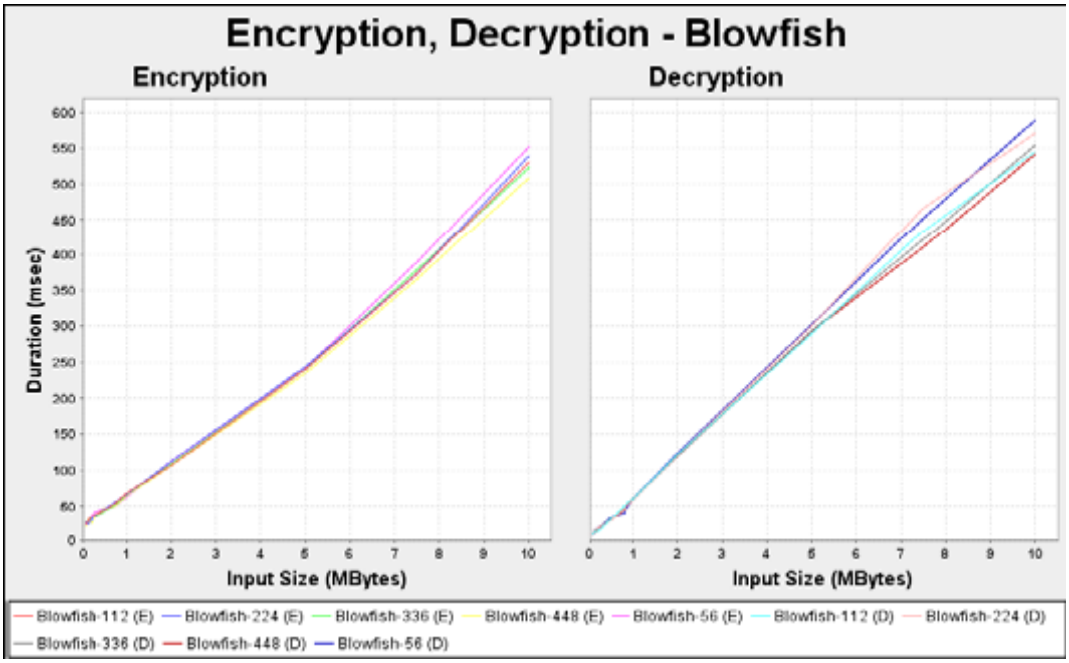
Τέλος, από το Σχήμα 4.11 συμπεραίνουμε ότι ο RC4 είναι εξαιρετικά ταχύς, γεγονός αναμενόμενο αφού πρόκειται για stream cipher αλγόριθμο. Το μέγεθος του κλειδιού δε φαίνεται να έχει κάποια επίδραση στην εκτέλεση του, ενώ οι χρόνοι δημιουργίας των κλειδιών μετρήθηκαν προσεγγιστικά ίσοι [19].



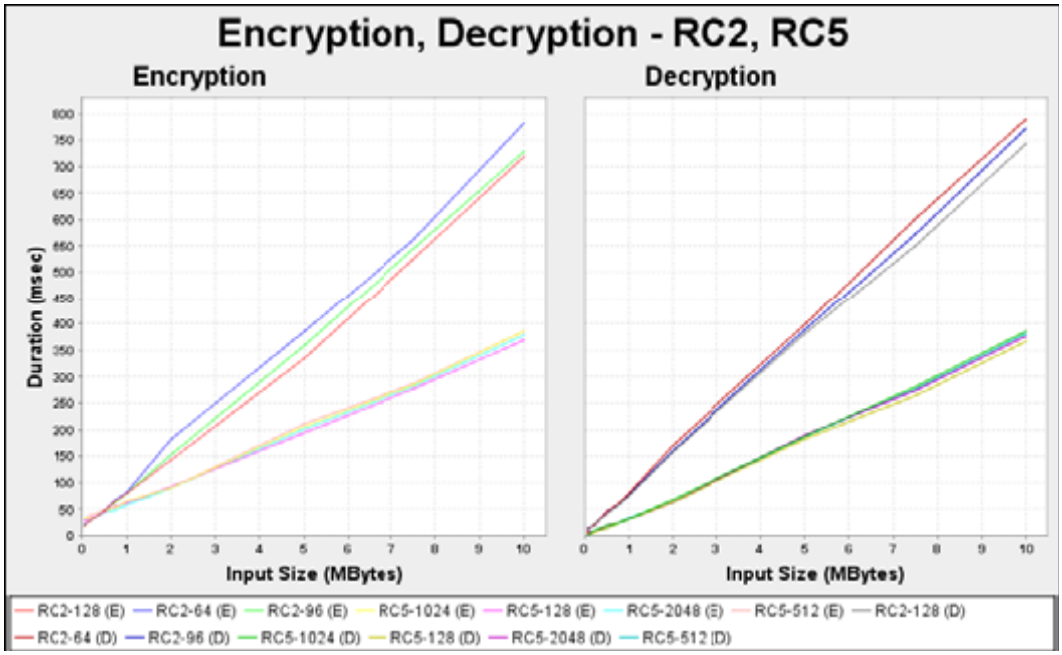
Σχήμα 4.7 Κρυπτογράφηση/αποκρυπτογράφηση με αλγόριθμους DES, Triple DES και IDEA



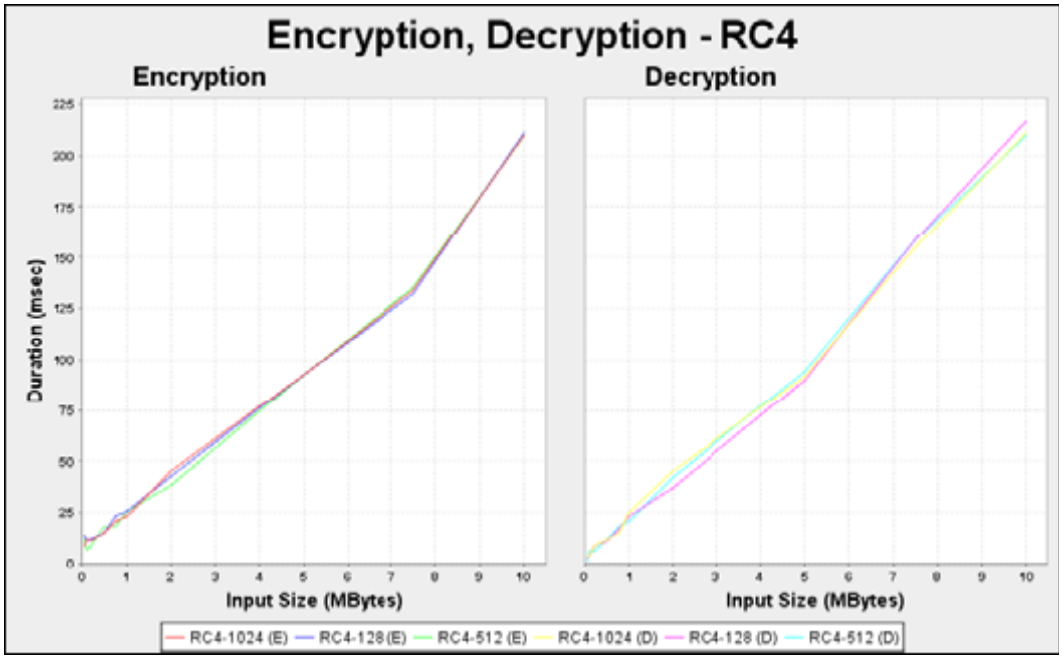
Σχήμα 4.8 Κρυπτογράφηση/αποκρυπτογράφηση με αλγόριθμο AES



Σχήμα 4.9 Κρυπτογράφηση/αποκρυπτογράφηση με αλγόριθμο Blowfish



Σχήμα 4.10 Κρυπτογράφηση/αποκρυπτογράφηση με αλγόριθμους RC2 και RC5



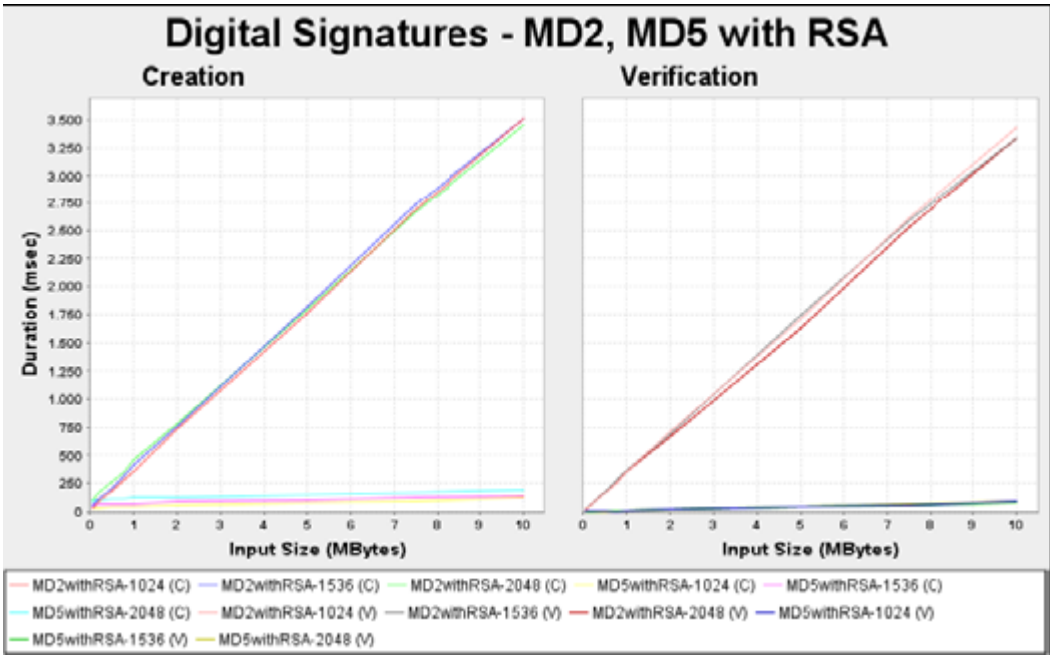
Σχήμα 4.11 Κρυπτογράφηση/αποκρυπτογράφηση με αλγόριθμο RC4

Γ. Δημιουργία Ψηφιακών Υπογραφών και Επαλήθευση

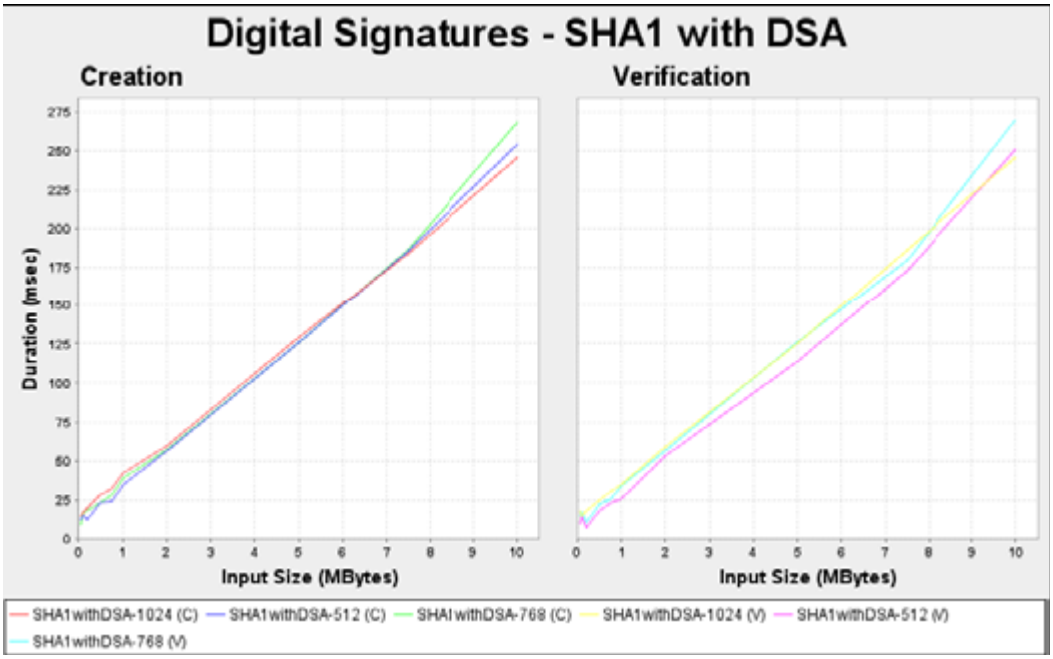
Τα Σχήματα 4.12 – 4.16 απεικονίζουν τα αποτελέσματα εκτέλεσης για την δημιουργία και επαλήθευση ψηφιακών υπογραφών. Ξεκινώντας από τους αλγόριθμους MD2 και MD5 με RSA (Σχήμα 4.12), ο MD5 είναι σίγουρα ταχύτερος από τον MD2. Επιπλέον, το μέγεθος του RSA κλειδιού έχει μικρή επίδραση στην εκτέλεση, σε ό,τι αφορά τους χρόνους δημιουργίας και επαλήθευσης των ψηφιακών υπογραφών. Ωστόσο, επηρεάζει σημαντικά το χρόνο που χρειάζεται για την δημιουργία του ίδιου του κλειδιού.

Σε ό,τι αφορά τις συναρτήσεις κατακερματισμού SHA με RSA και ECDSA (Σχήματα 4.14 – 4.16), μια πρώτη παρατήρηση είναι, όπως είναι αναμενόμενο, ότι όσο μεγαλύτερο είναι το αποτέλεσμα της συνάρτησης κατακερματισμού τόσο περισσότερος χρόνος χρειάζεται για να ολοκληρωθούν οι διαδικασίες δημιουργίας και επαλήθευσης των ψηφιακών υπογραφών. Γενικά ο RSA αλγόριθμος είναι ταχύτερος από τον ECDSA, ειδικά στο στάδιο της επαλήθευσης των ψηφιακών υπογραφών. Αυτό παρατηρείται εντονότερα για μικρές εισόδους, όπου ο χρόνος εκτέλεσης της συνάρτησης κατακερματισμού είναι μικρότερος και το τελικό αποτέλεσμα εξαρτάται από την εκτέλεση του αλγορίθμου που πραγματοποιεί την υπογραφή. Ωστόσο, ο RSA χρειάζεται περισσότερο χρόνο για την δημιουργία των κλειδιών.

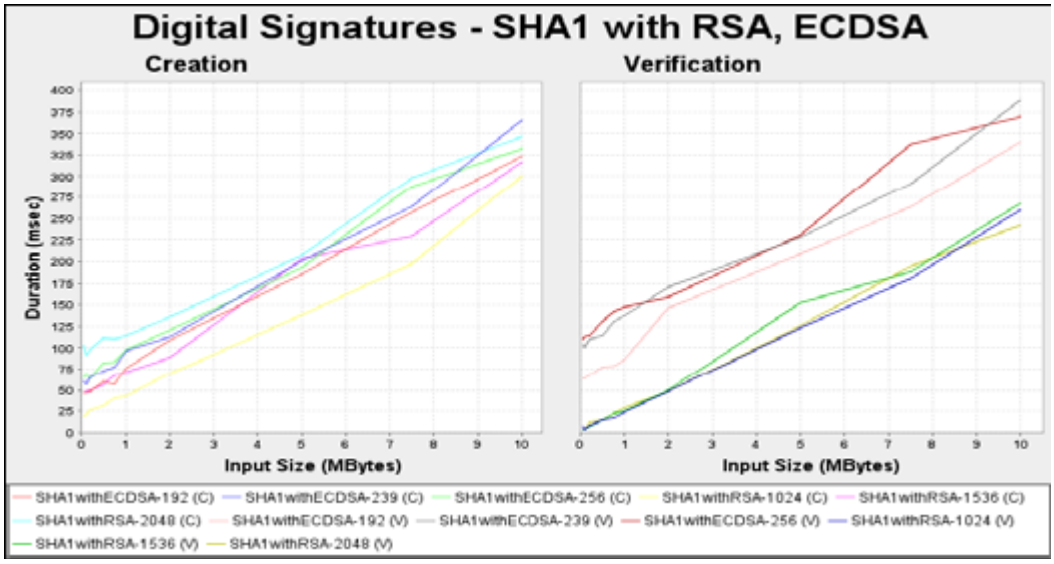
Τέλος, ο DSA (Σχήμα 4.13) δείχνει να είναι λίγο ταχύτερος όταν χρησιμοποιείται με SHA-1 συγκρινόμενος με τους RSA και ECDSA, ενώ και τα ζεύγη των κλειδιών δημιουργούνται πιο γρήγορα. Βέβαια, αυτές οι διαφορές δεν αντισταθμίζουν τα μειονεκτήματα του DSA σε ό,τι αφορά την ασφάλεια [19].



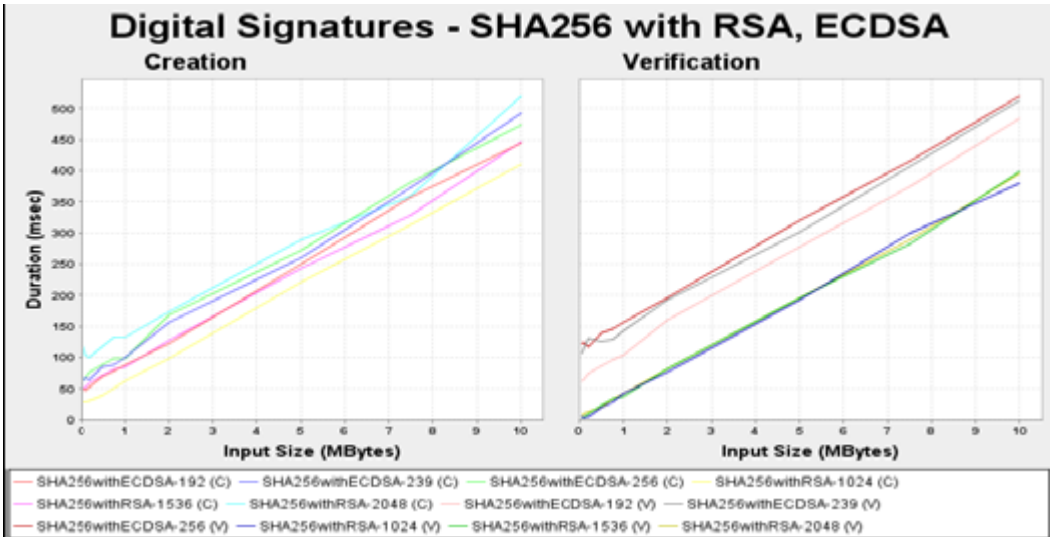
Σχήμα 4.12 Δημιουργία και επαλήθευση ψηφιακών υπογραφών, MD2 και MD5 με RSA



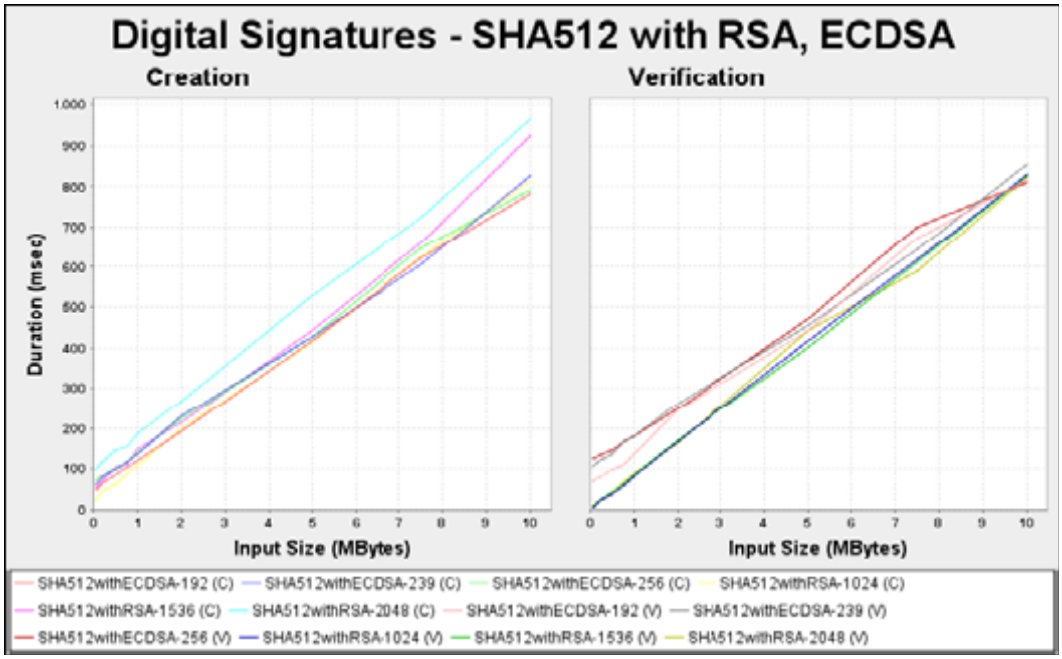
Σχήμα 4.13 Δημιουργία και επαλήθευση ψηφιακών υπογραφών, SHA1 με DSA



Σχήμα 4.14 Δημιουργία και επαλήθευση ψηφιακών υπογραφών, SHA1 με RSA, ECDSA



Σχήμα 4.15 Δημιουργία και επαλήθευση ψηφιακών υπογραφών, SHA256 με RSA, ECDSA



Σχήμα 4.16 Δημιουργία και επαλήθευση ψηφιακών υπογραφών, SHA512 με RSA, ECDSA.

4.5 Βιβλιογραφία

[1] Ιάκωβος Στ. Βενιέρης, Ευγενία Νικολούζου, "Τεχνολογίες διαδικτύου", Εκδόσεις Τζιόλα, Αθήνα 2003, pp. 89-97.

[2] Seymour Bosworth, M.E. Kabay, "Computer Security Handbook", John Wiley and sons, Inc., fourth edition, USA, 2002, pp. 23.1-23.13.

[3] National Institute of Standards and Technology, "Data Encryption Standard (DES)", Federal Information Processing Standard Publication 46-3, October 1999.

[4] National Institute of Standards and Technology, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher", Special Publication 800-67, May 2004.

[5] Xuejia Lai and James L. Massey, "A Proposal for a New Block Encryption Standard," in Advances in Cryptology - EUROCRYPT '90: Workshop on the Theory and Application of Cryptographic Techniques, Aarhus, Denmark, May 1990, Springer-Verlang, LNCS vol. 473, pp. 389–404, 1991.

[6] P. R. Zimmermann, "The official PGP user's guide", MIT Press, Cambridge, MA, 1995.

[7] National Institute of Standards and Technology, "Advance Encryption Standard (AES)", Federal Information Processing Standard Publication 197, November 2001.

[8] National Security Agency, "Fact Sheet NSA Suite B Cryptography", available: http://www.nsa.gov/ia/industry/crypto_suite_b.cfm.

[9] Bruce Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)", in Proc. of Fast Software Encryption Cambridge Security Workshop 1993, Springer-Verlang, LNCS vol. 809, pp. 191–204, 1994.

[10] Ronald Rivest, "A Description of the RC2(r) Encryption Algorithm", Request For Comments 2268, RFC Editor, 1998.

[11] Robert Baldwin, Ronald Rivest, "The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms", Request For Comments 2040, RFC Editor, 1996.

[12] Ronald Rivest, "The RC4 encryption algorithm", RSA Data Security, Inc., March, 1992.

[13] SSL 3.0 Specification, available: http://wp.netscape.com/eng/ssl3/, 1996.

[14] Burton Kaliski, "The MD2 Message-Digest Algorithm", Request For Comments 1319, RFC Editor, 1992.

[15] Ronald Rivest, "The MD5 Message-Digest Algorithm", Request For Comments 1321, RFC Editor, 1992.

[16] Ronald Rivest, Adi Shamir, Leonard Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, Communications of the ACM, vol . 21 no. 2, pp. 120–126, Feb. 1978.
[17] National Institute of Standards and Technology, “Secure Hash Standard (SHS)”, Federal Information Processing Standard Publication 180-2, Aug. 2002.
[18] National Institute of Standards and Technology, “Digital Signature Standard (DSS)”, Federal Information Processing Standard Publication 186-2, Jan. 2000.
[19] Georgios V. Lioudakis, Nikolaos L. Dellas, Eleftherios A. Koutsoloukas, Dimitra I. Kaklamani, Iakovos S. Venieris, “A Delay Performance Evaluation of Cryptographic Algorithms”, Technical Report, National Technical University of Athens, Athens, Jan. 2007.

5 Εργαλεία υλοποίησης

Στο κεφάλαιο αυτό γίνεται μια ανασκόπηση στις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της συγκεκριμένης εφαρμογής.

5.1 Η γλώσσα Java

Η ραγδαία εξάπλωση του Internet και του WorldWideWeb δημιούργησαν την ανάγκη νέων τρόπων ανάπτυξης και διανομής του λογισμικού. Η γλώσσα προγραμματισμού Java σχεδιάστηκε με σκοπό την ανάπτυξη εφαρμογών που θα τρέχουν σε ετερογενή δικτυακά περιβάλλοντα, ενώ γίνεται σταδιακά όλο και πιο δημοφιλής για τη δημιουργία ποικίλων και επιδεχόμενων κλιμάκωσης, ανεξαρτήτων από πλατφόρμα, λύσεων. Έχει τα ακόλουθα χαρακτηριστικά:

-Αντικειμενοστραφής.
-Δημιουργία ανεξάρτητων εφαρμογών και applets.
-Είναι interpreted γλώσσα.
-Κατανεμημένη (distributed).
-Ασφαλής (secure).
-Είναι multithreaded.
-Υποστηρίζει multimedia εφαρμογές.
-Συμβατότητα με υπάρχοντα πρότυπα.
-Ανεξαρτησία από πλατφόρμα.
-Μεγάλη δυνατότητα κλιμάκωσης.
-Συνέπεια στην απόδοση.

Πέρα από το ότι ενδείκνυται κατά κόρον για την ανάπτυξη δικτυακών εφαρμογών, η πλατφόρμα της Java δίνει ιδιαίτερη έμφαση στον τομέα της ασφάλειας, υποστηρίζοντας ασφάλεια της γλώσσας (language safety), κρυπτογραφία, υποδομή δημόσιου κλειδιού (public key infrastructure, PKI), πιστοποίηση (authentication), ασφαλή επικοινωνία και έλεγχο πρόσβασης (access control). Οι αρχιτεκτονικές JCA (Java Cryptography Architecture) και JCE (Java Cryptography Extension) αποτελούν μείζον κομμάτι της πλατφόρμας, προσφέροντας μια αρχιτεκτονική «παρόχου» και ένα σύνολο API (Application Programming Interfaces) για ψηφιακές υπογραφές, σύνοψη μηνυμάτων (hashes), πιστοποιητικά και επικύρωση πιστοποιητικών, κρυπτογράφηση (symmetric/asymmetric block/stream ciphers), παραγωγή και διαχείριση κλειδιών, γένεση τυχαίων αριθμών και άλλες λειτουργίες, οι οποίες επιτρέπουν στον προγραμματιστή να ενσωματώνει στοιχεία ασφάλειας στις εφαρμογές του. Η συγκεκριμένη αρχιτεκτονική σχεδιάστηκε με βάση τις ακόλουθες αρχές:

1. Ανεξαρτησία υλοποίησης (Implementation Independence)

Οι εκάστοτε εφαρμογές δε χρειάζεται να υλοποιούν αλγόριθμους ασφάλειας. Αντ' αυτού, τους δίνεται η δυνατότητα να ζητούν υπηρεσίες ασφάλειας από την πλατφόρμα της java. Οι υπηρεσίες αυτές υλοποιούνται σε παρόχους (providers), οι οποίοι είναι ενσωματωμένοι στην πλατφόρμα της java μέσω ενός καθορισμένου interface. Μια εφαρμογή μπορεί να βασιστεί σε πολλαπλούς ανεξάρτητους παρόχους προκειμένου να πετύχει την επιθυμητή λειτουργικότητα ασφάλειας.

2. Διαλειτουργικότητα υλοποίησης (Implementation interoperability)

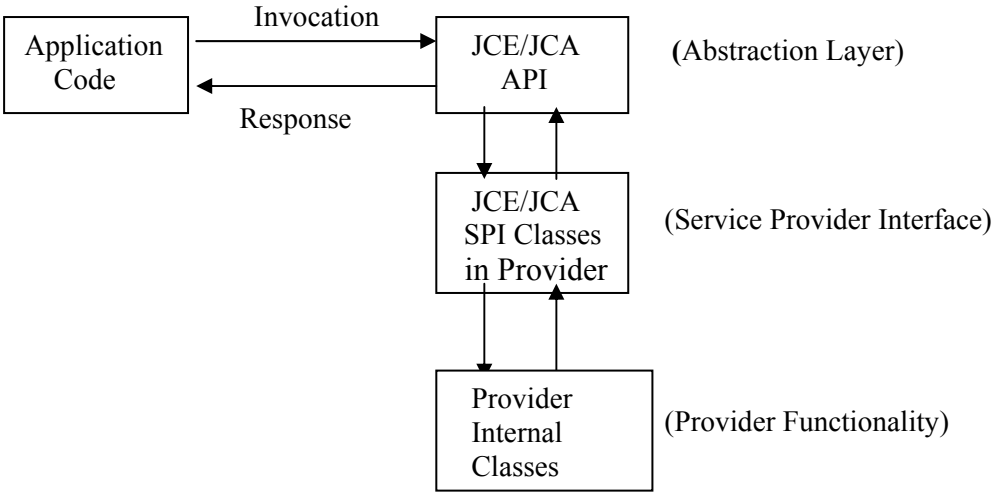
Οι πάροχοι χαρακτηρίζονται από διαλειτουργικότητα σε σχέση με τις εφαρμογές. Συγκεκριμένα, μια εφαρμογή δε συνδέεται με συγκεκριμένο πάροχο, ούτε αντίστροφα.

3.Επεκτασιμότητα κώδικα

Η πλατφόρμα της java περιλαμβάνει έναν αριθμό ενσωματωμένων παρόχων που υλοποιούν ένα βασικό σύνολο υπηρεσιών ασφάλειας, που χρησιμοποιούνται ευρέως σήμερα.

Κάποιες εφαρμογές ωστόσο μπορεί να βασίζονται σε ανακύπτουσες απαιτήσεις που δεν έχουν ακόμα υλοποιηθεί ή σε δικές τους ιδιαίτερες υπηρεσίες. Η πλατφόρμα της java υποστηρίζει την εγκατάσταση επιπλέον παρόχων που υποστηρίζουν τέτοιες υπηρεσίες [1].

Το παρακάτω σχήμα δείχνει τη λειτουργία της αρχιτεκτονικής: ο κώδικας της εφαρμογής καλεί τις κατάλληλες JCA/JCE API κλάσεις. Αυτές με τη σειρά τους καλούν τις κλάσεις στον πάροχο που διαθέτει υλοποιήσεις για τις JCA/JCE κλάσεις της διεπαφής του παρόχου υπηρεσίας (service provider interface, SPI). Οι κλάσεις αυτές, τέλος, καλούν τον εσωτερικό κώδικα στον πάροχο ώστε να διαθέσουν την επιθυμητή λειτουργία [2].



Σχήμα 5.1 JCE/JCA Αρχιτεκτονική.

Εκτός από τα παραπάνω, το JSSE (Java Secure Socket Extension) παρέχουν πρόσβαση σε SSL (Secure Socket Layer) και TLS (Transport Layer Security) υλοποιήσεις. Τα JGSS (Java Generic Security Services) APIs και το SASL (Simple Authentication and Security Layer) μπορούν επίσης να χρησιμοποιηθούν για ασφαλή ανταλλαγή μηνυμάτων μεταξύ εφαρμογών που επικοινωνούν.

Στην παρούσα εργασία χρησιμοποιήθηκε η τελευταία έκδοση Java SE 6 (JDK 1.6.0, JRE 1.6.0), η οποία μεταξύ των διάφορων βελτιώσεων εμφανίζει ενισχυμένα χαρακτηριστικά και ως προς την ασφάλεια.

5.2 Apache Tomcat 5.0

Για την ανάπτυξη της web εφαρμογής μας βασιστήκαμε στον εξυπηρετητή (server) Apache Tomcat 5.0. Ο **Tomcat** είναι μια open-source εφαρμογή των προδιαγραφών των τεχνολογιών Java Servlets και JavaServer Pages, που αναπτύσσεται κάτω από το Jakarta project στο Apache Software Foundation. Ο Tomcat εφαρμόζει ένα νέο Servlet framework (ονομαζόμενο Catalina) που στηρίζεται επάνω σε μια εντελώς νέα αρχιτεκτονική με βάση τις Servlet 2.3 and JSP 1.2 προδιαγραφές. Περιλαμβάνει πολλές νέες δυνατότητες που τον ανάγουν σε μια χρήσιμη πλατφόρμα για την δημιουργία αλλά και ανάπτυξη εφαρμογών αλλά και υπηρεσιών web. Με λίγα λόγια, ο Tomcat είναι ένας διακομιστής εφαρμογών γραμμένος σε 100% καθαρή Java.

Ο Tomcat χρησιμοποιείται για πολλούς σκοπούς και δεν περιορίζεται μόνο στη χρήση του ως διακομιστή εφαρμογών. Παρέχει μια ανοιχτή πλατφόρμα για την δημιουργία επεκτάσιμων υπηρεσιών ιστού αλλά και υπηρεσιών content management. Η χρήση του Tomcat μπορεί να παρέχει πολύ συνεπείς αλλά και υψηλών ρυθμών εξυπηρέτησης υπηρεσίες.

Έτσι, υπάρχει λογισμικό που «τρέχει» στον τοπικό εξυπηρετητή και έχει ως κύρια ευθύνη την συλλογή των απαραίτητων, σχετικών με την ηλεκτρονική ψηφοφορία, πληροφοριών μέσω ενός interface επικοινωνίας με τον ψηφοφόρο. Το ενδιάμεσο αυτό λογισμικό είναι υλοποιημένο σε γλώσσα Java, επιτυγχάνοντας έτσι την μέγιστη δυνατή ομοιογένεια της

εφαρμογής. Γενικότερα, ο εξυπηρετητής είναι το σημείο που λαμβάνουν χώρα οι περισσότερες διεργασίες και αποτελεί το απαραίτητο περιβάλλον για την χρήση των JSP που διαχειρίζονται τις αιτήσεις του χρήστη αλλά και την ανάκτηση δεδομένων από τη βάση.

Η χρήση των JSP, στην παρούσα εργασία, αποσκοπεί στο να δώσει στον εξυπηρετητή την πληροφορία για το πώς θα κατευθύνει τις αιτήσεις του ψηφοφόρου στα κατάλληλα αρχεία ή διαδικασίες που είναι εγκατεστημένες μέσα του. Ο Tomcat είναι, μεταξύ των άλλων, ένας servlet container, δηλαδή ένα περιβάλλον μέσα από το οποίο εκτίθενται στο διαδίκτυο όσα εκτελούν τα διάφορα από αυτόν φιλοξενούμενα servlets. Ο Tomcat τρέχει σε ένα περιβάλλον ενός κοινού web ή HTTP server, στην συγκεκριμένη περίπτωση του Apache. Αυτός συνυπάρχει αλλά δεν «φαίνεται», καθόσον ασχολείται στο να δέχεται και να αποστέλλει HTTP μηνύματα [11]. Η χρήση της συγκεκριμένης τεχνολογίας γίνεται και για λόγους ολοκλήρωσης με την υποκείμενη πλατφόρμα (βλέπε 5.7).

5.3 Java Server Pages (JSP)

Μια πολύ σημαντική και αναπτυσσόμενη τεχνολογία που αναδύθηκε μέσα από την Java είναι αυτή των **JSP (Java Server Pages)**.

Οι JSP (JavaServer Pages) είναι μια server-side τεχνολογία, που παρουσιάστηκε από την Sun Microsystems Corp., και παρέχει ένα γρήγορο και απλό τρόπο για την παραγωγή δυναμικού περιεχομένου μέσα από HTML (HyperText Markup Language) σελίδες. Χρησιμοποιεί XML(Extensible Markup Language) tags μαζί με Java scriptlets για να ενσωματώσει και να διαχωρίσει έτσι τη λογική από τον αισθητικό σχεδιασμό και εμφάνιση. Όταν καλείται μια JSP σελίδα, μετατρέπεται δυναμικά σε ένα Servlet το οποίο υπόκειται σε επεξεργασία από τον server για να παραχθεί η τελική HTML/XML σελίδα που θα δει ο client. Όταν η JSP τεχνολογία χρησιμοποιείται σε συνδυασμό με την τεχνολογία JavaBeans, είναι δυνατό να παραχθούν ποικιλόμορφες και επιδεχόμενες μεγάλης κλιμάκωσης εφαρμογές.

Πλεονεκτήματα από τη χρήση JSP: Από τη χρήση της τεχνολογίας JSP απορρέουν κάποια αρκετά σημαντικά πλεονεκτήματα συγκριτικά με άλλες τεχνολογίες.

Σε σύγκριση με τεχνολογία Active Server Pages (ASP)

Η ASP είναι μια ανταγωνιστική τεχνολογία της Microsoft. Τα πλεονεκτήματα της χρήσης JSP εντοπίζονται σε 2 σημεία. Καταρχάς, το δυναμικό μέρος είναι γραμμένο σε Java και όχι σε VBScript ή σε κάποια άλλη ASP-specific γλώσσα., κι έτσι αποτελεί πιο σίγουρη επιλογή για υλοποίηση σύνθετων εφαρμογών, οι οποίες απαιτούν επαναχρησιμοποιούμενες συνιστώσες. Επιπρόσθετα, οι JSP μπορούν να μεταφερθούν και να ενσωματωθούν σε άλλα διαχειριστικά συστήματα και εξυπηρετητές Web, οπότε αποφεύγεται κάθε ενδεχόμενη αναγκαστική προσκόλληση στα Windows NT/2000 και IIS (Internet Information Services). Τα ίδια επιχειρήματα θα μπορούσε να φέρει κανείς συγκρίνοντας την τεχνολογία JSP με την ColdFusion: στις JSP μπορεί να χρησιμοποιηθεί γλώσσα Java και δεν απαιτούν την ύπαρξη κάποιου συγκεκριμένου εξυπηρετητή.

Σε σύγκριση με τεχνολογία PHP

Η PHP (Hypertext Preprocessor) είναι μια ελεύθερη, open-source, scripting γλώσσα, ενσωματωμένη στον HTML κώδικα, η οποία κατά κάποιον τρόπο είναι παρεμφερής τόσο με την ASP όσο και με την JSP. Το πλεονέκτημα των JSP είναι και πάλι το γεγονός ότι το δυναμικό μέρος είναι γραμμένο σε Java. Υπάρχει εκτενές API για δικτύωση (networking), δυνατότητα πρόσβασης σε βάση δεδομένων, κατανεμημένα αντικείμενα, τη στιγμή που η PHP απαιτεί την εκμάθηση μίας εξολοκλήρου νέας γλώσσας.

Σε σύγκριση με απλά Servlets

Στη γενική περίπτωση οι JSP δεν παρέχουν δυνατότητες, που δεν θα μπορούσαν να υλοποιηθούν με ένα servlet. Στην πραγματικότητα, τα JSP αρχεία μεταγλωττίζονται αυτόματα σε servlets. Όμως είναι πιο εύκολα υλοποιήσιμο να γράψει και να τροποποιήσει κάποιος ένα κανονικό HTML αρχείο, αντί για αμέτρητα println statements που παράγουν μια HTML φόρμα. Επίσης, διαχωρίζοντας την παρουσίαση από το περιεχόμενο, είναι δυνατόν να τοποθετηθούν διαφορετικοί άνθρωποι σε διαφορετικές εργασίες: οι ειδικοί σχεδιαστές της Web σελίδας είναι σε θέση να «χτίσουν» την HTML φόρμα χρησιμοποιώντας τα σχετικά

εργαλεία, αφήνοντας σημεία για τους προγραμματιστές των servlets να εισάγουν το δυναμικό περιεχόμενο.

Σε σύγκριση με τεχνολογία Server-Side Includes (SSI)

Η SSI είναι μια ευρέως υποστηριζόμενη τεχνολογία για την εισαγωγή εξωτερικά ορισμένων «κομματιών» σε μια στατική Web σελίδα. Η τεχνολογία JSP είναι καλύτερη σε αυτήν την περίπτωση, για το λόγο ότι παρέχει ένα πλουσιότερο σύνολο εργαλείων για κατασκευή τέτοιων εξωτερικών κομματιών και επίσης επειδή προσφέρει περισσότερες επιλογές σε ό,τι αφορά το στάδιο του HTTP response, κατά το οποίο εισάγεται ουσιαστικά το κομμάτι. Εξάλλου, η SSI αποσκοπεί αποκλειστικά στο σχεδιασμό απλών εφαρμογών, και όχι στο σχεδιασμό «αληθινών» προγραμμάτων, τα οποία χρησιμοποιούν form data, συνδέονται σε βάσεις δεδομένων, κλπ.

Σε σύγκριση με JavaScript

Η JavaScript, η οποία διαφέρει πλήρως από την γλώσσα προγραμματισμού Java, συνήθως χρησιμοποιείται για να δημιουργήσει δυναμικά HTML στην πλευρά του πελάτη (*client)*, χτίζοντας κομμάτια της Web σελίδας κατά το διάστημα που ο browser φορτώνει το αρχείο. Αυτή είναι μια ιδιαιτέρως χρήσιμη δυνατότητα, όμως χειρίζεται μόνο περιπτώσεις, στις οποίες η δυναμική πληροφορία στηρίζεται στο περιβάλλον του πελάτη. Με εξαίρεση τα cookies, το HTTP request data δεν είναι διαθέσιμο σε client-side JavaScript ρουτίνες. Και, καθώς η JavaScript δε διαθέτει ρουτίνες διαδικτυακού προγραμματισμού, ο κώδικας JavaScript στον πελάτη δεν έχει πρόσβαση σε server-side στοιχεία, όπως βάσεις δεδομένων, ευρετήρια, κλπ. Η γλώσσα JavaScript μπορεί επίσης να χρησιμοποιηθεί στον εξυπηρετητή (server), με πιο αξιοσημείωτη περίπτωση τους εξυπηρετητές Netscape, και σαν scripting γλώσσα για IIS. Η Java είναι σαφώς πιο ισχυρή, ευέλικτη, έμπιστη και μεταφέρσιμη.

Σε σύγκριση με Static HTML

Τα συνηθισμένα HTML, φυσικά , δεν μπορούν να περιέχουν δυναμική πληροφορία κι έτσι οι στατικές HTML σελίδες δεν είναι δυνατόν να βασιστούν σε user input ή server-side πηγές δεδομένων. Οι JSP είναι τόσο απλές και λειτουργικές ώστε είναι αρκετά λογικό να προτιμώνται από τις HTML σελίδες [10].

Στην παρούσα εφαρμογή χρησιμοποιήθηκε η προδιαγραφή 1.2 για JSP.

5.4 Διασύνδεση Java με Βάσεις Δεδομένων

Η τυποποίηση Java Database Connectivity (JDBC) ορίζει ένα API που μπορούν να χρησιμοποιήσουν τα προγράμματα Java για να συνδεθούν σε διακομιστές βάσεων δεδομένων. Το εκάστοτε πρόγραμμα πρέπει να «ανοίγει» πρώτα μια σύνδεση με μια βάση δεδομένων και μετά να εκτελεί SQL εντολές, αφού πρώτα φορτώσει τα κατάλληλα προγράμματα οδήγησης (JDBC technology-based driver) για την βάση.

Συγκεκριμένα, τα βήματα χρήσης της τυποποίησης JDBC είναι τα εξής:

1. «Φορτώνουμε» τον Driver.
2. Συνδεόμαστε με τη Βάση Δεδομένων. Φτιάχνουμε ένα αντικείμενο Connection
3. Φτιάχνουμε μια δήλωση SQL (Statement) προς τη σύνδεση αυτή.
4. Εκτελούμε διαδοχικά (μια ή περισσότερες) ερωτήσεις SQL στο αντικείμενο Statement με χρήση της executeQuery ή executeUpdate
5. Επεξεργαζόμαστε το αντικείμενο ResultSet που δημιουργείται όταν κάνουμε ερωτήσεις επιλογής στο βήμα 4.
6. Κλείνουμε τα Statement και Connection αντικείμενα.

Ο driver μπορεί να είναι υλοποιημένος καθαρά με χρήση γλώσσας Java ή σε συνδυασμό με Java Native Interface (JNI) κώδικα.

Έτσι, είναι δυνατή η εικονική πρόσβαση σε κάθε πηγή δεδομένων, από σχεσιακές βάσεις δεδομένων μέχρι και απλά αρχεία.

Το JDBC API είναι χωρισμένο σε 2 πακέτα:

1.το πακέτο `java.sql` και

2.`το πακέτο javax.sql`, το οποίο ενσωματώνει και server-side δυνατότητες [7].

5.5 Σύστημα διαχείρισης Βάσεων Δεδομένων MySQL

Η MySQL είναι ένα πολυνηματικό σύστημα διαχείρισης βάσεων δεδομένων (DBMS, Database Management System) που βασίζεται στη γλώσσα SQL. Το βασικό πρόγραμμα «τρέχει» σαν ένας εξυπηρετητής (server) που παρέχει πολλαπλή πρόσβαση σε ένα πλήθος βάσεων δεδομένων. Είναι ιδιαίτερα δημοφιλής για εφαρμογές ιστού και χρησιμοποιείται ιδιαίτερα από τις λεγόμενες LAMP, MAMP και WAMP πλατφόρμες (Linux/Mac/Windows-Apache-MySQL-PHP/Perl/Python).

Η MySQL θεωρείται ιδανική για μεσαίας κλίμακας εφαρμογές. Είναι γρήγορη, η διαχείριση και η ασφάλειά της αποτελεσματικές, όχι ιδιαίτερα πολύπλοκη στη χρήση, ενώ ενδείκνυται και για μεταφέρσιμο κώδικα. Από την άλλη στερείται πιο εξειδικευμένων χαρακτηριστικών, που διαθέτουν, για παράδειγμα, συστήματα όπως η Oracle.

Κυριότερα χαρακτηριστικά:

- Υλοποίηση και μεταφερσιμότητα

Είναι γραμμένη σε C και C++. Δουλεύει πάνω σε πολλές διαφορετικές πλατφόρμες: APIs για C, C++, Eiffel, Java, Perl, PHP, Python, Ruby και Tcl. Πραγματοποιεί πολύ γρήγορους συνδέσμους και ενώνει πίνακες από διαφορετικές βάσεις στο ίδιο query. Χρησιμοποιεί πίνακες κατακερματισμού in-memory σαν προσωρινούς πίνακες. Συνήθως δεν υπάρχει καθόλου ανάθεση μνήμης μετά την αρχικοποίηση κάποιου ερωτήματος.

- Ασφάλεια

Όλοι οι κωδικοί κατά τη σύνδεση με τον εξυπηρετητή κρυπτογραφούνται. Πολύ ευέλικτο σύστημα κωδικών και προνομίων που επιτρέπει την εξακρίβωση.

- Δυνατότητες

Χειρίζεται μεγάλες βάσεις δεδομένων. Μέγιστο μέγεθος πίνακα 8 TB (default 4 TB).

- Συνδεσιμότητα

Ο πελάτης συνδέεται στον εξυπηρετητή MySQL χρησιμοποιώντας TCP/IP Sockets, Unix Sockets ή Named Pipes (NT).

- Μηχανές αποθήκευσης (οι ακόλουθες δε συναντώνται σε άλλα συστήματα):
    - Πολλαπλές μηχανές αποθήκευσης, που επιτρέπουν επιλογή της πιο κατάλληλης για κάθε πίνακα της εφαρμογής.
    - Εγγενείς (native) μηχανές αποθήκευσης (MyISAM, Falcon, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, Cluster, BDB, EXAMPLE).
    - Partner-developed (InnoDB, solidDB, NitroEDB, BrightHouse).
    - Community-developed (memcached, httpd, PBXT)
    - Custom
    - Ομαδοποίηση, συγκεντρώνοντας πολλαπλές συναλλαγές από πολλαπλές συνδέσεις μαζί, ώστε να αυξάνεται ο αριθμός των ενεργειών ανά δευτερόλεπτο[3][4].

5.6 Κρυπτογραφία

5.6.1 Κρυπτογράφηση

Για την κρυπτογράφηση των μηνυμάτων πριν τη μεταφορά τους και την αποκρυπτογράφηση στον προορισμό χρησιμοποιήσαμε τα πακέτα javax.crypto και java.security της Java. Η κλάση Cipher του javax.crypto παρέχει τη λειτουργικότητα για το αντικείμενο εκείνο που χρησιμοποιείται για την κρυπτογράφηση και αποκρυπτογράφηση. Κρυπτογράφηση είναι η διαδικασία κατά την οποία από τα αρχικά δεδομένα (cleartext) και ένα κλειδί παράγονται νέα δεδομένα (ciphertext), ακατανόητα σε μια τρίτη οντότητα που δε γνωρίζει το κλειδί. Η αποκρυπτογράφηση είναι η αντίστροφη διαδικασία: από τα κρυπτογραφημένα δεδομένα και ένα κλειδί προκύπτουν τα αρχικά δεδομένα.

Υπάρχουν δύο βασικοί τύποι κρυπτογραφίας: η *συμμετρική* (ή *μυστικού κλειδιού*) και η *ασύμμετρη* (ή *δημόσιου κλειδιού*). Στη συμμετρική κρυπτογράφηση χρησιμοποιείται το ίδιο

μυστικό κλειδί τόσο για την κρυπτογράφηση όσο και για την αποκρυπτογράφηση. Η μυστικότητα του κλειδιού είναι κρίσιμος παράγοντας για το απόρρητο των δεδομένων. Από την άλλη, η ασύμμετρη κρυπτογραφία χρησιμοποιεί ζεύγος ιδιωτικού/ δημόσιου κλειδιού. Τα δεδομένα που κρυπτογραφούνται με το ένα κλειδί αποκρυπτογραφούνται με το άλλο. Ο χρήστης αρχικά παράγει ένα ζεύγος κλειδιών και στη συνέχεια δημοσιεύει το δημόσιο κλειδί σε μια έμπιστη βάση δεδομένων προσβάσιμη από όλους. Η οντότητα που επιθυμεί να επικοινωνήσει με ασφάλεια με το συγκεκριμένο χρήστη κρυπτογραφεί τα δεδομένα χρησιμοποιώντας το ανακτηθέν δημόσιο κλειδί. Μόνο ο κάτοχος του ιδιωτικού κλειδιού θα μπορέσει να τα αποκρυπτογραφήσει. Ο κρίσιμος παράγοντας σε αυτό το σχήμα είναι η μυστικότητα του ιδιωτικού κλειδιού.

Οι ασύμμετροι αλγόριθμοι (π.χ. RSA) είναι γενικά πολύ πιο αργοί από τους συμμετρικούς και δεν είναι σχεδιασμένοι για αποδοτική προστασία μεγάλου όγκου δεδομένων. Στην πράξη χρησιμοποιούνται για ανταλλαγή μικρότερων μυστικών κλειδιών προς αρχικοποίηση συμμετρικών αλγορίθμων. Εμείς ωστόσο χρησιμοποιήσαμε RSA για την κρυπτογράφηση της ψήφου, εφόσον δεν πρόκειται κάθε φορά για μεταφορά μεγάλου μηνύματος και λόγω των πλεονεκτημάτων που έχουμε δει σε προηγούμενο κεφάλαιο, αλλά και επειδή σε αυτή τη χρήση οδηγεί και η τοπολογία του συστήματος (αστέρας).

Stream vs. Block Ciphers

Άλλη μια διάκριση είναι η τμηματική (block) κρυπτογράφηση και η (stream) κρυπτογράφηση ροής. Η πρώτη επεξεργάζεται μεγάλα κομμάτια δεδομένων (blocks) κάθε φορά, με μήκος πολλών bytes. Αν τα δεδομένα δεν επαρκούν για να σχηματίσουν ένα τέτοιο κομμάτι, «γεμίζονται» (padded) πριν την κρυπτογράφηση, δηλαδή προσθέτονται dummy bytes, προκειμένου να έχουμε πολλαπλάσιο του μεγέθους του block, τα οποία αφαιρούνται κατά την αποκρυπτογράφηση. Ο τρόπος γεμίσματος καθορίζεται κατά την αρχικοποίηση του αντικειμένου κρυπτογράφησης με τον αντίστοιχο τύπο (π.χ. "PKCS5PADDING"). Αντίθετα, η κρυπτογράφηση ροής (stream) επεξεργάζεται στοιχειώδεις μονάδες δεδομένων (byte ή bit) κάθε φορά, ώστε να είναι δυνατή η επεξεργασία οποιουδήποτε μεγέθους μηνύματος χωρίς «γέμισμα».

Τύποι λειτουργίας (modes)

Από τα παραπάνω προκύπτει ότι πανομοιότυπα κομμάτια (blocks) δεδομένων θα παράγουν πανομοιότυπα κρυπτογραφήματα (ciphers), κάτι που θα διευκόλυνε την αποκρυπτογράφηση από τρίτους, εφόσον εντοπιστεί επαναλαμβανόμενο κείμενο. Η λύση είναι η ανατροφοδότηση κάθε block, χρησιμοποιώντας το προηγούμενό του, προτού εφαρμοστεί σε αυτό ο εκάστοτε αλγόριθμος. Το πρώτο block θα χρειαστεί μια αρχική τιμή, το διάνυσμα αρχικοποίησης *(initialization vector, IV)*, το οποίο πρέπει να είναι τυχαίο αλλά όχι απαραίτητα μυστικό. Υπάρχουν διάφοροι τύποι ανατροφοδότησης: CBC (Cipher Block Chaining), CFB (Cipher Feedback Mode), και OFB (Output Feedback Mode). Ο ECB (Electronic Codebook Mode) δηλώνει λειτουργία χωρίς ανατροφοδότηση.

Σε ό,τι αφορά το μέγεθος κλειδιού τέλος, κάποιοι αλγόριθμοι (AES, RSA) δίνουν τη δυνατότητα ρύθμισής του, ενώ κάποιοι άλλοι δίνουν σταθερό μέγεθος (DES, 3DES). Τα μεγαλύτερα κλειδιά παρέχουν γενικά ισχυρότερη αντίσταση στην ανάκτηση μηνύματος. Η επιλογή, ως συνήθως, γίνεται με βάση την αντιστάθμιση ανάμεσα στον επιθυμητό βαθμό ασφάλειας και το χρόνο επεξεργασίας. Εδώ χρησιμοποιήσαμε μέγεθος κλειδιού 1024, που είναι και η default τιμή για αλγόριθμο RSA.

Τα βήματα που ακολουθήσαμε για την κρυπτογράφηση κάθε μηνύματος είναι τα ακόλουθα:

Δημιουργία αντικειμένου Cipher

Κατά τη δημιουργία, προσδιορίζεται ο συγκεκριμένος «μετασχηματισμός» που χρησιμοποιείται, το είδος, δηλαδή, της λειτουργίας που θα εφαρμοστεί στη είσοδο για να παράγει την έξοδο. Περιλαμβάνει πάντα το όνομα μιας κρυπτογραφικής μεθόδου (π.χ. DES) και μπορεί να συνοδεύεται και από άλλα προσδιοριστικά (mode, padding scheme). Είναι της μορφής:

- "*algorithm/mode/padding*" ή
- "*algorithm*"

Στην περίπτωση που προσδιορίζεται μόνο ο αλγόριθμος, χρησιμοποιούνται οι default ρυθμίσεις του παρόχου, σε σχέση με mode, padding scheme. Εμείς δώσαμε μόνο τον αλγόριθμο με την εντολή:

```
Cipher cipher=Cipher.getInstance("RSA");          (κρυπτογράφηση)
Cipher decipher=Cipher.getInstance("RSA");        (αποκρυπτογράφηση)
```

Ο πάροχος SunJCE χρησιμοποιεί ECB σαν default mode και PKCS1Padding σαν default padding scheme για τον αλγόριθμο RSA.

Αρχικοποίηση του αντικειμένου

Το αντικείμενο Cipher που δημιουργήθηκε πρέπει να αρχικοποιηθεί με μια από τις ακόλουθες τέσσερις λειτουργίες (opmodes):

ENCRYPT_MODE
Κρυπτογράφηση δεδομένων.
DECRYPT_MODE
Αποκρυπτογράφηση δεδομένων.
WRAP_MODE
Εσωκλείει ένα αντικείμενο `java.security.Key` σε bytes ώστε να μπορεί να μεταφερθεί με ασφάλεια.
UNWRAP_MODE
Αποδεσμεύει ένα εσώκλειστο κλειδί σε αντικείμενο `java.security.Key`.

Άλλες παράμετροι αρχικοποίησης περιλαμβάνουν το κλειδί (key) ή το πιστοποιητικό (certificate) που το περιέχει, αλγοριθμικές παραμέτρους (params), και μια γεννήτρια τυχαιότητας (random). Αν κάποιες παράμετροι δεν αρχικοποιηθούν, χρησιμοποιούνται τυχαία παραγόμενες ή default τιμές από την πλατφόρμα. Με κάθε αρχικοποίηση, τέλος, ένα ήδη υπάρχον αντικείμενο χάνει την προηγούμενη κατάστασή του.

Κατά την κρυπογράφηση, αρχικοποιήσαμε ως εξής:

```
cipher.init(Cipher.ENCRYPT_MODE, pbk_cipher,random);
```

και κατά την αποκρυπτογράφηση:

```
decipher.init(Cipher.DECRYPT_MODE, kp_cipher1.getPrivate());
```

όπου pbk_cipher το δημόσιο κλειδί που έχει διανεμηθεί και kp_cipher1 το αρχείο του ζεύγους κλειδιών του χρήστη (κεντρικό σύστημα).

***Κρυπτογράφηση και αποκρυπτογράφηση*.**

Εφόσον έχουμε μικρά και δεδομένου μήκους μηνύματα, επιλέξαμε κρυπτογράφηση και αποκρυπτογράφηση σε ένα βήμα (*single-part operation)* με τις ακόλουθες εντολές:

```
encryptedVoteBytes=cipher.doFinal(voteBytes);
voteBytes1=decipher.doFinal(encryptedVoteBytes1);
```

[1]

5.6.2 Ψηφιακές υπογραφές

Η χρησιμότητα των ψηφιακών υπογραφών (digital signatures) έγκειται στο γεγονός, ότι μια ψηφιακή υπογραφή δημιουργείται με τη χρήση ενός «μυστικού», το οποίο είναι γνωστό μόνο στον δημιουργό της υπογραφής, και ότι η ψηφιακή αυτή υπογραφή μπορεί να επαληθευτεί κάνοντας χρήση αυτή τη φορά δημόσιας πληροφορίας, την οποία γνωστοποιεί ο δημιουργός της υπογραφής. Από τη στιγμή που μια υπογραφή προωθείται από τον δημιουργό της, αν ο δημιουργός αρνηθεί ότι είναι δική του η υπογραφή, υπάρχουν δύο τινά: είτε διακυβεύεται το «μυστικό» που αναφέραμε παραπάνω, είτε ο δημιουργός ψεύδεται.

Οι ψηφιακές υπογραφές βασίζονται στη χρήση κρυπτογραφικών μηχανισμών κατακερματισμού (cryptographic hashes), ή συνόψεις μηνύματος (message digests). Αυτό είναι απαραίτητο εξαιτίας των περιορισμών που θέτουν οι ασύμμετροι κρυπτογραφικοί αλγόριθμοι στο μέγεθος του μηνύματος. Έτσι εξάγεται το εξής συμπέρασμα: τελικά είναι ο περιορισμός μεγέθους εισόδου στις συνόψεις μηνύματος που χρησιμοποιούνται στην υπογραφή, που καθορίζει ποια ποσότητα δεδομένων είναι ασφαλές να υπογράφεται με έναν συγκεκριμένο αλγόριθμο σύνοψης. Το μέγεθος του κλειδιού εξυπηρετεί μόνο την προστασία της σύνοψης. Αν η προς υπογραφή πληροφορία είναι μεγαλύτερη από ένα επιτρεπτό για τον συγκεκριμένο αλγόριθμο σύνοψης maximum, η αύξηση του μεγέθους του κλειδιού δεν θα προσφέρει κάποια βοήθεια.

Υπάρχουν δύο διαδικασίες που σχετίζονται με τις ψηφιακές υπογραφές: η κατασκευή της υπογραφής και η επαλήθευσή της. Οι διαδικασίες αυτές περιλαμβάνουν κρυπτογραφικούς μηχανισμούς, κρυπτογράφηση και αποκρυπτογράφηση. Ένας χρήστης μπορεί μόνο να επαληθεύσει μια υπογραφή και όχι να διαβάσει τα περιεχόμενα της. Οι δύο παραπάνω διαδικασίες υλοποιούνται στη Java μέσω της κλάσης `Signature`.

Η κλάση `Signature`

Αντικείμενα τα οποία παρέχουν τη βασική λειτουργικότητα που προσφέρεται από το `java.security.Signature` δημιουργούνται χρησιμοποιώντας το εργαλείο `getInstance()`, όπως και άλλες παρεμφερείς κλάσεις στο JCA. Ανάλογα με το αν καλείται η μέθοδος `Signature.initSign()` ή η μέθοδος `Signature.initVerify()`, το αντικείμενο `Signature` προχωρά στην δημιουργία μιας υπογραφής ή στην επαλήθευση μιας υπογραφής αντίστοιχα.

Χρησιμοποιώντας την κλάση `Signature` για τη δημιουργία μίας Υπογραφής

Υπάρχουν δύο διαφοροποιήσεις της μεθόδου `Signature.initSign()`, των οποίων μπορεί να γίνει κλήση όταν τίθεται ένα αντικείμενο `Signature` στη διαδικασία δημιουργίας μιας υπογραφής. Και οι δύο παίρνουν ένα ιδιωτικό κλειδί, το οποίο είναι και το πιο σημαντικό bit, ενώ μία μέθοδος `initSign()` δίνει επίσης δυνατότητα τυχαιότητας.

Αφού το αντικείμενο `Signature` έχει αρχικοποιηθεί για υπογραφή, υπάρχει ένα σετ `Signature.update` μεθόδων, που χρησιμεύουν στον εφοδιασμό του αντικειμένου `Signature` με δεδομένα, κι όταν έχουν πια φορτωθεί όλα τα δεδομένα, καλείται η μέθοδος `Signature.sign()`. Ανάλογα με την παραλλαγή που καλείται, η μέθοδος είτε επιστρέφει την υπογραφή σαν ένα πίνακα byte είτε φορτώνει την υπογραφή σ' έναν πίνακα byte.

Χρησιμοποιώντας την κλάση `Signature` για την επαλήθευση μίας Υπογραφής

Υπάρχουν επίσης δύο διαφοροποιήσεις της μεθόδου `Signature.initVerify()`. Σε αντίθεση με τη μέθοδο `initSign()`, η δεύτερη παραλλαγή, η οποία λαμβάνει ένα αντικείμενο `Certificate` σαν όρισμα, είναι μια άλλη, για λόγους ευκολίας στην χρήση, παραλλαγή της πρώτης `initVerify()` μεθόδου, η οποία λαμβάνει σαν όρισμα ένα δημόσιο κλειδί.

Η εισαγωγή δεδομένων στο αντικείμενο `Signature`, σαν στάδιο της διαδικασίας επιβεβαίωσης, είναι ταυτόσημη με το αντίστοιχο στάδιο της διαδικασίας δημιουργίας της υπογραφής. Απλά χρησιμοποιούνται οι μέθοδοι `update()`. Το σημείο, όμως, όπου γίνεται ουσιαστικά η επιβεβαίωση της υπογραφής είναι ελαφρώς διαφορετικό. Αφού όλα τα απαραίτητα δεδομένα έχουν εισαχθεί στο αντικείμενο `Signature` κατά την δημιουργία της υπογραφής με χρήση των μεθόδων `update()`, καλείται η boolean μέθοδος `Signature.verify()` και ο byte πίνακας που αναπαριστά την υπογραφή η οποία περνά από έλεγχο εισάγεται σαν όρισμα. Αν η μέθοδος `verify()` επιστρέψει τιμή `true`, η υπογραφή είναι έγκυρη, ενώ σε περίπτωση που επιστρέψει `false`, δεν υπάρχει «ταίριασμα» μεταξύ δεδομένων και υπογραφής.

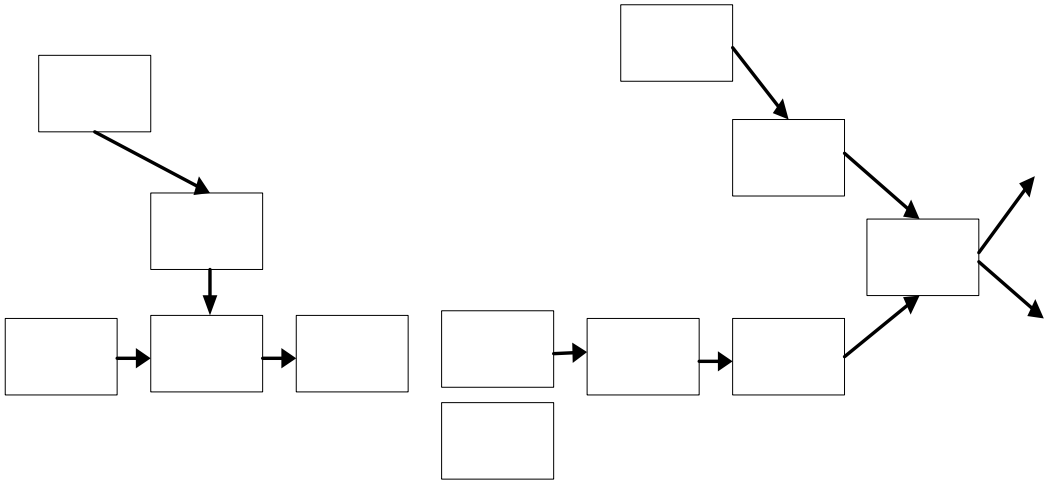
RSA-based Αλγόριθμοι Υπογραφής

Στην περίπτωση αυτή, οι υπογραφές δημιουργούνται με εφαρμογή του RSA αλγόριθμου χρησιμοποιώντας το ιδιωτικό κλειδί και στη συνέχεια διανέμεται σαν αποτέλεσμα η υπογραφή. Η υπογραφή είναι δυνατόν να αποκρυπτογραφηθεί με χρήση του δημοσίου κλειδιού. Οι RSA υπογραφές είναι ιδιαιτέρως αποτελεσματικές, γιατί αν μια υπογραφή αποκρυπτογραφείται επιτυχώς με το δοσμένο δημόσιο κλειδί, τότε θα πρέπει να έχει δημιουργηθεί με το αντίστοιχο ιδιωτικό κλειδί [8].

Χρησιμοποιήσαμε αλγόριθμο RSA με SHA1. Ο RSA θεωρείται εξαιρετικά ασφαλής, ενώ ο αλγόριθμος κατακερματισμού SHA1 έχει τιμή κατακερματισμού μικρότερη από άλλους της ίδιας κατηγορίας, οπότε η επεξεργασία και επαλήθευση των υπογραφών γίνονται ταχύτερα, όπως είδαμε σε προηγούμενο κεφάλαιο.

Το στιγμιότυπο της κλάσης Signature δημιουργήθηκε ως εξής:

```
Signature signature=Signature.getInstance("SHA1withRSA");
```



Σχήμα 5.2 Δημιουργία και επαλήθευση ψηφιακής υπογραφής

[8]

5.6.3 Πιστοποιητικά δημόσιου κλειδιού

Τα πιστοποιητικά, και συγκεκριμένα τα πιστοποιητικά δημοσίου κλειδιού, παρέχουν ένα μηχανισμό, ο οποίος επιτρέπει σε μια τρίτη πλευρά, ή εκδότη, να εγγυηθεί για το γεγονός, ότι ένα συγκεκριμένο δημόσιο κλειδί συνδέεται με έναν συγκεκριμένο ιδιοκτήτη. Κάθε πιστοποιητικό έχει ένα αντίστοιχο ιδιωτικό κλειδί. Μια αλυσίδα πιστοποιητικών είναι μια λίστα πιστοποιητικών, όπου κάθε πιστοποιητικό, εκτός του πρώτου και του τελευταίου, έχει ένα ιδιωτικό κλειδί με το οποίο υπογράφει το επόμενο στην αλυσίδα πιστοποιητικό. Το πρώτο πιστοποιητικό στην αλυσίδα (root certificate) είναι self-signed. Πρέπει να γίνει η παραδοχή ότι το πρώτο πιστοποιητικό είναι έγκυρο, ώστε να είναι και η αλυσίδα έγκυρη. Το τελευταίο πιστοποιητικό στην αλυσίδα απλά παρέχει ένα δημόσιο κλειδί, το οποίο ο ενδιαφερόμενος θεωρεί αυθεντικό, αν θεωρεί την αλυσίδα έγκυρη. Η οντότητα που είναι υπεύθυνη για την έκδοση των πιστοποιητικών ονομάζεται *CA* (Certification Authority, CA).

Data Digest Function

5.6.3.1 Η κλάση Certificate

Η κλάση `java.security.cert.Certificate` παρέχει τη βασική υποστήριξη για πιστοποιητικά δημοσίου κλειδιού. Παρέχει τις απαραίτητες μεθόδους για την απόσπαση του δημοσίου κλειδιού που φυλάσσεται μέσα στο πιστοποιητικό, την αναγνώριση του τύπου του πιστοποιητικού, την επαλήθευση της υπογραφής που περιέχεται στο πιστοποιητικό και την επιστροφή του πιστοποιητικού σε κωδικοποιημένο μορφή.

Private Key RSA Engine Signature Signature

Public Key

Certificate.getType()

Η `getType()` μέθοδος επιστρέφει ένα String που δηλώνει τον τύπο του πιστοποιητικού. Στη μεγάλη πλειοψηφία των περιπτώσεων αναμένεται η μέθοδος αυτή να επιστρέψει "X.509", που σημαίνει ότι το πιστοποιητικό είναι τύπου "X.509" και υλοποιείται με την κλάση `java.security.cert.X509Certificate.`

Certificate.getPublicKey()

Η μέθοδος `getPublickey` επιστρέφει το δημόσιο κλειδί, το οποίο εμπεριέχεται στο πιστοποιητικό.

Το κλειδί είναι κατάλληλο γα χρήση, οποιουδήποτε παρόχου και αν γίνεται χρήση για την δημιουργία του πιστοποιητικού, αλλά το αποτέλεσμα θα διαφέρει αν γίνει προσπάθεια να χρησιμοποιηθούν σε πιστοποιητικά ενός παρόχου κρυπτογράφηση ή αλγόριθμοι υπογραφής που υλοποιούνται από άλλον πάροχο. Σ' αυτήν την περίπτωση, είναι αναγκαία η ύπαρξη ενός `KeyFactory`, που δημιουργείται για τον πάροχο του αλγορίθμου, ώστε να μετατραπεί το κλειδί που αποσπάται από το πιστοποιητικό σε κάτι λειτουργικό για τον ίδιο τον πάροχο.

Certificate.verify()

Η μέθοδος `verify()` ελέγχει αν η υπογραφή που περιέχεται στο πιστοποιητικό μπορεί να επαληθευτεί χρησιμοποιώντας το δημόσιο κλειδί της οντότητας, η οποία παρουσιάζεται να έχει υπογράψει το πιστοποιητικό. Υπάρχουν δύο εκδοχές της μεθόδου `verify()`. Η πρώτη παίρνει σαν όρισμα το δημόσιο κλειδί για να το χρησιμοποιήσει για την επαλήθευση, ενώ η δεύτερη παίρνει επιπλέον σαν όρισμα και το όνομα του παρόχου.

Σε αντίθεση με την `Signature.verify()`, η μέθοδος αυτή είναι τύπου void και μπορεί να «πετάξει» πέντε εξαιρέσεις, από τις οποίες η πλέον συνηθισμένη είναι η `SignatureException`, η οποία αποτελεί ένδειξη ότι το δημόσιο κλειδί είναι λάθος και ότι η υπογραφή δεν ήταν η αναμενόμενη. Οι υπόλοιπες 4 είναι οι παρακάτω:

- `NoSuchProviderExcepion`, σε περίπτωση που δεν μπορεί να βρεθεί πάροχος.
- `NoSuchAlgorithmException`, αν ο αλγόριθμος που χρησιμοποιείται στο πιστοποιητικό δεν μπορεί να αναγνωριστεί.
- `InvalidKeyException`, εάν το δημόσιο κλειδί που προωθείται είναι για λάθος αλγόριθμο.
- `CertificateException`, σε περίπτωση που υπάρχει πρόβλημα στην κωδικοποίηση του πιστοποιητικού, με σκοπό να υπολογιστεί κάποιος κατακερματισμός σαν μέρος της επαλήθευσης υπογραφής.

Certificate.getEncoded()

Η μέθοδος αυτή επιστρέφει έναν πίνακα byte που περιέχει την κωδικοποίηση ενός πιστοποιητικού. Το τι αυτό περιέχει, εξαρτάται από το αποτέλεσμα της μεθόδου getType(), αλλά στην περίπτωση του X.509 πιστοποιητικού, ο πίνακας byte που επιστρέφει η `getEncoded`() μέθοδος, θα περιέχει συνήθως το πιστοποιητικό σε κωδικοποίηση DER της ASN.1 δομής του.

X.509 Certificates

Τα πιστοποιητικά που χρησιμοποιήθηκαν στην παρούσα εργασία ήταν τύπου X.509. Η βασική κλάση που τα υλοποιεί είναι η X509Certificate Class.

X509Certificate.getSerialNumber()

Η μέθοδος `getSerialNumber`() επιστρέφει ένα αντικείμενο `BigInteger` που αναπαριστά την τιμή του πεδίου `SerialNumber`. Η τιμή αυτή δεν μπορεί να είναι ούτε `int` ούτε `long`, παρά μόνο `BigInteger`.

Η κλάση `CertificateFactory`

Όπως και άλλα σχήματα στην αρχιτεκτονική JCA, τα αντικείμενα του τύπου `java.security.cert.CertificateFactory` δεν δημιουργούνται αυτόματα, αλλά χρησιμοποιώντας την μέθοδο `getInstance()`.

Η κλάση αυτή περιέχει μεθόδους για την ανάγνωση πιστοποιητικών, λιστών ανακληθέντων πιστοποιητικών και μονοπατιών πιστοποίησης.

`CertificateFactory.generateCertificate()`

Η μέθοδος `generateCertificate()` λαμβάνει σαν όρισμα ένα `InputStream` και εκτελεί ανάγνωση πιστοποιητικών από αυτό. Εάν υπάρχουν περισσότερα από ένα πιστοποιητικά στο `InputStream`, ενώ το τελευταίο υποστηρίζει τις μεθόδους `InputStream.mark()` και `InputStream.reset()` , η μέθοδος `generateCertificate()` θα επιστρέψει ένα πιστοποιητικό ανά κλήση, μέχρι να μην υπάρχει άλλο πιστοποιητικό, όπου επιστρέφει πλέον `null`[9].

5.6.3.2 Το keytool

Το **keytool** είναι ένα εργαλείο διαχείρισης κλειδιών και πιστοποιητικών. Δίνει στο χρήστη τη δυνατότητα να διαχειρίζεται τα δικά του ζεύγη ιδιωτικών/δημόσιων κλειδιών και τα αντίστοιχα πιστοποιητικά με σκοπό την αυτο-πιστοποίηση (όταν, δηλαδή, ο χρήστης δημιουργεί self-signed πιστοποιητικά για να πιστοποιεί τον εαυτό του σε άλλους χρήστες/ υπηρεσίες) ή την ακεραιότητα των δεδομένων και την πιστοποίηση υπηρεσιών, με χρήση ψηφιακών υπογραφών. Επιτρέπει επίσης στο χρήστη να αποκρύπτει τα δημόσια κλειδιά (με τη μορφή πιστοποιητικών) από τους επικοινωνιακούς ομότιμούς τους.

Το πιστοποιητικό είναι μια ψηφιακά υπογεγραμμένη δήλωση από μια οντότητα (πρόσωπο, εταιρία κλπ.) που δηλώνει ότι το δημόσιο κλειδί (και κάποιες άλλες πληροφορίες) κάποιας άλλης οντότητας έχουν μια ορισμένη αξία. Όταν τα δεδομένα υπογράφονται ψηφιακά, η υπογραφή μπορεί να επαληθευτεί, ώστε να ελεγχθεί η ακεραιότητα και πιστοποίηση των δεδομένων. *Ακεραιότητα* σημαίνει ότι τα δεδομένα δεν έχουν τροποποιηθεί ή παραποιηθεί και *πιστοποίηση* σημαίνει ότι πράγματι προέρχονται από την οντότητα που φέρεται να το έχει δημιουργήσει και υπογράψει.

Το keytool αποθηκεύει τα κλειδιά και τα πιστοποιητικά στο λεγόμενο *keystore*, το οποίο υλοποιείται by default ως αρχείο και προστατεύει τα ιδιωτικά κλειδιά με έναν κωδικό. Το keystore είναι στην ουσία μια βάση δεδομένων που χρησιμοποιείται από την JDK εντολή "keytool" και την κλάση KeyStore για την αποθήκευση από το χρήστη των ιδιωτικών κλειδιών και των πιστοποιητικών δημόσιου κλειδιού που έχει παραλάβει από κάποιον άλλον. Υποστηρίζει τα ακόλουθα χαρακτηριστικά:

- Δύο τύπους εισόδων (entries): εισόδους κλειδιού για ιδιωτικά κλειδιά και εισόδους πιστοποιητικού για πιστοποιητικά δημόσιου κλειδιού.
- Μια είσοδος κλειδιού περιλαμβάνει το ιδιωτικό κλειδί και μια αλυσίδα πιστοποιητικών του αντίστοιχου δημόσιου κλειδιού.
- Κάθε είσοδος έχει ένα μοναδικό κωδικό όνομα (alias).
- Οι είσοδοι κλειδιών προστατεύονται από ξεχωριστούς κωδικούς.
- Το keystore μπορεί να έχει διαφορετικές υλοποιήσεις από διαφορετικούς παρόχους πακέτων ασφάλειας (security package providers). Η default υλοποίηση της Sun ονομάζεται JKS.

Το keytool είναι εργαλείο της γραμμής εντολών και προσφέρει διάφορες λειτουργίες, με βάση τις ακόλουθες επιλογές:

- "-genkey": δημιουργεί ένα ζεύγος κλειδιών και το αποθηκεύει ως είσοδο κλειδιού στο keystore.
- "-list": παρουσιάζει σε λίστα όλες τις εισόδους στο keystore.
- "-export": εξάγει το πιστοποιητικό από την προσδιοριζόμενη είσοδο κλειδιού ή πιστοποιητικού του keystore σε αρχείο (certificate file).

- "-printcert": τυπώνει συνοπτικές πληροφορίες για ένα πιστοποιητικό από το αντίστοιχο αρχείο.
- "-import": εισάγει ένα πιστοποιητικό από αρχείο πιστοποιητικού του ως είσοδο πιστοποιητικού στο keystore.
- "-keyclone": δημιουργεί μια νέα είσοδο κλειδιού αντιγράφοντας μια υπάρχουσα.
- "-selfcert": αντικαθιστά το πιστοποιητικό σε μια είσοδο κλειδιού με ένα νέο self-signed πιστοποιητικό.
- "-delete": διαγράφει την είσοδο που αντιστοιχεί στο προσδιοριζόμενο κωδικό όνομα.

Το keytool αυτή τη στιγμή διαχειρίζεται πιστοποιητικά X.509.

Στην παρούσα εργασία δημιουργήθηκαν με το keytool X.509 ψηφιακά πιστοποιητικά, με την παραδοχή βέβαια ότι σε μια πραγματική εφαρμογή θα αντιστοιχούσαν σε πιστοποιητικά εκδοθέντα από μια έμπιστη εκδούσα αρχή (CA). Ακολουθεί παράδειγμα εντολής για τη δημιουργία πιστοποιητικού:

```
keytool -genkey -alias markouk -keystore mariza.jks
Enter keystore password:  240184
What is your first and last name?
  [Unknown]:  Mariza Koukovini
What is the name of your organizational unit?
  [Unknown]:  ntua
What is the name of your organization?
  [Unknown]:  ntua
What is the name of your City or Locality?
  [Unknown]:  Athens
What is the name of your State or Province?
  [Unknown]:  -
What is the two-letter country code for this unit?
  [Unknown]:  GR
Is <CN= Mariza Koukovini, OU= ntua, O= ntua, L= Athens, ST=-,
   C=GR> correct?
  [no]:  yes
Enter key password for <markouk>
         (RETURN if same as keystore password):  My1stKey
```

Με την παρακάτω εντολή το πιστοποιητικό εξάγεται σε αρχείο .crt.

```
keytool -export -alias markouk -file marizakoukovini.crt -
keystore mariza.jks -storepass 240184
```

Με την ακόλουθη εντολή, τέλος, το παραπάνω πιστοποιητικό εξάγεται σε εκτυπώσιμη μορφή στο αρχείο mar.rfc, βασιζόμενο στην RFC 1421 προδιαγραφή και χρησιμοποιώντας τον αλγόριθμο κωδικοποίησης BASE64.

```
keytool -export -alias markouk -file mar.rfc -rfc
   -keystore mariza.jks -storepass 240184
```

Ακολουθεί το αρχείο mar.rfc:

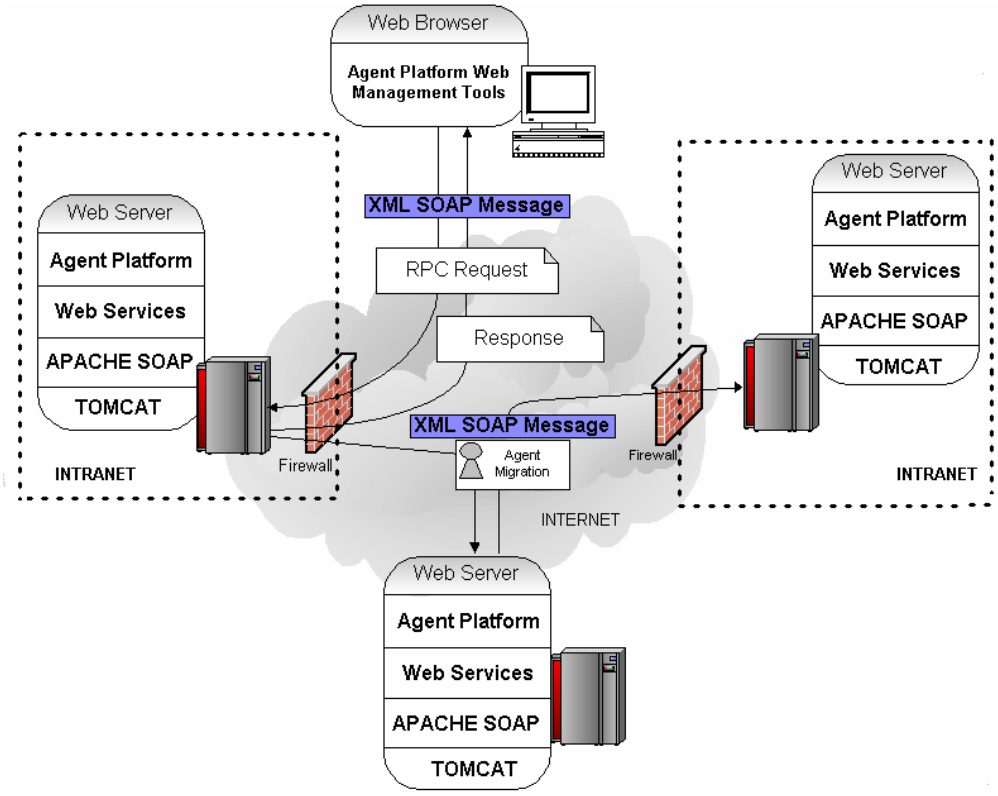
```
-----BEGIN CERTIFICATE-----
MIIDCzCCAsmgAwIBAgIERw4mTDALBgcqhkjOOAQDBQAwaTELMAkGA1UEBhMCZ3
IxEDAOBgNVBAgT
B1Vua25vd24xDzANBgNVBAcTBkF0aGVuczENMAsGA1UEChMEbnR1YTENMAsGA1
UECxMEbnR1YTEZ
MBcGA1UEAxMQbWFyaXphIGtvdWtvdmluaTAeFw0wNzEwMTExMzM0MDRaFw0wOD
AxMDkxMzM0MDRa
MGkxCzAJBgNVBAYTAmdyMRAwDgYDVQQIEwdVbmtub3duMQ8wDQYDVQQHEwZBdG
hlbnMxDTALBgNV
```

BAoTBG50dWExDTALBgNVBAsTBG50dWExGTAXBgNVBAMTEG1hcml6YSBrb3Vrb3
ZpbmkwggG3MIIB
LAYHKoZIzjgEATCCAR8CgYEA/X9TgR11EilS30qcLuzk5/YRt1I870QAwx4/gL
ZRJmlFXUAiUftZ
PY1Y+r/F9bow9subVWzXgTuAHTRv8mZgt2uZUKWkn5/oBHsQIsJPu6nX/rfGG/
g7V+fGqKYVDwT7
g/bTxR7DAjVUE1oWkTL2dfOuK2HXKu/yIgMZndFIAccCFQCXYFCPFSMLzLKSuY
Ki64QL8Fgc9QKB
gQD34aCF1ps93su8q1w2uFe5eZSvu/o66oL5V0wLPQeCZ1FZV4661FlP5nEHEI
GAtEkWcSPoTCgW
E7fPCTKMyKbhPBZ6i1R8jSjgo64eK7OmdZFuo38L+iE1YvH7YnoBJDvMpPG+qF
GQiaiD3+Fa5Z8G
kotmXoB7VSVkAUw7/s9JKgOBhAACgYA24SwwqCHjdacPIePzSFMZX/3gV9PTkH
qPIDNHuwr83HHE
rZQcIPjkAYwPFrYNb8A2EtD6Xj3wHu6XMCacSTsLj4B8PjLyKlE9jCW3FZkfGX
4etMvT9XNd80GG
7hu0AtDcVFu+TFbsvfOdVl9M8zWXBc8hMXUJAO7DjkFDHJ3PvjALBgcqhkjOOA
QDBQADLwAwLAIU
QLStaTduN9Yr9Vm2HETtqktVlygCFB/B72OK0fRJorrmYOMaxoOQy7Um
-----END CERTIFICATE-----

[5][6]

5.7 Πλατφόρμα κινητών πρακτόρων

Η υποκείμενη πλατφόρμα, πάνω στην οποία αναπτύσσεται η αρχιτεκτονική της παρούσας διπλωματικής εργασίας, βασίζεται στις υπηρεσίες του διαδικτύου και συγκεκριμένα στο μοντέλο των υπηρεσιών ιστού και ειδικότερα στο πρωτόκολλο SOAP, όπως αυτό χρησιμοποιείται στον παγκόσμιο ιστό. Ξεκινά από ήδη διαθέσιμες υποδομές διότι προσδίδει τις λειτουργίες των κινητών πρακτόρων σε ήδη υπάρχοντες διακομιστές διαδικτύου. Τα συστατικά της πλατφόρμας αναπτύσσονται σε διακομιστές διαδικτύου *Apache* χρησιμοποιώντας την βιβλιοθήκη *Apache SOAP* και αποτελείται από ένα σύνολο προγραμματιστικών διεπαφών και ένα σύνολο από σελίδες για την διαχείριση της [12].



Σχήμα 5.2 Η γενική αρχιτεκτονική της πλατφόρμας κινητών πρακτόρων SOAP

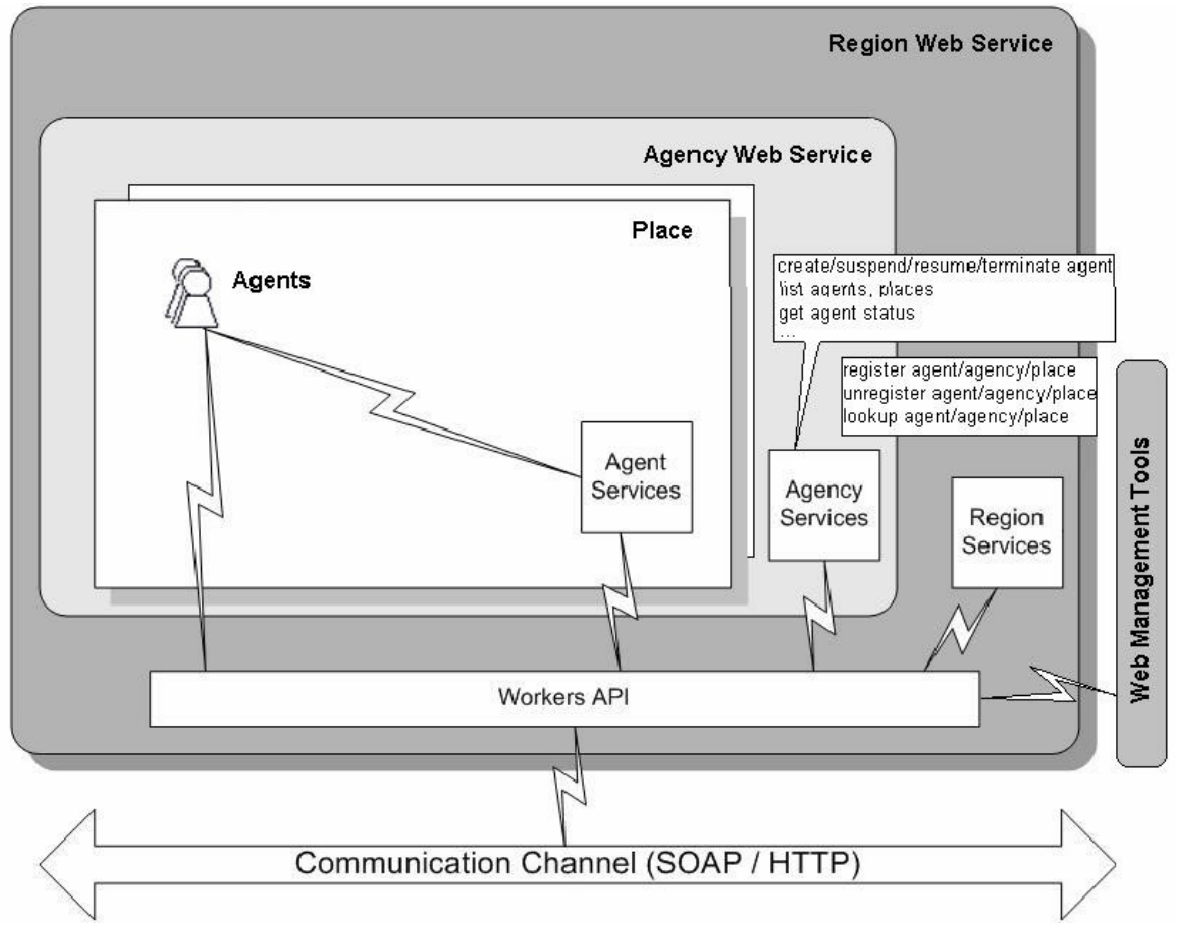
Οι συνιστώσες της πλατφόρμας υλοποιήθηκαν εξ' ολοκλήρου χρησιμοποιώντας τη γλώσσα Java και αποτελούνται από τον κύριο πυρήνα συνιστωσών στην πλευρά του εξυπηρετητή και από τα διαχειριστικά εργαλεία στην πλευρά του πελάτη. Τα τελευταία χρόνια η Java έχει αποδειχτεί ως η πλέον κατάλληλη για την υλοποίηση μιας πλατφόρμας κινητών πρακτόρων εξαιτίας της πολυνηματικής επεξεργασίας, της σειριοποίησης αντικειμένων και των δικτυακών διεπαφών που προσφέρει.

Όλες οι συνιστώσες λογισμικού εγκαθίστανται σε *TOMCAT* υποδοχείς *servlets*. Η πλατφόρμα εκμεταλλεύεται τις κλήσεις *SOAP-RPC* χρησιμοποιώντας τη βιβλιοθήκη *Apache SOAP*. Το *SOAP* επιτρέπει επίσης την εγκατάσταση της πλατφόρμας πρακτόρων στα δίκτυα που ασφαλίζονται από διακομιστές *firewall*. Η πλατφόρμα πρακτόρων αποτελείται από δύο μέρη, την πλευρά εξυπηρετητή και την πλευρά πελάτη.

5.7.1 Η πλευρά του εξυπηρετητή

Η πλευρά του εξυπηρετητή περιέχει τα κύρια συστατικά της πλατφόρμας. Όπως φαίνεται στο Σχήμα 5.3, έχουμε υιοθετήσει τις έννοιες των προδιαγραφών MASIF (Mobile Agent System Interoperability Facility), των θέσεων (places), των αντιπροσωπειών (agencies) και των περιοχών (regions) που χρησιμοποιούνται επίσης σε διάφορες υπάρχουσες πλατφόρμες πρακτόρων. Η βασική διαφορά είναι ότι τώρα η αντιπροσωπεία και η περιοχή είναι υπηρεσίες Ιστού που εκθέτουν τις μεθόδους σε άλλες συνιστώσες λογισμικού της πλατφόρμας.

Κάθε διακομιστής ιστού περιέχει δύο κύρια αντικείμενα, της αντιπροσωπεία και την περιοχή, με την ύπαρξη της τελευταίας να μην είναι απαραίτητη. Η τρέχουσα υλοποίηση της πλατφόρμας επιτρέπει μόνο μια αντιπροσωπεία σε κάθε διακομιστή ιστού.



Σχήμα 5.3 Οι λογικές συνιστώσες της πλατφόρμας των πρακτόρων

Η υπηρεσία ιστού "περιοχή" χρησιμοποιείται ως υπηρεσία καταλόγου. Η δημιουργία, μετακίνηση καθώς και κάθε κύρια λειτουργία ενός πράκτορα κατά τον κύκλο ζωής του έχει ως αποτέλεσμα την ενημέρωση της περιοχής από την αντιπροσωπεία που έχει καταχωρηθεί στα μητρώα της. Επιπρόσθετα, μια αντιπροσωπεία ή ένας πράκτορας (διαμέσου της αντιπροσωπείας) μπορούν να αναζητήσουν πληροφορία από την περιοχή για την διαθεσιμότητα άλλων αντιπροσωπειών ή τη θέση άλλων πρακτόρων κ.τ.λ. με χρήση ειδικών φίλτρων αναζήτησης.

Οι υπηρεσίες ιστού "αντιπροσωπείες" αποτελούν τις κύριες συνιστώσες λογισμικού της πλατφόρμας, παρέχοντας το περιβάλλον εκτέλεσης και τη δίοδο επικοινωνίας για άλλες αντιπροσωπείες και πράκτορες. Μια υπηρεσία ιστού "αντιπροσωπεία" είναι ικανή να δεχτεί είτε κινητούς είτε στατικούς πράκτορες. Ο κύριος σκοπός της είναι η δημιουργία, αποθήκευση, μετακίνηση πρακτόρων και το να παρέχει υπηρεσίες επικοινωνίας μεταξύ τους. Κάθε αντιπροσωπεία χωρίζεται σε τοποθεσίες (*places*). Κάθε τοποθεσία ομαδοποιεί τη λειτουργικότητα στην αντιπροσωπεία ενθυλακώνοντας ορισμένες δυνατότητες και περιορισμούς σε επισκεπτόμενους πράκτορες, βοηθώντας έτσι τον προγραμματιστή να διαχωρίσει τους πράκτορες μεταξύ τους.

5.7.2 Η πλευρά του πελάτη

Η πλευρά πελάτη της πλατφόρμας *SOAP* ελέγχεται από διάφορες σελίδες *JSP* (*Java Server Pages*), οι οποίες επιτρέπουν την απομακρυσμένη διαχείριση των περιοχών, των αντιπροσωπειών και των πρακτόρων. Οι διαχειριστικές υπηρεσίες επιτρέπουν την παρακολούθηση και τον έλεγχο των πρακτόρων και τις θέσεις (*places*) μιας αντιπροσωπείας από τους χρήστες. Είναι πιθανό, μεταξύ άλλων, να δημιουργηθούν, να διαγραφούν και να ανασταλούν πράκτορες, υπηρεσίες και θέσεις, να ανακτηθούν πληροφορίες για

συγκεκριμένους πράκτορες και υπηρεσίες, να εμφανιστεί λίστα με όλους τους πράκτορες που κατοικούν σε μια συγκεκριμένη θέση κ.τ.λ.

5.7.3 Φόρτωση και εκτέλεση του κινητού πράκτορα

Ο προγραμματιστής υλοποιεί μια εφαρμογή κινητών πρακτόρων βασιζόμενος στο σετ των κλάσεων του *API* της πλατφόρμας. Μία από τις κλάσεις αυτές θα πρέπει οπωσδήποτε να επεκτείνει την κλάση *StationaryAgent* ή *MobileAgent* και το σύνολο των κλάσεων θα πρέπει να συμπιεστεί στη μορφή *JAR* της *Java*. Με χρήση μιας *HTML* σελίδας, το αρχείο και το όνομα της βασικής κλάσης του πράκτορα αποστέλλονται σε μια σελίδα *JSP*. Η *JSP* σελίδα από τη μεριά της δρομολογεί μια *SOAP RPC* αίτηση στην αντιπροσωπεία για τη δημιουργία του πράκτορα. Έπειτα η αντιπροσωπεία φορτώνει την κλάση του πράκτορα και δημιουργεί ένα ενθυλακωτή γι' αυτόν. Τέλος, δημιουργείται ένα νέο *ThreadGroup* το οποίο αναλαμβάνει την εκτέλεση του πράκτορα.

Η υποδομή της πλατφόρμας παρέχει την υποστήριξη υπηρεσιών κινητικότητας και επικοινωνίας κατά την ώρα εκτέλεσης και γι' αυτό το λόγο κατάλληλες λειτουργίες είναι διαθέσιμες στη γλώσσα υλοποίησης της πλατφόρμας. Έτσι είναι δυνατή η μετανάστευση ενός πράκτορα από ένα διακομιστή σε ένα άλλο ώστε να μπορεί να έχει πρόσβαση σε δεδομένα και πληροφορίες τις οποίες να μεταφέρει πίσω στην αρχική του θέση.

Οι κινητοί πράκτορες είναι οντότητες λογισμικού, οι οποίες μπορούν αυτόνομα να μεταναστεύσουν από έναν εξυπηρετητή σε κάποιον άλλο κατά τη διάρκεια της εκτέλεσής τους. Αποτελούνται από τον κώδικα, τα δεδομένα καθώς επίσης και από την κατάσταση εκτέλεσής τους. Η μεταφορά της κατάστασης εκτέλεσης επιτρέπει σε έναν αντιπρόσωπο να συνεχίσει την εργασία του αμέσως μόλις μεταναστεύσει προς έναν άλλο εξυπηρετητή, από το σημείο όπου είχε σταματήσει πριν τη μετανάστευση. Όσον αφορά την Java, η εικονική μηχανή της δεν είναι ικανή να ανακτήσει το σωρό εκτέλεσης ενός νήματος. Για αυτόν τον λόγο, σε περίπτωση μετανάστευσης ενός πράκτορα μόνο οι ψηφιολέξεις (bytecodes) της κλάσης και τα δεδομένα του στιγμιότυπου της κλάσης μπορούν να μεταφερθούν στο απομακρυσμένο εξυπηρετητή δίχως το σωρό εκτέλεσης. Μια λύση για αυτό το πρόβλημα είναι οι κινητοί πράκτορες να αποθηκεύουν τις απαραίτητες πληροφορίες κατάστασης σε μεταβλητές μελών της κλάσης, που επιτρέπουν σε μια μέθοδο που καθορίζεται ως σημείο εισόδου να εξετάσει αυτές τις μεταβλητές και να προχωρήσει ανάλογα με την κατάσταση στην οποία βρίσκονται οι υπολογισμοί. Αυτό το σχήμα αναφέρεται ως "ασθενής μετανάστευση" σε αντίθεση με τη σχήμα της "ισχυρής μετανάστευσης" όπου η μεταφορά του σωρού εκτέλεσης επιτρέπει στην υποδομή να επαναστιγμιοτυπήσει τον πράκτορα από το σημείο που είχε σταματήσει την εκτέλεση προτού κληθεί η μέθοδος "move" στον προηγούμενο εξυπηρετητή.

Το τρίτο μέρος ενός κινητού πράκτορα αποτελείται από τις ιδιότητες, οι οποίες περιγράφουν τις απαιτήσεις και την ιστορία του στην υποδομή. Αυτό περιλαμβάνει στοιχεία όπως ένα μοναδικό προσδιοριστή, τον ιδιοκτήτη του πράκτορα ως μια διεύθυνση για τα ενδιάμεσα αποτελέσματα, μηνύματα λάθους για τη σωστή συμπεριφορά ενός πράκτορα, τον τόπο και χρόνο προέλευσης και το ιστορικό των μετακινήσεών του. Αυτές οι πληροφορίες μπορούν να προσαρμοστούν σύμφωνα με την προέλευση ή το σκοπό του εκάστοτε πράκτορα. Το πλεονέκτημα αυτού του σχήματος είναι ότι κανένας εξυπηρετητής δεν χρειάζεται τις πλήρεις πληροφορίες για όλους τους πιθανούς προορισμούς για τους κινητούς πράκτορες.

Τέλος, χρησιμοποιείται μια υπηρεσία καταλόγου, ώστε να ανακτάται η τρέχουσα θέση ενός πράκτορα.

Μηχανισμοί επικοινωνίας-Κινητικότητα

Η επικοινωνία μεταξύ των πρακτόρων με ανταλλαγή *SOAP* μηνυμάτων ακολουθεί τρία διαφορετικά μοντέλα: σύγχρονη, ασύγχρονη και call-back-call.

Οι κινητοί πράκτορες σε αυτή την πλατφόρμα διαχωρίζονται σε δύο κύριες κατηγορίες: στους κινητούς και στους στατικούς. Ενώ οι τελευταίοι παραμένουν στην αντιπροσωπεία που τους δημιούργησε μέχρι το τέλος της εκτέλεσής τους, οι πρώτοι έχουν την ικανότητα να μεταναστεύουν σε απομακρυσμένες αντιπροσωπείες αυτόνομα ή να τους δοθεί εντολή γι' αυτό. Αυτή η μεταφορά επιτυγχάνεται και από τις δύο αντιπροσωπείες με διαπραγμάτευση και εάν είναι δυνατή η μεταφορά, τότε την πραγματοποιούν.

Και οι δύο τύποι αντιπροσώπων είναι απόγονοι της κλάσης Agent. Στην κλάση αυτή ορίζονται τρεις μέθοδοι: η init, η start και η destroy. Ο κύκλος ζωής ενός αντιπροσώπου ακολουθεί την παραπάνω σειρά, δηλαδή πρώτα καλείται η init, μετά η start, και τέλος η destroy. Κάθε αντιπρόσωπος μπορεί να υλοποιήσει μία, μερικές ή όλες αυτές τις μεθόδους. Ο σκοπός ύπαρξης της κάθε μίας είναι συγκεκριμένος. Η init υπάρχει για να μπορεί να γίνει κάποια αρχικοποίηση του αντιπροσώπου. Αν και η αρχικοποίηση μπορεί να γίνει και στον constructor, είναι προτιμότερο να πραγματοποιείται στην init. Η start περιέχει το «χρήσιμο» κώδικα του αντιπροσώπου, τον κώδικα δηλαδή που αναλαμβάνει να εκτελέσει την εργασία για την οποία έχει δημιουργηθεί ο αντιπρόσωπος. Τέλος, η destroy καλείται αμέσως μόλις τελειώσει η start, και χρησιμοποιείται για οποιαδήποτε ανάγκη υπάρχει στο τέλος της εκτέλεσης του αντιπροσώπου (π.χ. αποδέσμευση μνήμης). Μια σημαντική λεπτομέρεια είναι ότι μετά το τέλος της destroy, ο αντιπρόσωπος συνεχίζει να υπάρχει στην αντιπροσωπεία, με τη διαφορά ότι τώρα θεωρείται ανενεργός [12].

5.8 Βιβλιογραφία

[1] "JavaTMCryptography Architecture (JCA) Reference Guide, for JavaTM Platform Standard Edition 6", available: http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html.

[2] David Hook, "Beginning Cryptography with Java™", Wiley Publishing, Inc., Indianapolis, Indiana, 2005, pp 1-4.

[3] Wikipedia, The free encyclopedia, "MySQL" (article), available: http://en.wikipedia.org/wiki/MySQL.

[4] "MySQL Reference Manual", available: http://orgs.man.ac.uk/documentation/mysql/manual_toc.html.

[5] "keytool-Key and Certificate Management Tool", available: http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html.

[6] Dr. Herong Yang, "JCA-Certificates, 'keytool' and 'keystore'", JDK Tutorials-Herong's Tutorial Notes, version 4.32, 1997-2006, available: http://www.herongyang.com/jdk/.

[7] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Συστήματα Βάσεων Δεδομένων", εκδόσεις Μ. Γκιούρδας, Αθήνα, 2004.

[8] David Hook, "Beginning Cryptography with Java™", Wiley Publishing, Inc., Indianapolis, Indiana, 2005, pp 121-123, 128.

[9] David Hook, "Beginning Cryptography with Java™", Wiley Publishing, Inc., Indianapolis, Indiana, 2005, pp 187-204.

[10] Marty Hall, Larry Brown, "Core Servlets and Java Server Pages", Second Edition, available: http://pdf.coreservlets.com/, pp. 9-12.

[11] Γ. Ι. Στασινόπουλος, "Διαδίκτυο και Εφαρμογές: Βασικές Έννοιες", Αθήνα, 2007, pp. 24-26.

[12] Ιωάννης Ε. Φουκαράκης, "Ανάπτυξη υποδομής κινητών πρακτόρων με χρήση τεχνολογιών διαδικτύου", Διπλωματική Εργασία, ΕΜΠ, Αθήνα, Νοέμβριος 2003, pp. 33-40.

6 Ανάλυση απαιτήσεων και σχεδιασμός συστήματος

6.1 Ανάλυση απαιτήσεων

Αρχική προϋπόθεση είναι η ψηφοφορία να γίνεται μέσα στο έγκυρο χρονικό διάστημα. Διαφορετικά, όταν ο ψηφοφόρος ζητήσει από το τερματικό την αρχική σελίδα της εφαρμογής, ενημερώνεται από το σύστημα ότι δεν μπορεί να ψηφίσει, επειδή έχει προσέλθει είτε πριν την έναρξη είτε μετά το πέρας της ψηφοφορίας.

Κάθε ψηφοφόρος, προκειμένου να έχει δικαίωμα ψήφου, έχει μαζί του μια «έξυπνη κάρτα» (smart card) που περιέχει το πιστοποιητικό που του δίνει την απαραίτητη εξουσιοδότηση. Θεωρούμε ότι έχει εκδοθεί από μια έμπιστη εκδούσα αρχή ειδικά γι αυτό το σκοπό. Κάθε τέτοιο πιστοποιητικό έχει ένα μοναδικό σειριακό αριθμό (serial number), με βάση τον οποίο γίνεται η αντιστοίχιση με το συγκεκριμένο ψηφοφόρο και η καταχώρησή του στο σύστημα. Σε περίπτωση μη ύπαρξης πιστοποιητικού, η διαδικασία δεν προχωράει και με τον τρόπο αυτό, διασφαλίζεται η εξουσιοδοτημένη υποβολή ψήφου.

Σαν πρόσθετη δικλείδα ασφαλείας, υπάρχει αντιστοίχιση κάθε πιστοποιητικού με συγκεκριμένο όνομα χρήστη και κωδικό (username, password). Έτσι, για παράδειγμα, κάποιος που έχει στην κατοχή του «παράνομα» ένα έγκυρο πιστοποιητικό, δεν θα μπορεί να το χρησιμοποιήσει. Τα στοιχεία αυτά βρίσκονται καταχωρημένα τόσο στις βάσεις δεδομένων των εκλογικών κέντρων, όσο και στην κεντρική βάση δεδομένων. Κάθε φορά που κάποιος επιχειρεί να ψηφίσει, γίνεται έλεγχος των παραπάνω στοιχείων στην τοπική βάση δεδομένων και, εφόσον υπάρχει ταίριασμα, γίνεται καταχώρηση του γεγονότος ότι έχει ψηφίσει, ώστε να μην έχει τη δυνατότητα να ξαναψηφίσει. Κάθε τοπική βάση δεδομένων ενημερώνει ανά τακτά χρονικά διαστήματα την κεντρική, η οποία με τη σειρά τις ενημερώνει και τις υπόλοιπες περιφερειακές βάσεις. Έτσι διασφαλίζεται ότι κάποιος που έχει ήδη ψηφίσει σε ένα εκλογικό κέντρο, δεν θα έχει τη δυνατότητα να ξαναψηφίσει και σε κάποιο άλλο εκλογικό κέντρο. Θεωρούμε ότι τα διαστήματα αυτά είναι επαρκώς μικρά, ώστε να μην προλαβαίνει κανείς να μεταβεί από ένα εκλογικό κέντρο σε κάποιο άλλο. Τέλος, σε περίπτωση μη εισαγωγής των παραπάνω διαπιστευτηρίων, η διαδικασία «κολλάει», μέχρι αυτά να εισαχθούν.

Σε ό,τι αφορά τη διαδικασία μεταφοράς του μηνύματος, αυτό θα πρέπει καταρχάς να κρυπτογραφείται (RSA αλγόριθμος), ώστε να αποτραπεί η ανάγνωσή του κατά τη μεταφορά. Αφού κρυπτογραφηθεί, υπογράφεται από τον τοπικό εξυπηρετητή και όταν το μήνυμα φτάσει στον κεντρικό εξυπηρετητή, πριν την αποκρυπτογράφησή του, γίνεται επαλήθευση της υπογραφής. Έτσι, διαπιστώνεται η ακεραιότητα ή μη του μηνύματος, καθώς και το αν έχει αποσταλεί πράγματι από εξυπηρετητή του συστήματος ηλεκτρονικής ψηφοφορίας.

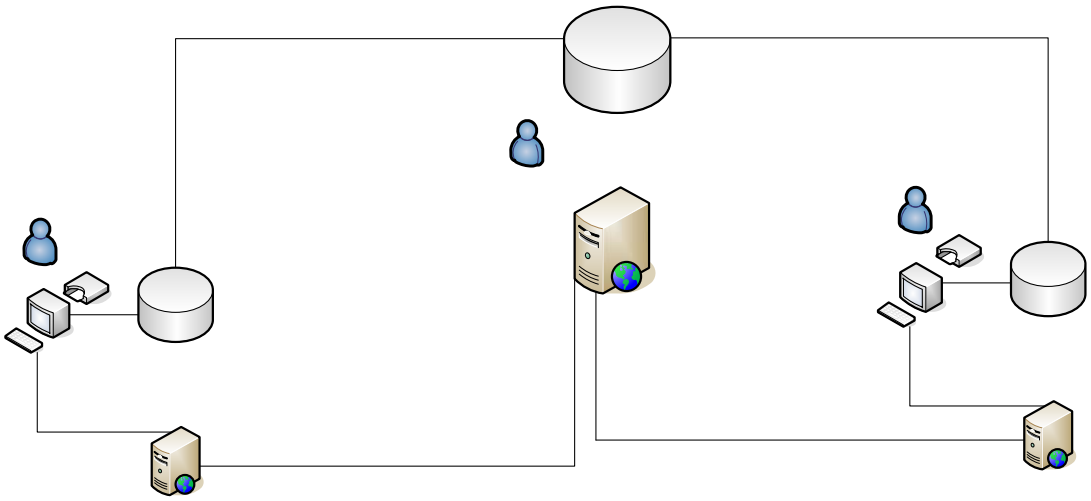
Αξίζει να σημειωθεί ότι το μήνυμα μεταφέρει μόνο την ψήφο, και κανένα άλλο στοιχείο που να σχετίζεται με την ταυτότητα του ψηφοφόρου, ώστε να μην είναι δυνατή η αντιστοίχηση σε κανένα στάδιο της διαδικασίας και να διασφαλίζεται το απόρρητο της ψήφου. Εξάλλου, ο σειριακός αριθμός του πιστοποιητικού κάθε ψηφοφόρου μεταφέρεται από διαφορετικό «κανάλι» και σε εντελώς ασυσχέτιστες χρονικές στιγμές σε σχέση με την ψήφο.

Απαραίτητη παραδοχή για τη συνολική λειτουργία του συστήματος είναι η μη ανάμειξη ανθρώπινου παράγοντα στη διαχείρισή του, τόσο στην κεντρική όσο και στις περιφερειακές μονάδες, ώστε ενδεχόμενη κακόβουλη παρέμβαση να μην αφορά τα σημεία διαχείρισης.

6.2 Σχεδίαση του συστήματος

6.2.1 Βασική αρχιτεκτονική

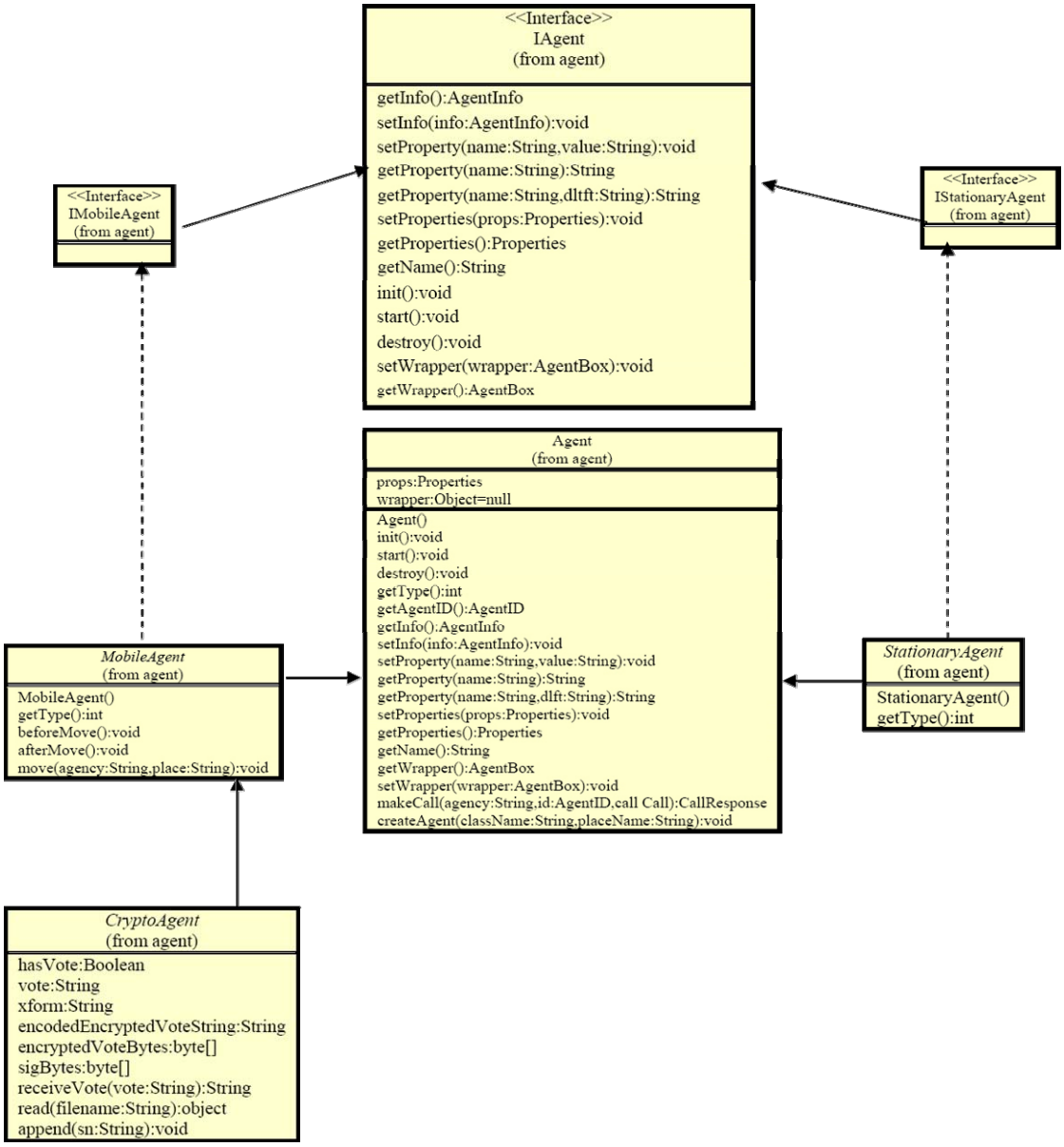
Ξεκινάμε με την παρουσίαση της βασικής αρχιτεκτονικής του συστήματος ηλεκτρονικής ψηφοφορίας.



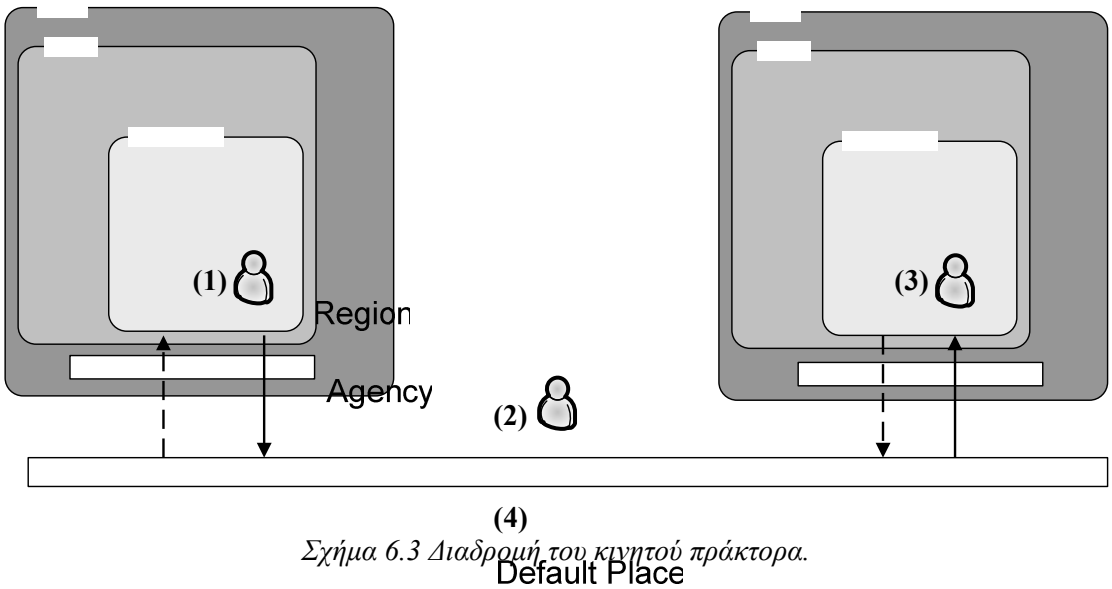
Σχήμα 6.1 Γενική αρχιτεκτονική συστήματος.

Όπως φαίνεται παραπάνω, το κατανεμημένο σύστημά μας περιλαμβάνει ένα κεντρικό υπολογιστικό σύστημα, αποτελούμενο κατά βάση από ένα εξυπηρετή και μία βάση δεδομένων, καθώς και από πολλά περιφερειακά συστήματα, τα οποία αντιστοιχούν στα εκλογικά κέντρα. Το κεντρικό σύστημα αναλαμβάνει τη συγκέντρωση των ψήφων, ενώ βρίσκεται σε διαρκή επικοινωνία με τα περιφερειακά. Τα περιφερειακά συστήματα των εκλογικών κέντρων είναι αυτά στα οποία οι ψηφοφόροι, μέσω τερματικών συσκευών, καταθέτουν την ψήφο τους και περιλαμβάνουν επιπλέον εξυπηρετητή web και βάση δεδομένων.

Η μεταφορά της ψήφου γίνεται με έναν κινητό πράκτορα, τον CryptoAgent, η κλάση του οποίου αποτελεί επέκταση της κλάσης MobileAgent της πλατφόρμας κινητών πρακτόρων. Στη μέθοδο start() του πράκτορα επιτελούνται επιπλέον οι λειτουργίες της κρυπτογράφησης, αποκρυπτογράφησης, δημιουργίας και ελέγχου γνησιότητας της ψηφιακής υπογραφής. Στο παρακάτω σχήμα παρουσιάζεται η ιεραρχία των κλάσεων των πρακτόρων με βάση τη γλώσσα *UML* (*Unified Modelling Language*).



Σχήμα 6.2 Uml διάγραμμα των κλάσεων του πράκτορα



Σχήμα 6.3 Διαδρομή του κινητού πράκτορα.

Στο παραπάνω σχήμα βλέπουμε τα διάφορα στάδια της διαδρομής κάθε πράκτορα. Στη θέση **(1)** ο κινητός πράκτορας βρίσκεται ακόμα στην αντιπροσωπεία του τοπικού εξυπηρετητή. Όταν γίνει αναζήτησή του και κληθεί με σύγχρονο τρόπο από την εφαρμογή, διαβάζει την ψήφο, την κρυπτογραφεί, την υπογράφει και ξεκινά για τον απομακρυσμένο εξυπηρετητή πάνω από SOAP/HTTP(θέση **(2)**). Αφού φτάσει στη νέα αντιπροσωπεία, επαληθεύει την υπογραφή, αποκρυπτογραφεί την ψήφο και την καταχωρεί στον πίνακα στον οποίο συγκεντρώνονται όλες οι ψήφοι (θέση **(3)**). Στη συνέχεια ο πράκτορας επιστρέφει στην αρχική αντιπροσωπεία πάνω από τον ίδιο δίαυλο, ώστε να ξαναχρησιμοποιηθεί με τον ίδιο τρόπο **(4)**.

CryptoAgent

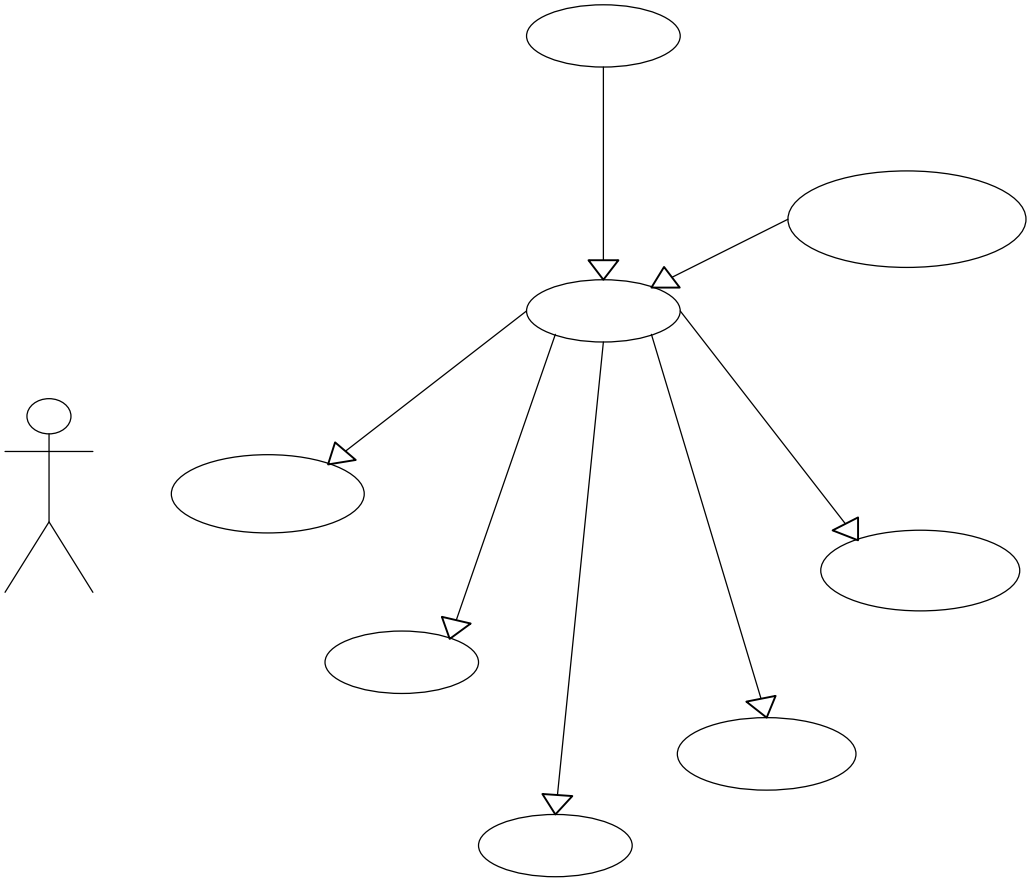
Workers

Crypto

Communication C

6.2.2 Περιπτώσεις χρήσης

Διακρίνουμε τις εξής 2 περιπτώσεις χρήσης:
Α)Ψηφοφορία



Σχήμα 6.4 Use case διάγραμμα-ψηφοφορία

Ο ψηφοφόρος προσέρχεται στο εκλογικό κέντρο και ζητά την αρχική σελίδα της εφαρμογής (index.jsp). Η διαδικασία προχωρά, εφόσον βρισκόμαστε εντός έγκυρου χρονικού διαστήματος ψηφοφορίας, στην εισαγωγή του πιστοποιητικού από το χρήστη. Με σύνδεση στην τοπική βάση δεδομένων, ελέγχεται αν το συγκεκριμένο άτομο έχει ήδη ψηφίσει, με βάση τον σειριακό αριθμό του πιστοποιητικού και την τιμή της boolean μεταβλητής has_voted. Αν το πιστοποιητικό που εισάγεται είναι έγκυρο, δηλαδή ο σειριακός του αριθμός είναι καταχωρημένος και το άτομο δεν έχει ξαναψηφίσει (has_voted=0), ζητείται η εισαγωγή username και password. Ο σειριακός αριθμός του πιστοποιητικού φυλάσσεται σ' ένα προσωρινό πίνακα, η χρήση του οποίου θα εξηγηθεί παρακάτω. Πραγματοποιείται νέος έλεγχος στη βάση για τη αντιστοίχηση των τελευταίων με το σειριακό αριθμό κι εφόσον αυτή εξακριβωθεί, ζητείται από τον ψηφοφόρο να επιλέξει ανάμεσα σε 3 υποψήφιους. Σε τελικό στάδιο γίνεται κλήση του κινητού πράκτορα από την υποκείμενη πλατφόρμα, στον οποίο πράκτορα φορτώνεται το μήνυμα προς μεταφορά, δηλαδή η ψήφος. Ο πράκτορας κρυπτογραφεί την ψήφο με το δημόσιο κλειδί, που το κεντρικό σύστημα έχει διανείμει στα περιφερειακά, το υπογράφει με το ιδιωτικό κλειδί του συγκεκριμένου περιφερειακού συστήματος και ξεκινά τη μετακίνηση του προς την αντιπροσωπεία που βρίσκεται στον κεντρικό εξυπηρετητή. Εκεί ο πράκτορας ελέγχει τη γνησιότητα της υπογραφής και, αφού αυτή επαληθευτεί, αποκρυπτογραφεί την ψήφο και την καταχωρεί.

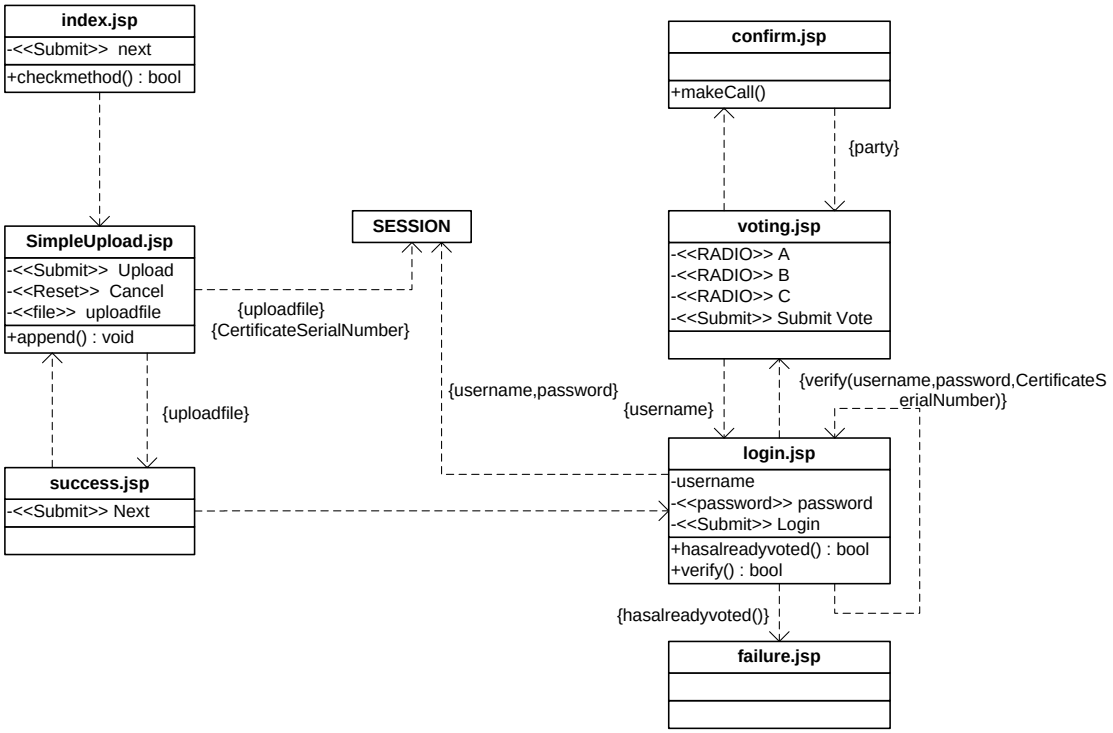
«uses»

«us

Upload Certificate

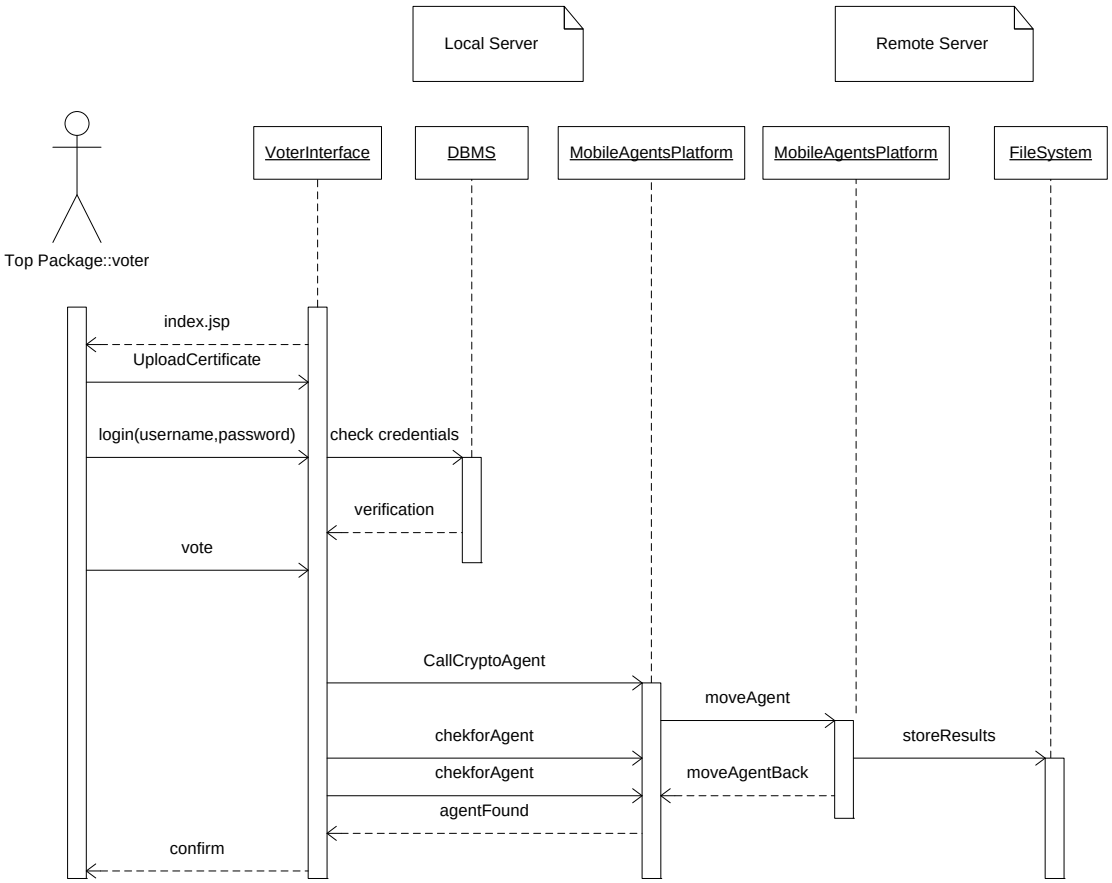
Voter

Login



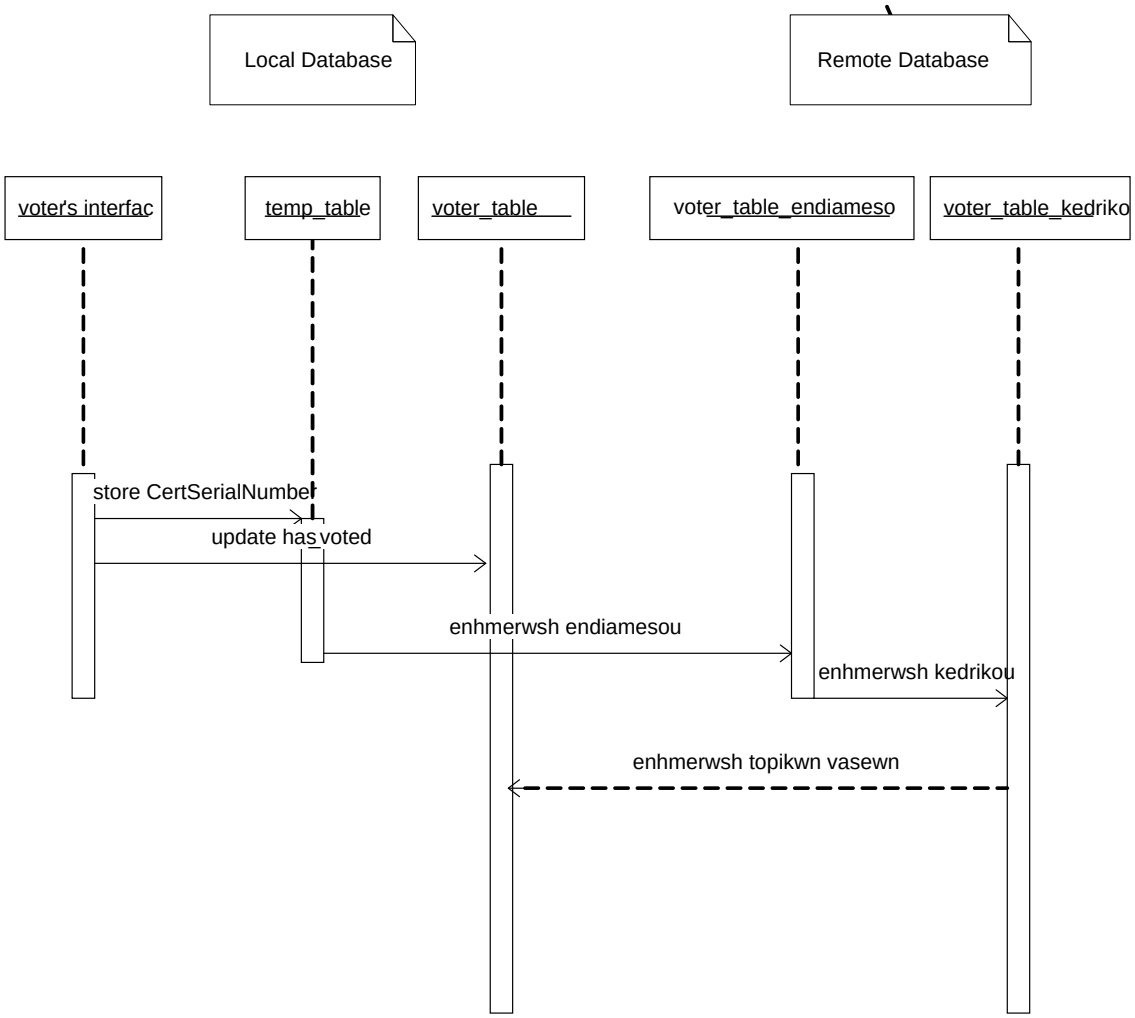
Σχήμα 6.5 Uml διάγραμμα για τις JSP.

Παρακάτω βλέπουμε και το ακολουθιακό uml διάγραμμα της διαδικασίας της ψηφοφορίας.

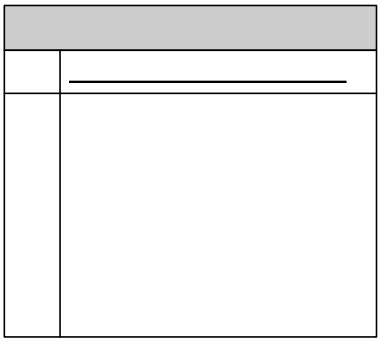


Σχήμα 6.6 Ακολουθιακό διάγραμμα.

Ιδιαίτερα σημαντικός στην όλη διαδικασία είναι και ο ρόλος του συστήματος διαχείρισης βάσεων δεδομένων, γι αυτό και θα μελετηθεί ξεχωριστά. Κάθε φορά που κάποιος ψηφίζει, ο σειριακός αριθμός του πιστοποιητικού του καταχωρείται σε ένα προσωρινό πίνακα στο τοπικό σύστημα. Από την έναρξη της λειτουργίας του συνολικού συστήματος, «τρέχουν» δύο κλάσεις: η μία στον τοπικό υπολογιστή, η οποία ανά 50 δευτερόλεπτα διαβάζει τον παραπάνω πίνακα, καταχωρεί τα καταγεγραμμένα σε αυτόν άτομα που ψήφισαν σε αυτά τα 50 δευτερόλεπτα (με βάση τους σειριακούς αριθμούς των πιστοποιητικών) σε έναν άλλο ενδιάμεσο προσωρινό πίνακα στην κεντρική βάση δεδομένων και στη συνέχεια σβήνει τον πρώτο πίνακα. Η δεύτερη «τρέχει» στον κεντρικό υπολογιστή και κάθε 60 δευτερόλεπτα (ώστε να έχει προλάβει να ολοκληρωθεί η προηγούμενη ενημέρωση) ενημερώνει τόσο την κεντρική βάση όσο και τις περιφερειακές με βάση τα εκάστοτε δεδομένα του ενδιάμεσου πίνακα και τέλος σβήνει τον τελευταίο. Ο σκοπός των προσωρινών πινάκων είναι να αποφευχθεί η μετακίνηση μεγάλου και περιττού όγκου δεδομένων πάνω από το δίκτυο.



Σχήμα 6.7 Ακολουθιακό διάγραμμα που απεικονίζει το συγχρονισμό και την επικοινωνία στο σύστημα διαχείρισης βάσεων δεδομένων.



Σχήμα 6.8 Διάγραμμα σχήματος περιφερειακών βάσεων δεδομένων.

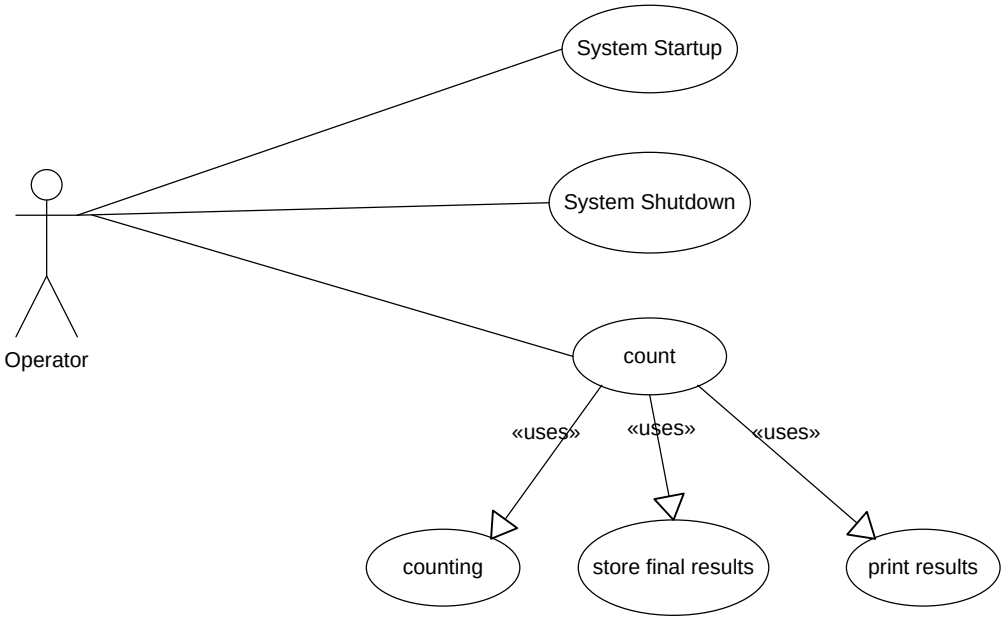
Σχήμα 6.9 Διάγραμμα σχήματος κεντρικής βάσης δεδομένων.

voter_table_endiameso

PK serial_num:VARCHAR(45)

has_voted:TINYINT(3)

Β) Καταμέτρηση ψήφων



Σχήμα 6.10 Use case διάγραμμα για την καταμέτρηση

Αφού ολοκληρωθεί η διαδικασία της ψηφοφορίας, οι ψήφοι που έχουν συγκεντρωθεί καταμετρούνται από την κλάση final_counter και εμφανίζονται συγκεντρωτικά.

7 Πιθανά σενάρια

Ξεκινώντας η διαδικασία υπάρχουν δύο ειδών πίνακες βάσεων δεδομένων. Το πρώτο βρίσκεται στα τοπικά συστήματα και περιλαμβάνει 9 στήλες για κάθε ψηφοφόρο: το serial_num του πιστοποιητικού, username, password, has_voted με τιμή 0, εφόσον κανείς δεν έχει ψηφίσει ακόμα, name, surname, father's name, birth_date και id_number. Όλα τα τοπικά συστήματα έχουν πανομοιότυπο πίνακα που περιλαμβάνουν όλους τους ψηφοφόρους κι έτσι όλοι μπορούν να ψηφίσουν σε οποιοδήποτε εκλογικό τμήμα. Το δεύτερο βρίσκεται στην κεντρική βάση δεδομένων και περιλαμβάνει μόνο serial_num και has_voted για κάθε ψηφοφόρο.

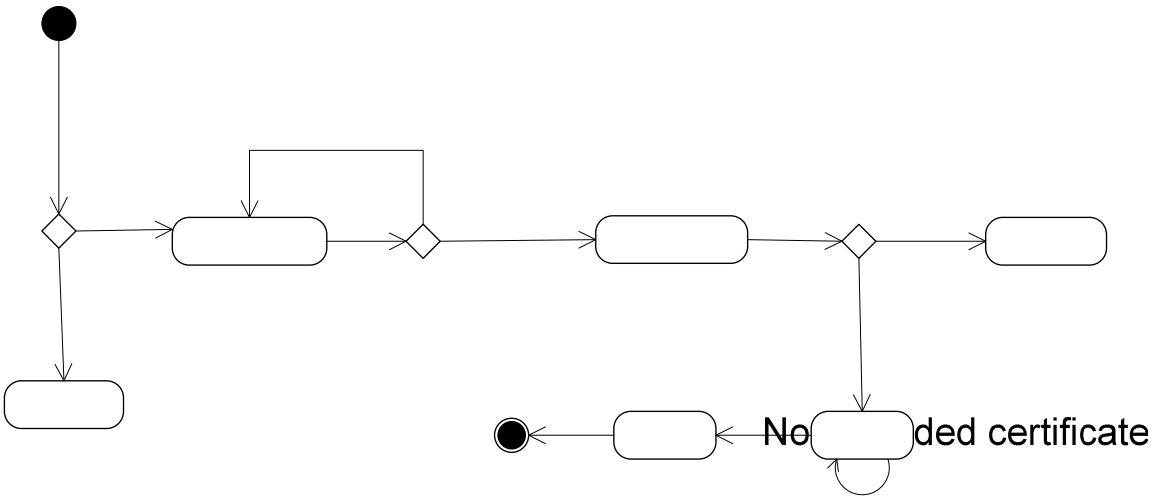
serial_num	username	passwd	has_voted	name	surname	father's name	birth_date	id_number
470e264c	markouk	240184ab	0	maria	koukovini	nikolaos	24.01.1984	S276351
470e290e	evgpapagian	100583cd	0	eugenia	papagiannakopoulou	iwannis	10.05.1983	A857655
470e29e2	gelioud	211107ef	0	georgios	lioudakis	vasilios	21.03.1987	S432752
470e2ab3	tastas	241207gh	0	anastasios	tasoulas	xaralampos	14.02.1986	T562347
4735fe20	alkiskouk	170386ij	0	alkis	koukovinis	nikolaos	17.03.1986	T265749
4735fe8e	alexispapa	123456kl	0	alexios	papagiannopoulos	iwannis	16.12.1985	S624574
4735ff07	irskot	270479mn	0	iraklis	skoteinos	petros	29.04.1979	Y541263
4735ffa2	ionfouk	657321op	0	iwannis	foukarakis	emmanouil	30.06.1980	T476348
4735fffe	ananosssss	234567qr	0	anastasios	nanos	pelopidas	02.01.1983	P234572
4736005a	agiok	765432st	0	aggelos	giokas	ksenofon	12.12.1979	E827348

Σχήμα 7.1 voter_table περιφερειακών βάσεων πριν την έναρξη της διαδικασίας

serial_num	has_voted
470e264c	0
470e290e	0
470e29e2	0
470e2ab3	0
4735fe20	0
4735fe8e	0
4735ff07	0
4735ffa2	0
4735fffe	0
4736005a	0

Σχήμα 7.2 voter_table_kedriko της κεντρικής βάσης πριν την έναρξη της διαδικασίας

Στη συνέχεια παρουσιάζουμε τα πιθανά σενάρια των περιπτώσεων χρήσης που αναλύθηκαν στο προηγούμενο κεφάλαιο, τα οποία προκύπτουν με βάση το παρακάτω διάγραμμα:

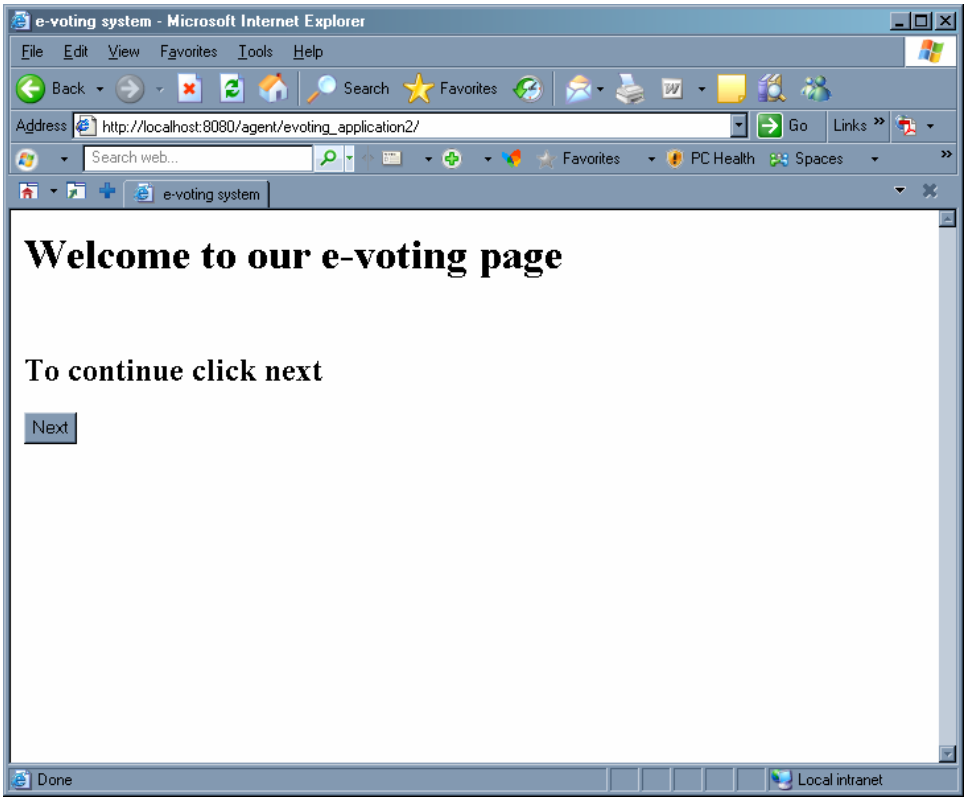


Σχήμα 7.3 Διάγραμμα καταστάσεων (state diagram).

7.1 Πρώτη περίπτωση χρήσης: Ψηφοφορία

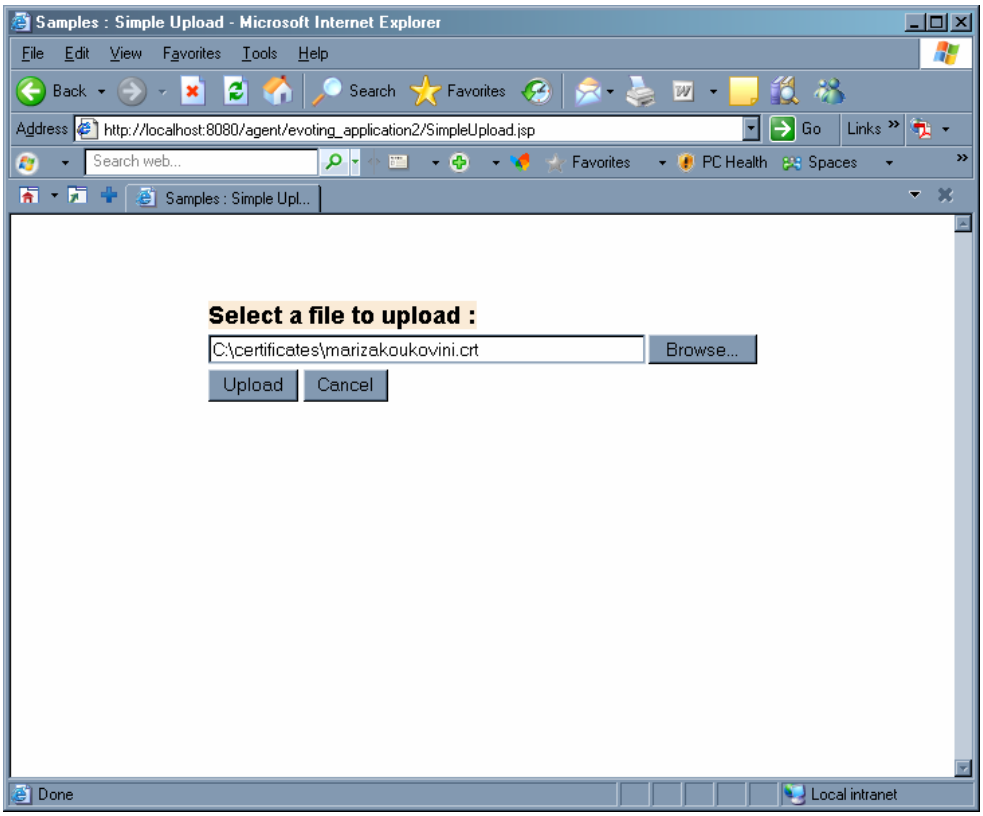
7.1.1 Σενάριο Α: επιτυχής ψηφοφορία

Ο ψηφοφόρος ζητά από τον τοπικό εξυπηρετητή την αρχική σελίδα, η οποία εμφανίζεται εφόσον βρισκόμαστε εντός του έγκυρου χρονικού διαστήματος ψηφοφορίας.



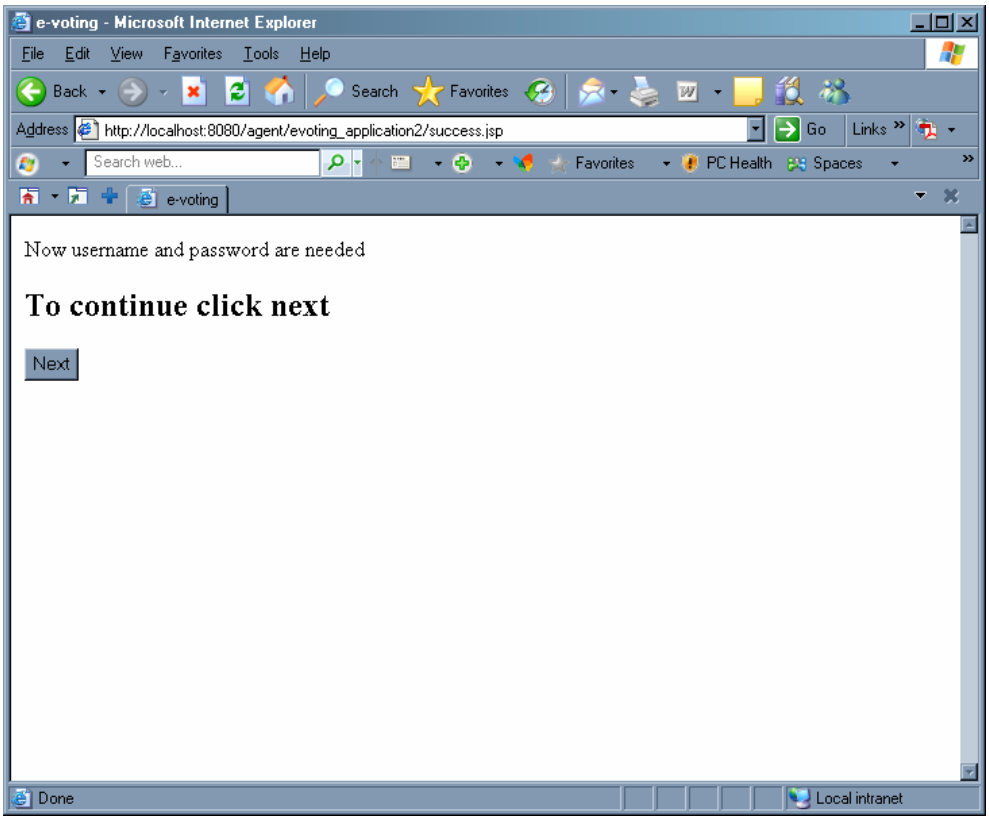
Σχήμα 7.4 index.jsp

Αμέσως μετά του ζητείται η εισαγωγή του πιστοποιητικού (μέσω της smart card).



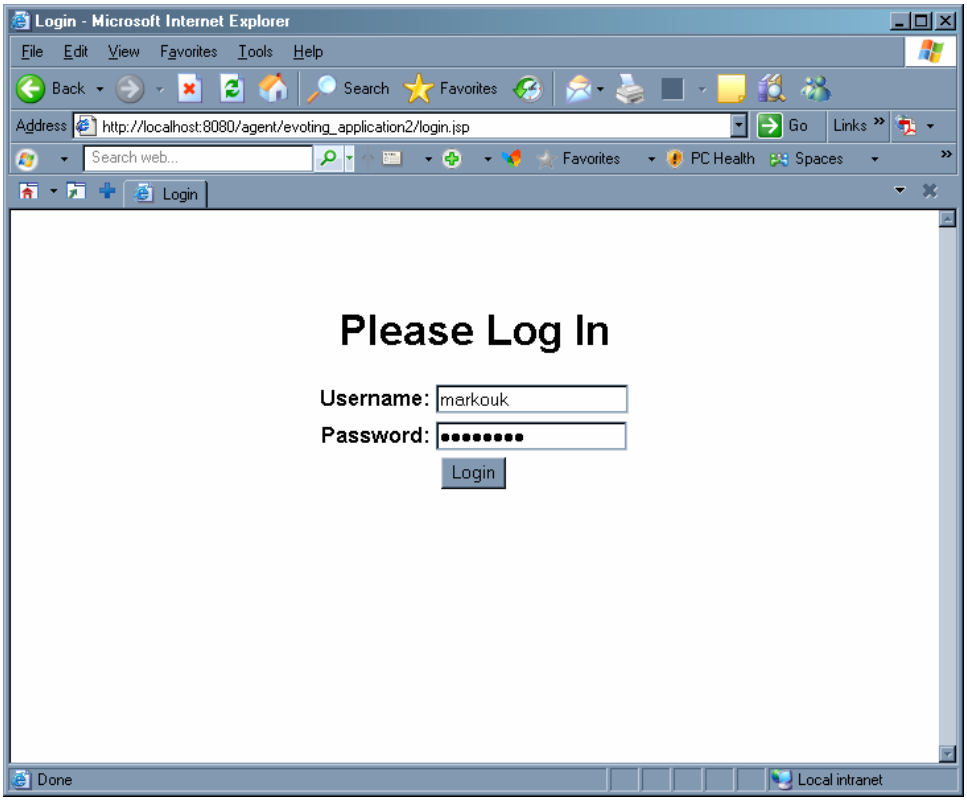
Σχήμα 7.5 SimpleUpload.jsp

Εφόσον γίνει σωστά η εισαγωγή του πιστοποιητικού, εμφανίζεται η επόμενη σελίδα.



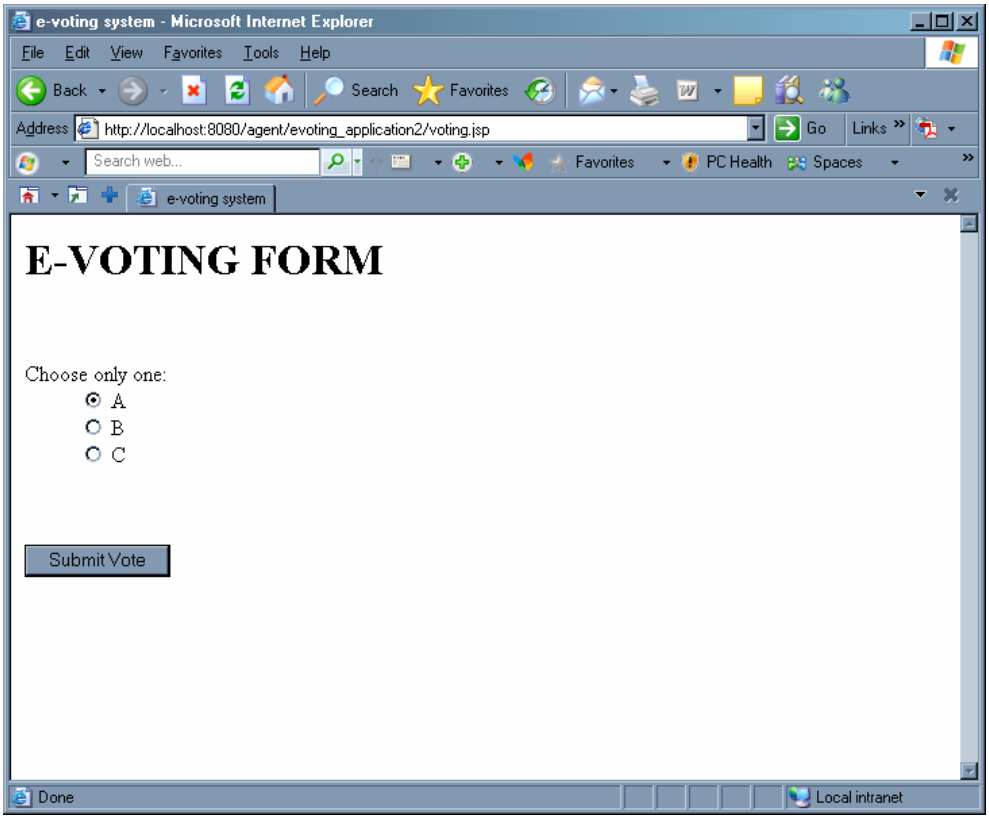
Σχήμα 7.6 success.jsp

Στη συνέχεια ζητείται από το χρήστη να εισάγει username και password.



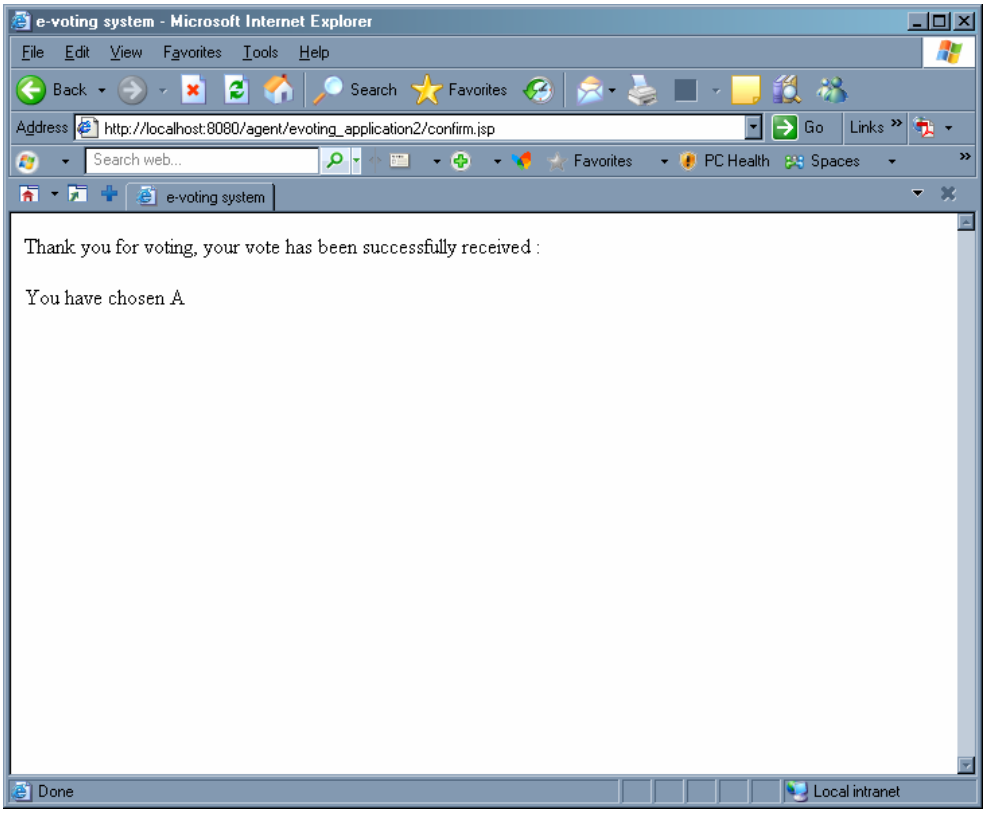
Σχήμα 7.7 login.jsp

Υπό την προϋπόθεση ότι ο ψηφοφόρος δεν έχει ξαναψηφίσει και ταυτόχρονα τα username και password που εισάγει αντιστοιχούν στον σειριακό αριθμό του πιστοποιητικού που έχει εισάγει, σύμφωνα με τις εγγραφές στην τοπική βάση δεδομένων, εμφανίζεται η επόμενη σελίδα, όπου καλείται να ψηφίσει.



Σχήμα 7.8 voting.jsp

Μετά την επιλογή του επιθυμητού υποψηφίου ξεκινάει η αποστολή της ψήφου και αν αυτή ολοκληρωθεί με επιτυχία, εμφανίζεται μια οθόνη επιβεβαίωσης, ότι η διαδικασία έχει διεξαχθεί ομαλά.



Σχήμα 7.9 confirm.jsp

Αν υποθέσουμε ότι μέσα στα τελευταία 50 δευτερόλεπτα έχουν ψηφίσει 2 άτομα συνολικά, σε 2 διαφορετικά εκλογικά κέντρα, οι πίνακες των 2 τοπικών βάσεων δεδομένων αμέσως μετά την καταχώρηση της ψήφου θα έχουν ως εξής:

serial_num	username	passwd	has_voted	name	surname	father's name	birth_date	id_number
470e264c	markouk	240184ab	1	maria	koukovini	nikolaos	24.01.1984	S276351
470e290e	evgpapagian	100583cd	0	eugenia	papagiannakopoulou	iwannis	10.05.1983	A857655
470e29e2	gelioud	211107ef	0	georgios	lioudakis	vasilios	21.03.1987	S432752
470e2ab3	tastas	241207gh	0	anastasios	tasoulas	xaralampos	14.02.1986	T562347
4735fe20	alkiskouk	170386ij	0	alkis	koukovinis	nikolaos	17.03.1986	T265749
4735fe8e	alexispapa	123456kl	0	alexios	papagiannopoulos	iwannis	16.12.1985	S624574
4735ff07	irskot	270479mn	0	iraklis	skoteinos	petros	29.04.1979	Y541263
4735ffa2	ionfouk	657321op	0	iwannis	foukarakis	emmanouil	30.06.1980	T476348
4735fffe	ananosssss	234567qr	0	anastasios	nanos	pelopidas	02.01.1983	P234572
4736005a	agiok	765432st	0	aggelos	giokas	ksenofon	12.12.1979	E827348

Σχήμα 7.10 voter_table στο πρώτο εκλογικό κέντρο

serial_num	username	passwd	has_voted	name	surname	father's name	birth_date	id_number
470e264c	markouk	240184ab	0	maria	koukovini	nikolaos	24.01.1984	S276351
470e290e	evgpapagian	100583cd	1	eugenia	papagiannakopoulou	iwannis	10.05.1983	A857655
470e29e2	gelioud	211107ef	0	georgios	lioudakis	vasilios	21.03.1987	S432752
470e2ab3	tastas	241207gh	0	anastasios	tasoulas	xaralampos	14.02.1986	T562347
4735fe20	alkiskouk	170386ij	0	alkis	koukovinis	nikolaos	17.03.1986	T265749
4735fe8e	alexispapa	123456kl	0	alexios	papagiannopoulos	iwannis	16.12.1985	S624574
4735ff07	irskot	270479mn	0	iraklis	skoteinos	petros	29.04.1979	Y541263
4735ffa2	ionfouk	657321op	0	iwannis	foukarakis	emmanouil	30.06.1980	T476348
4735fffe	ananosssss	234567qr	0	anastasios	nanos	pelopidas	02.01.1983	P234572
4736005a	agiok	765432st	0	aggelos	giokas	ksenofon	12.12.1979	E827348

Σχήμα 7.11 voter_table στο δεύτερο εκλογικό κέντρο.

Μετά τα 50 δευτερόλεπτα ο ενδιάμεσος πίνακας που έχει δημιουργηθεί στην κεντρική βάση δεδομένων έχει ως εξής:

serial_num	has_voted
470e264c	1
470e290e	1

Σχήμα 7.12 voter_table_endiameso της κεντρικής βάσης δεδομένων

Μετά από 10 δευτερόλεπτα έχει ενημερωθεί και ο κεντρικός πίνακας, όπως επίσης και οι πίνακες των περιφερειακών συστημάτων, ενώ ο ενδιάμεσος πίνακας έχει διαγραφεί.

serial_num	has_voted
470e264c	1
470e290e	1
470e29e2	0
470e2ab3	0
4735fe20	0
4735fe8e	0
4735ff07	0
4735ffa2	0
4735fffe	0
4736005a	0

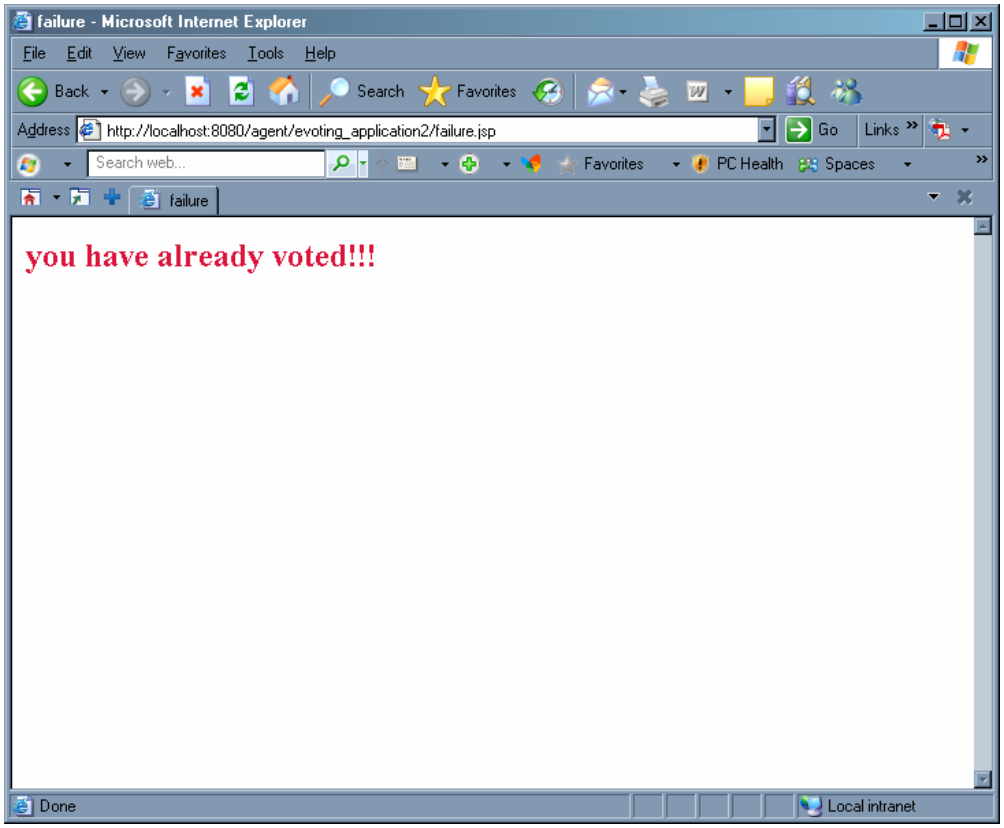
Σχήμα 7.13 voter_table_kedriko της κεντρικής βάσης δεδομένων

serial_num	username	passwd	has_voted	name	surname	father's name	birth_date	id_number
470e264c	markouk	240184ab	1	maria	koukovini	nikolaos	24.01.1984	S276351
470e290e	evgpapagian	100583cd	1	eugenia	papagiannakopoulou	iwannis	10.05.1983	A857655
470e29e2	gelioud	211107ef	0	georgios	lioudakis	vasilios	21.03.1987	S432752
470e2ab3	tastas	241207gh	0	anastasios	tasoulas	xaralampos	14.02.1986	T562347
4735fe20	alkiskouk	170386ij	0	alkis	koukovinis	nikolaos	17.03.1986	T265749
4735fe8e	alexispapa	123456kl	0	alexios	papagiannopoulos	iwannis	16.12.1985	S624574
4735ff07	irskot	270479mn	0	iraklis	skoteinos	petros	29.04.1979	Y541263
4735ffa2	ionfouk	657321op	0	iwannis	foukarakis	emmanouil	30.06.1980	T476348
4735fffe	ananosssss	234567qr	0	anastasios	nanos	pelopidas	02.01.1983	P234572
4736005a	agiok	765432st	0	aggelos	giokas	ksenofon	12.12.1979	E827348

Σχήμα 7.14 voter_table των περιφερειακών βάσεων δεδομένων.

7.1.2 Σενάριο Β: Ο ψηφοφόρος έχει ήδη ψηφίσει

Μετά την εισαγωγή του πιστοποιητικού, ο έλεγχος στη βάση δείχνει ότι ο κάτοχος του πιστοποιητικού με το συγκεκριμένο σειριακό αριθμό έχει ήδη ψηφίσει (has_voted=1), οπότε στον ψηφοφόρο εμφανίζεται η σελίδα που βλέπουμε παρακάτω και η διαδικασία σταματάει εδώ.

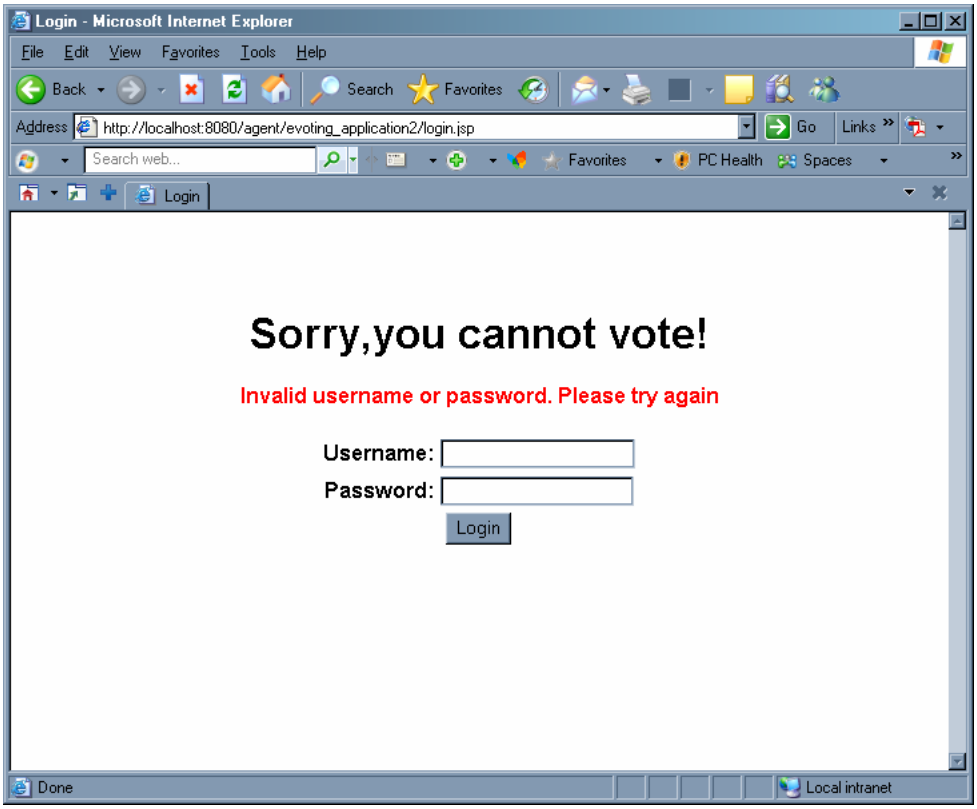


Σχήμα 7.15 failure.jsp

7.1.3 Σενάριο Γ: λάθος διαπιστευτήρια

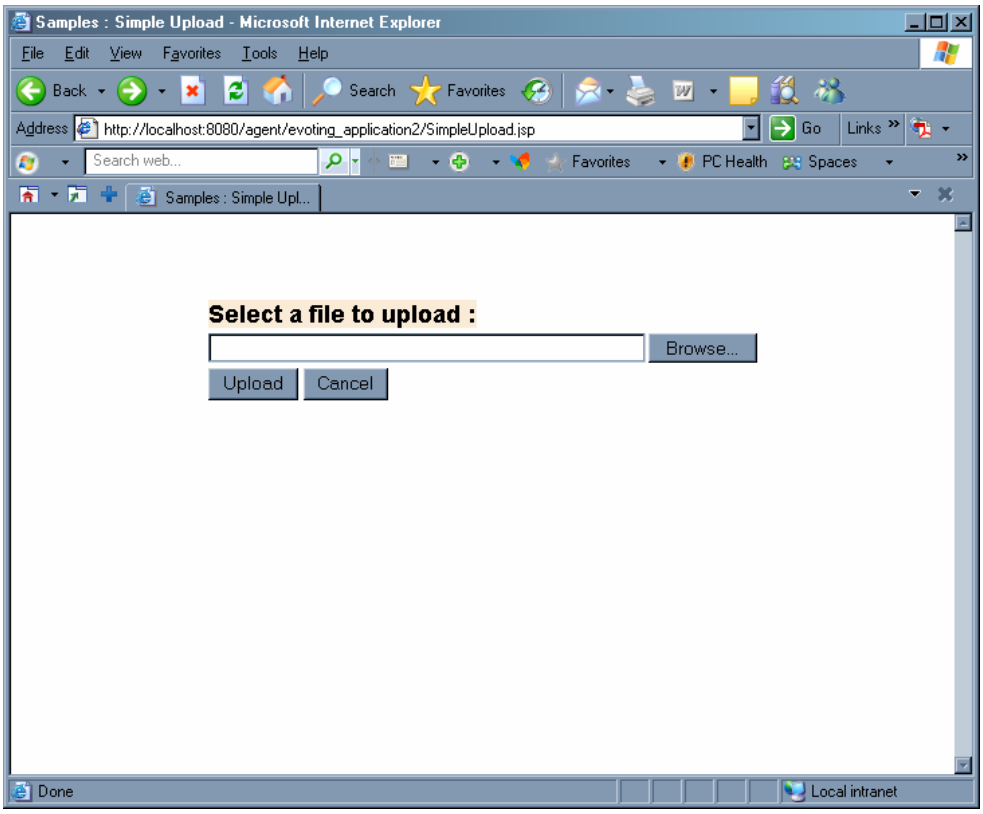
Αφού ο ψηφοφόρος εισάγει το πιστοποιητικό του και ο απαραίτητος έλεγχος στη βάση δείξει ότι δεν έχει ξαναψηφίσει, ζητούνται από το χρήστη τα username και password. Αν το username ή/και το password δεν είναι αυτά που αντιστοιχούν στο σειριακό αριθμό του

πιστοποιητικού που έχει εισαχθεί, εμφανίζεται η παρακάτω οθόνη, μέχρι ο ψηφοφόρος να εισάγει σωστά τα στοιχεία του:

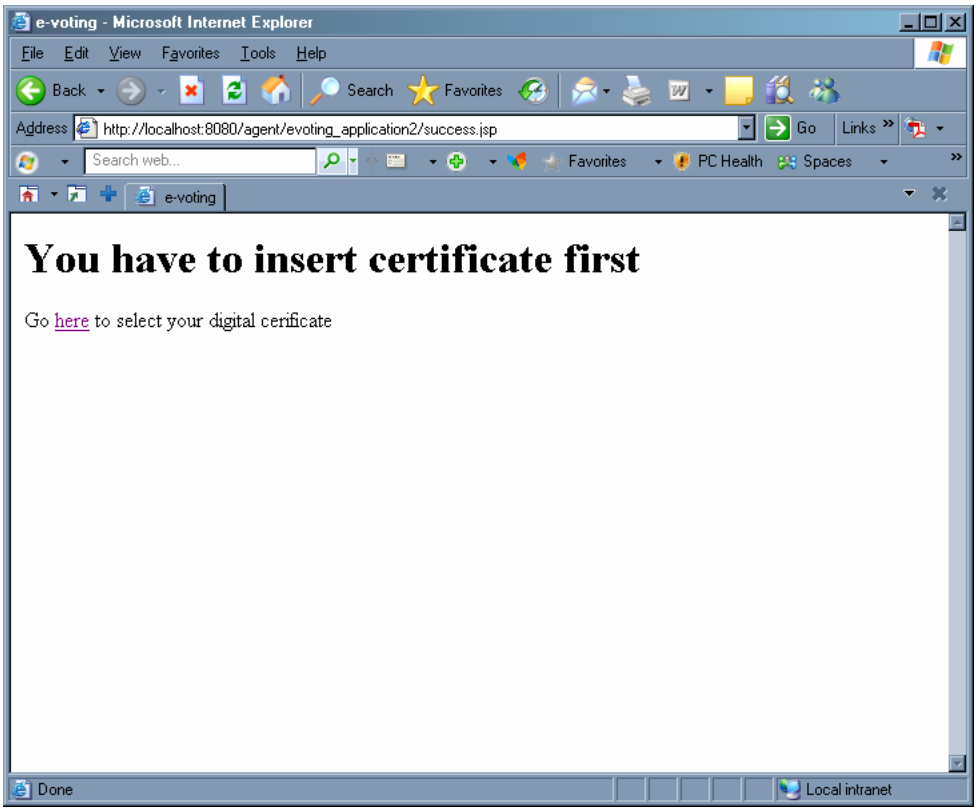


Σχήμα 7.16 login.jsp

7.1.4 Σενάριο Δ: Δεν έχει εισαχθεί πιστοποιητικό



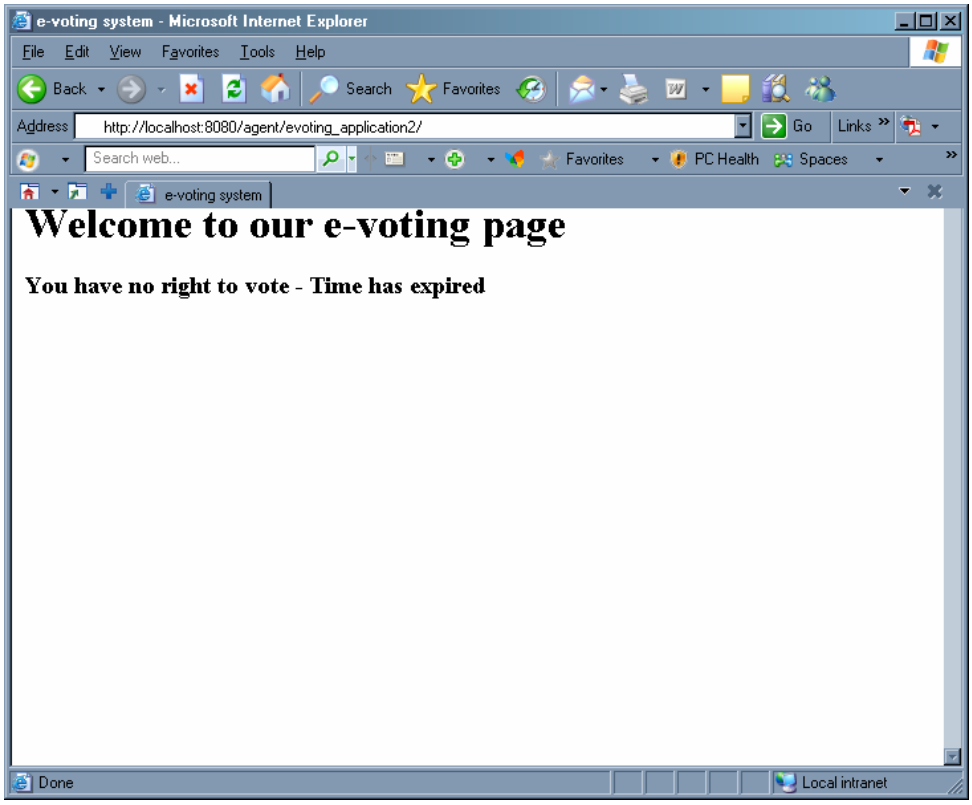
Στην περίπτωση που ο χρήστης πατήσει το κουμπί upload, χωρίς προηγουμένως να εισάγει την smart card, εμφανίζεται η παρακάτω οθόνη, με το μήνυμα ότι δεν μπορεί να συνεχιστεί η διαδικασία εάν πρώτα δεν εισαχθεί πιστοποιητικό.



Σχήμα 7.17 success.jsp σε περίπτωση μη εισαγωγής πιστοποιητικού

7.1.5 Σενάριο Ε: Μη έγκυρη στιγμή ψηφοφορίας

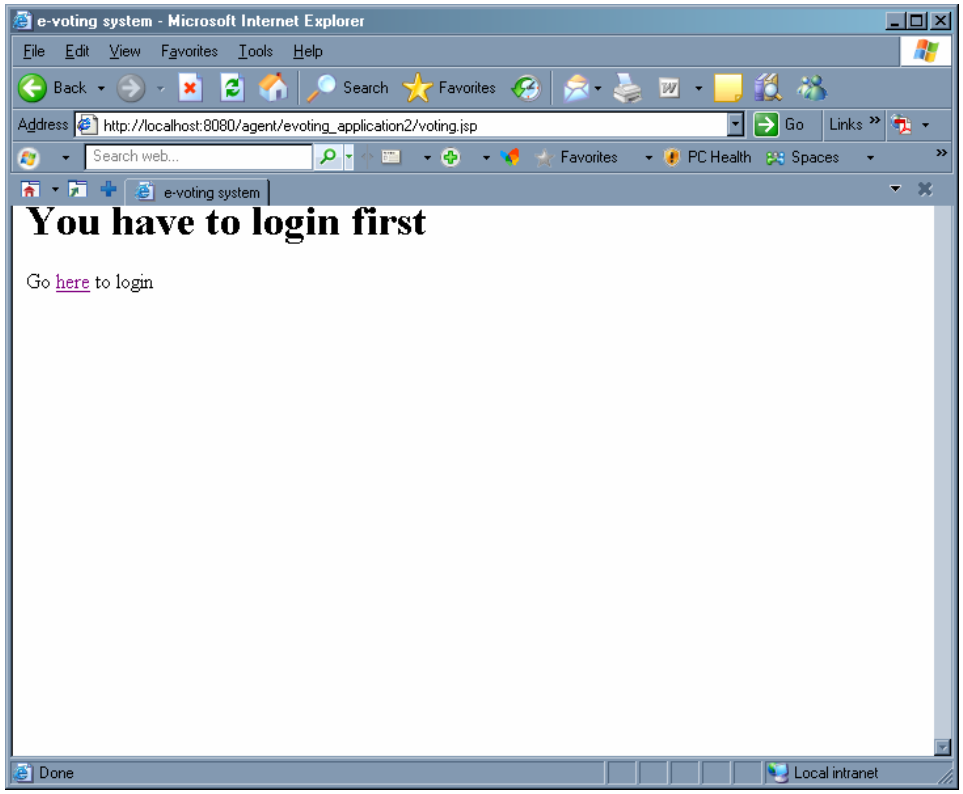
Ο ψηφοφόρος προσέρχεται στο εκλογικό κέντρο είτε νωρίτερα είτε αργότερα από το έγκυρο χρονικό διάστημα ψηφοφορίας. Όταν ζητά την αρχική σελίδα, βλέπει μόνο την παρακάτω οθόνη:

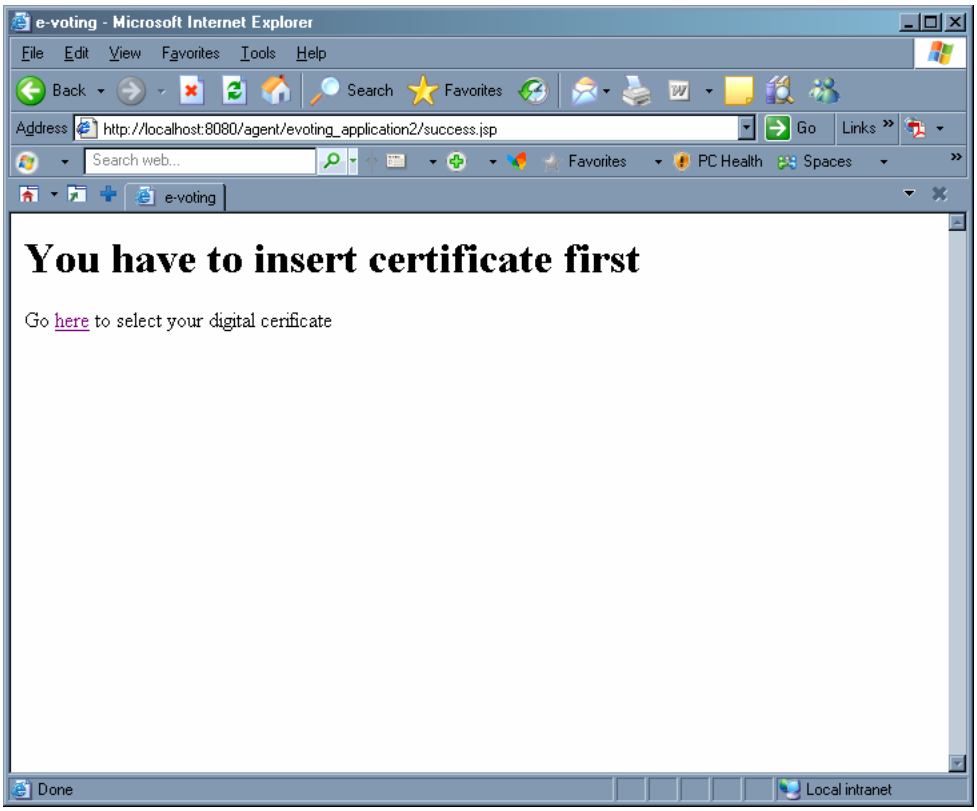


Σχήμα 7.18 index.jsp σε περίπτωση μη έγκυρης χρονικής στιγμής ψηφοφορίας.

7.1.6 Σενάριο ΣΤ: Ψηφοφορία χωρίς έλεγχο εξουσιοδότησης

Αν κάποιος επιχειρήσει να «μπει» κατευθείαν στη σελίδα υποβολής ψήφου (voting.jsp) , χωρίς προηγουμένως να έχει εισάγει κανένα από τα απαραίτητα διαπιστευτήρια, εμφανίζονται με τη σειρά οι επόμενες οθόνες:



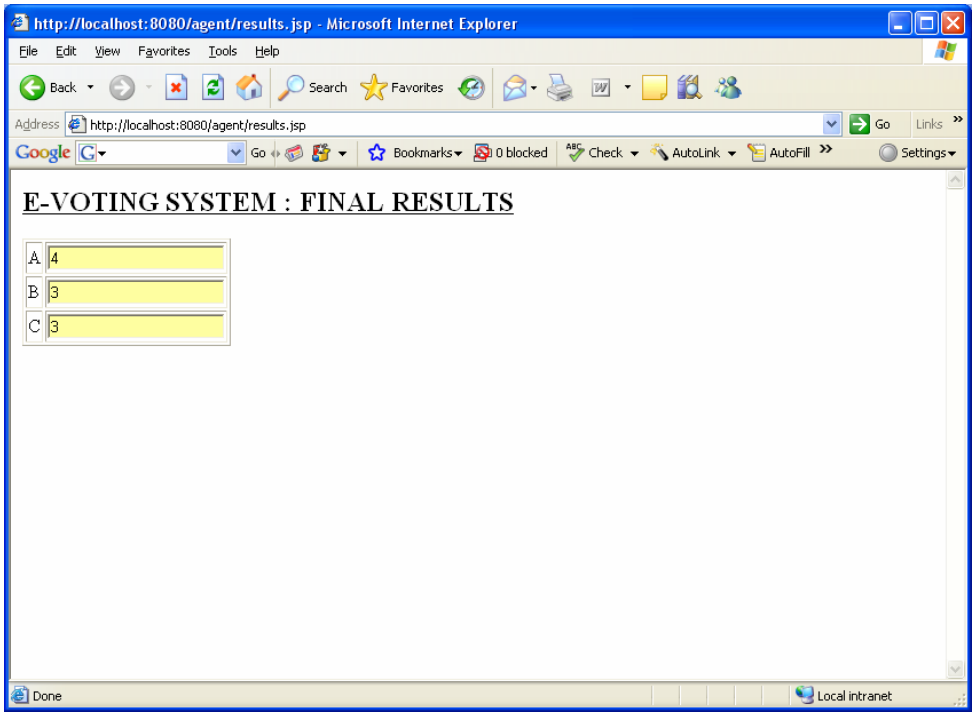


Σχήμα 7.19 voting.jsp και success.jsp σε περίπτωση μη εισαγωγής διαπιστευτηρίων.

Αν τώρα εισάγει σωστά όλα τα ζητούμενα, η διαδικασία συνεχίζεται κανονικά.

7.2 Δεύτερη περίπτωση χρήσης: Καταμέτρηση και εμφάνιση αποτελεσμάτων

Αφού έχει ολοκληρωθεί η διαδικασία της ψηφοφορίας, εμφανίζεται μια οθόνη αποτελεσμάτων, σε οποιονδήποτε ζητήσει την αντίστοιχη σελίδα, όπως παρακάτω:



8 Συμπεράσματα

Η παρούσα διπλωματική εργασία επιχείρησε να επιλύσει αρκετά από τα προβλήματα που προκύπτουν κατά τη χρήση συστημάτων ηλεκτρονικής ψηφοφορίας. Αρχικά θα πρέπει να αναφερθεί ότι πρόκειται για ένα κατανεμημένο, υβριδικό σύστημα Polling Place Internet voting, δηλαδή μεταξύ Polling Place Voting, εφόσον οι ψήφοι αποστέλλονται από συγκεκριμένα εκλογικά κέντρα, και Internet Voting, εφόσον έχουμε μεταφορά ψήφων και λοιπών πληροφοριών μέσω του διαδικτύου. Αφενός μειώνονται έτσι σημαντικά οι πιθανότητες εμφάνισης φαινομένων, όπως ο πολιτικός εξαναγκασμός, η άρση του ιδιωτικού απορρήτου και της μυστικότητας της ψηφοφορίας, οι οποίες είναι μεγαλύτερες στην περίπτωση του Remote Internet Voting, που λαμβάνει χώρα σε μη ελεγχόμενο περιβάλλον. Επιπλέον δεν υπάρχει η απαίτηση κατοχής του εξειδικευμένου λογισμικού από ιδιώτες, γεγονός που δημιουργεί καλύτερες προϋποθέσεις για μεγαλύτερη συμμετοχή. Αφετέρου γίνεται χρήση των τεχνολογιών διαδικτύου και επιπρόσθετων μηχανισμών ασφάλειας.

Σε ό,τι αφορά την παρέμβαση του ανθρώπινου παράγοντα, είναι αντιληπτό ότι δεν μπορεί να προβλεφθεί ή ελεγχθεί πλήρως, είτε πρόκειται για δολιοφθορά είτε για ακούσιο λάθος ή παράλειψη. Για το λόγο αυτό γίνεται η παραδοχή ότι δεν υπάρχει κανενός είδους πρόσβαση στο υλικό και το λογισμικό του συστήματος, όλα λειτουργούν όπως ακριβώς έχουν σχεδιαστεί και οι εκάστοτε διαχειριστές είναι αντικειμενικοί και αμερόληπτοι. Στην πράξη, αυτό επιτυγχάνεται με τον περιορισμό της πρόσβασης σε κρίσιμες παραμέτρους του συστήματος (κρυπτογραφημένες βάσεις δεδομένων, trusted computing).

Προκειμένου να αποφευχθεί η αντιστοίχιση των ψήφων με τους ψηφοφόρους και κατ' επέκταση η παραβίαση της μυστικότητας, η ψήφος και ο αναγνωριστικός αριθμός κάθε ψηφοφόρου μεταφέρονται μέσα από διαφορετικά κανάλια και σε ασυσχέτιστες μεταξύ τους χρονικές στιγμές. Ακόμα και ο πίνακας που αποθηκεύει προσωρινά και μόνο τοπικά τους σειριακούς αριθμούς των πιστοποιητικών των ψηφοφόρων διαγράφεται ανά τακτά χρονικά διαστήματα.

Επίσης αντιμετωπίστηκε επαρκώς η περίπτωση νοθείας, μέσω της δημιουργίας και συνεχούς ενημέρωσης των βάσεων δεδομένων, τόσο σε περιφερειακό όσο και σε κεντρικό επίπεδο. Μέσω αυτού του μηχανισμού, είναι δυνατόν να αποτραπεί επιπλέον και η κατάρρευση του συστήματος λόγω φόρτου, εφόσον η συχνότητα ενημέρωσης των βάσεων μπορεί να ρυθμιστεί, ώστε να είναι επαρκώς μεγάλη, οπότε κάθε φορά θα μεταφέρεται όχι απαγορευτικός όγκος δεδομένων.

Ως μηχανισμός διανομής πιστοποιητικών και πιστοποίησης επιλέχθηκαν οι έξυπνες κάρτες, οι οποίες θεωρούνται ανθεκτικές στην παραβίαση. Η κρυπτογράφηση της ψήφου εξασφαλίζει τη μυστικότητά της κατά τη μεταφορά, ενώ η ψηφιακή υπογραφή της πιστοποιεί την αυθεντικότητα τόσο της ίδιας όσο και του πράκτορα.

Σε ό,τι αφορά τη μεταφορά της ψήφου με τη βοήθεια κινητών πρακτόρων, εναλλακτικές υλοποιήσεις θα ήταν είτε να «φορτώνονται» περισσότερες από μία ψήφοι στον πράκτορα είτε να υπάρχουν περισσότεροι από έναν πράκτορες σε κάθε αντιπροσωπεία.

Παράρτημα

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@page import="java.text.*"%>
<%@page import="java.text.DateFormat"%>
<%@page import="java.text.SimpleDateFormat"%>
<%@page import="java.util.*"%>
<%@page import="java.util.Calendar.*"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>e-voting system</title>
</head>
<body>
<h1>Welcome to our e-voting page</h1>
<%
      TimeZone tz = TimeZone.getTimeZone("Europe/Athens");
      GregorianCalendar cal1 = new GregorianCalendar(2007,
                  Calendar.NOVEMBER, 30, 20, 00);
      GregorianCalendar cal2 = new GregorianCalendar(2007,
                  Calendar.NOVEMBER, 30, 8, 00);
      Date d1 = cal1.getTime();
      Date d2 = cal2.getTime();
      long l1 = d1.getTime();
      long l2 = d2.getTime();
      GregorianCalendar cal = new GregorianCalendar(tz);
      Date d = cal.getTime();
      long l = d.getTime();
      //System.out.println(l);
      //System.out.println(l1);
      //System.out.println(l2);
      if ((l < l1) && (l > l2)) {
%>
<form action="SimpleUpload.jsp" method="post"><br>


<BR>
<input type='hidden' name='title' value='e-voting'> <input
type='hidden' name='nextpage' value='success.jsp'>

<h2>To continue click next</h2>
<TR>
      <TH><INPUT TYPE="SUBMIT" VALUE="Next"> <BR>
</form>
<%
} else {
%>
<h3>You have no right to vote - Time has expired</h3>

<%
}
```

```
%>
</body>
</html>
```

Κώδικας 1. index.jsp

```
<%@ page import="java.io.*"%>
<%@ page import="java.security.*"%>
<%@ page import="java.security.cert.*"%>
<%@ page import="java.math.*"%>
<%@ page import="pack.appendToFile"%>
<%@ page language="java" import="javazoom.upload.*,java.util.*"%>
<%@ page errorPage="ExceptionHandler.jsp"%>

<jsp:useBean id="upBean" scope="page"
class="javazoom.upload.UploadBean">
      <jsp:setProperty name="upBean" property="folderstore"
value="c:/uploads" />
</jsp:useBean>

<head>
<title>Samples : Simple Upload</title>
<style TYPE="text/css">
<!--
.style1 {
      font-size: 12px;
      font-family: Verdana;
}
-->
</style>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
</head>
<body bgcolor="#FFFFFF" text="#000000">
<ul class="style1">

      <%
            String nextPage = null;
            if
(MultipartFormDataRequest.isMultipartFormData(request)) {
                  // Uses MultipartFormDataRequest to parse the HTTP
request.
                  MultipartFormDataRequest mrequest = new
MultipartFormDataRequest(
                  request);
                  String todo = null;
                  if (mrequest != null)
                        todo = mrequest.getParameter("todo");
                  if ((todo != null) &&
(todo.equalsIgnoreCase("upload"))) {
                        Hashtable files = null;
                        files = mrequest.getFiles();

                        if ((files != null) && (!files.isEmpty())) {
                  UploadFile file = (UploadFile)
files.get("uploadfile");
                  if (file != null)
                        out.println("<li>Form field : uploadfile"
                        + "<BR> Uploaded file : "
                        + file.getFileName() + " ("
                        + file.getFileSize() + " bytes)"
```

```
                        + "<BR> Content Type : "
                        + file.getContentType());

                  upBean.store(mrequest, "uploadfile");
                  session.setAttribute("uploadfile", file.getData());
                  try {
                        FileInputStream fis = new FileInputStream(
                        "C:\\certificates\\" + file.getFileName());

                        CertificateFactory cf = CertificateFactory
                        .getInstance("X.509");

                        java.security.cert.X509Certificate cert =
(X509Certificate) cf
                        .generateCertificate(fis);
                        fis.close();

                        BigInteger c = cert.getSerialNumber();
                        int f = c.intValue();

                        String hexstr = Integer.toHexString(f);
                        System.out.println("SerialNumber = " +
hexstr);

      session.setAttribute("CertificateSerialNumber",
                        hexstr);
                        // Create file if it does not exist
                        File file1 = new File("C:\\voters.txt");
                        boolean success = file1.createNewFile();
                        if (success) {
                              System.out.println("dimiourgithike");
                        } else {
                              System.out.println("uparxei idi");
                        }
                        appendToFile.append(hexstr);
                        System.out.println(hexstr);

                  } catch (java.io.FileNotFoundException e) {

                  }

                  nextPage = "success.jsp";
                  response.sendRedirect(nextPage);
                        } else {

                  out.println("<li>No uploaded files");
                        }

                  } else
                        out.println("<BR> todo=" + todo);
            }
      %>
</ul>
<form method="post" action="SimpleUpload.jsp" name="upform"
enctype="multipart/form-data">
<table width="60%" border="0" cellspacing="1" cellpadding="1"
align="center" class="style1">
      <tr>
            <td align="left"><b>Select a file to upload :</b></td>
      </tr>
      <tr>
```

```
            <td align="left"><input type="file" name="uploadfile"
size="50"></td>
      </tr>
      <tr>
            <td align="left"><input type="hidden" name="todo"
value="upload"> <input type="submit" name="Submit"
                  value="Upload"> <input type="reset" name="Reset"
value="Cancel"</td>
      </tr>
</table>

<br>
<br>

<p> </p>
<p> </p>
</form>
</body>
```

Κώδικας 2. SimpleUpload.jsp

```
package pack;

import java.io.*;

public class appendToFile {

      public static void append(String sn) {
            try {
                  String filename = "C:\\voters.txt";
                  boolean append = true;
                  FileWriter fw = new FileWriter(filename, append);

                  fw.write(sn + "\n");//appends the string to the
file
                  fw.close();
            }

            catch (IOException ioe) {
                  System.err.println("IOException: " +
ioe.getMessage());
            }

      }
}
```

Κώδικας 3. appendToFile.java

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>e-voting</title>
</head>
<body>
<%
                  String b1 = (String) session
                  .getAttribute("CertificateSerialNumber");
      System.out.println(b1);
      if (session.getAttribute("uploadfile") != null) {
%>

<form action="login.jsp" method="post">Now username and password are
needed

<h2>To continue click next</h2>
<TR>
      <TH><INPUT TYPE="SUBMIT" VALUE="Next"> <input type='hidden'
name='nextpage' value='voting.jsp'> <BR>
      <%
      } else {
      %>
      <h1>You have to insert certificate first</h1>

      Go <a href="SimpleUpload.jsp">here</a> to select your digital
cerificate <%
      }
      %>

</body>
</html>
```

Κώδικας 4. success.jsp

```
<%@page language='java' contentType='text/html'%>
<%@ page import="java.io.*"%>
<%@ page import="java.security.*"%>
```

```
<%@ page import="java.security.cert.*"%>
<%@ page import="java.math.*"%>
<%@ page import="java.sql.*"%>
<%@ page import="java.util.*"%>



<%!// Begin declaration block

      // This method checks in local database whether the user has
already voted or not

      public static boolean hasalreadyvoted(String b) throws
SQLException {

            String username_db = "root";
            String password_db = "240184";
            String url = "jdbc:mysql://localhost:3306/evoting";

            //Fortwsi toy Driver
            try {

      Class.forName("com.mysql.jdbc.Driver").newInstance();
            } catch (ClassNotFoundException cnfe) {
                  System.err.println("Den fortwthike swsta o JDBC
driver!");
                  cnfe.printStackTrace();
                  System.exit(1);
            } catch (InstantiationException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (IllegalAccessException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            //    syndesi sti database
            Connection connection = null;
            try {
                  connection = DriverManager.getConnection(url,
username_db,
                              password_db);
            } catch (SQLException se) {
                  System.out.println("Den kataferame na syndethoume
me to server.");
                  se.printStackTrace();
                  System.exit(1);
            }//Dimiourgia Statement
            Statement st = connection.createStatement();
            ResultSet results = null;
            try {
                  results = st.executeQuery("select has_voted " +
"from voter_table "
                              + "where serial_num =  '" + b + "' ");

            } catch (SQLException se1) {
                  System.out.println("Lathos sto SQL erwtima.");
                  se1.printStackTrace();
                  System.exit(1);
            }
```

```
            boolean d = false;
            while (results.next()) {
                  int hv = results.getInt("has_voted");
                  if (hv == 1) {
                        d = true;
                  }
            }

            st.close();
            connection.close();

            return d;

      }

      //This method does the user authentication.

      public static boolean verify1(String username, String password,
String b)
                  throws SQLException {

            String username_db = "root";
            String password_db = "240184";
            String url = "jdbc:mysql://localhost:3306/evoting";

            //Fortwsi toy Driver
            try {

      Class.forName("com.mysql.jdbc.Driver").newInstance();
            } catch (ClassNotFoundException cnfe) {
                  System.err.println("Den fortwthike swsta o JDBC
driver!");
                  cnfe.printStackTrace();
                  System.exit(1);
            } catch (InstantiationException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (IllegalAccessException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            //syndesi sti database
            Connection connection = null;
            try {
                  connection = DriverManager.getConnection(url,
username_db,
                              password_db);
            } catch (SQLException se) {
                  System.out.println("Den kataferame na syndethoume
me to server.");
                  se.printStackTrace();
                  System.exit(1);
            }//Dimiourgia Statement
            Statement st = connection.createStatement();

            ResultSet results = null;
            try {
                  results = st.executeQuery("select
username,passwd,has_voted "
```

```
                              + "from voter_table " + "where
serial_num =  '" + b + "' ");

            } catch (SQLException se1) {
                  System.out.println("Lathos sto SQL erwtima.");
                  se1.printStackTrace();
                  System.exit(1);
            }

            boolean d = false;
            while (results.next()) {

                  String pass = results.getString("passwd");
                  String usr = results.getString("username");
                  int hv = results.getInt("has_voted");
                  if ((username != null) && (password != null)
                              && (username.equals(usr)) &&
(password.equals(pass))
                              && (hv == 0)) {
                        d = true;
                        System.out.println(usr);
                        Statement st2 = connection.createStatement();
                        st2
                                    .executeUpdate("update
voter_table set has_voted=1  where serial_num ='"
                                                + b + "' ");
                        st2.close();
                  }
            }

            st.close();
            connection.close();

            return d;

      }%>
<%-- End declaration block --%>


<%
      // Begin scriptlet
      // This request parameter is required.  It specifies what
should be
      // displayed if the login attempt is successful

      String nextPage = request.getParameter("nextpage");

      // This request parameter specifies a title for the login page
      String title = request.getParameter("title");
      if (title == null)
            title = "Please Log In"; // If not specified, use a
default

      // Look for username and password parameters in the request
      String username1 = request.getParameter("username");
      String password1 = request.getParameter("password");
      String b1 = (String) session
                  .getAttribute("CertificateSerialNumber");
      System.out.println(b1);
      // If the username and password are defined and  valid, then
store
```

```
      // the username and the password in the session, and redirect
to the specified page
      // We do this without displaying any content of our own.
      if ((hasalreadyvoted(b1))) {
            nextPage = "failure.jsp";
            response.sendRedirect(nextPage);

      } else if ((username1 != null) && (password1 != null)
                  && (b1 != null) && verify1(username1, password1,
b1)) {
            session.setAttribute("username", username1);
            session.setAttribute("password", password1);

            response.sendRedirect(nextPage);
      } else {
            // Otherwise, we're going to have to display the login
screen.
            // If the username and password properties are totally
undefined,
            // then this is the first time, and all we display is the
screen.
            // Otherwise, if they are defined, then we've just had a
failed login
            // attempt, so display an additional "please try again"
message.
            String message = "";
            if ((username1 != null) || (password1 != null)) {

                  title = "Sorry,you cannot vote!";
                  message = "Invalid username or password. Please try
again";

            }
%>

<%-- This is the body of the else block started above.  It displays -
-%>
<%-- the login page.--%>
<head>
<title>Login</title>
</head>
<body bgcolor='white'>
<br>
<br>
<br>
<decor:box color='yellow' margin='25' borderWidth='3' title='Login'>
      <div align=center><%-- Center everything inside the box --%>
<%-- Display the login title and optional error message --%>
      <font face='helvetica'>
      <h1><%=title%></h1>
      </font> <font face='helvetica'
color='red'><b><%=message%></b></font> <%-- Now display an HTML form
for the user to enter login information --%>
      <form action='login.jsp' method='post'>
      <table>
            <tr>
                  <%-- First row: username --%>
                  <td align='right'><b><font
face='helvetica'>Username:</font></b></td>
                  <td><input name='username'></td>
            </tr>
```

```
            <tr>
                  <%-- Second row: password --%>
                  <td align='right'><b><font
face='helvetica'>Password:</font></b></td>
                  <td><input type='password' name='password'></td>
            </tr>
            <tr>
                  <%-- Third row: login button --%>
                  <td align='center' colspan=2><font
face='helvetica'><b> <input type=submit value='Login'>
</b></font></td>
            </tr>
      </table>

      <input type='hidden' name='nextpage' value='<%=nextPage%>'>
<input type='hidden' name='title'
            value='<%=title%>'></form>
      </div>
</decor:box>
<%-- End of the custom box tag --%>
</body>
<%-- End of the HTML output --%>
<%
}
%>
```

Κώδικας 5. login.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>failure</title>
</head>
<body>
<span style="color: crimson; font-size: large"><span
      style="color: crimson; font-size: x-large; font-weight:
bold">you have already voted!!!</span> </span>
</body>
</html>
```

Κώδικας 6. failure.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>e-voting system</title>
</head>
<body>
<%
if (session.getAttribute("username") != null) {
%>
<h1>E-VOTING FORM</h1>
<form action="confirm.jsp" method="post"><br>
<br>
Choose only one:
<DD><INPUT TYPE="RADIO" NAME="party" VALUE="A"> A
<DD><INPUT TYPE="RADIO" NAME="party" VALUE="B"> B
<DD><INPUT TYPE="RADIO" NAME="party" VALUE="C"> C
</P>
<br>
<br>
<input type="submit" name="submit" value="Submit Vote">
</form>
<%
} else {
%>
<h1>You have to login first</h1>
Go
<a href="login.jsp">here</a>
to login
<%
}
%>

</body>
</html>
```

Κώδικας 7. voting.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@page import="gr.ntua.ifouk.communication.region.*"%>
<%@page import="gr.ntua.ifouk.communication.*"%>
<%@page import="gr.ntua.ifouk.agency.*"%>
<%@page import="gr.ntua.ifouk.agent.rpc.*"%>
<%@page import="gr.ntua.ifouk.agent.*"%>
<%@page import="gr.ntua.ifouk.agent.Agent"%>
<%@page import="gr.ntua.ifouk.agent.helpers.*"%>
<%@page import="java.net.URL"%>
<%@page import="gr.ntua.ifouk.agent.helpers.*"%>
<%@page import="gr.ntua.ifouk.region.*"%>
<%@page import="java.util.*"%>
<%@page import="java.net.URL"%>
<%@page import="java.net.MalformedURLException"%>
<%@page import="gr.ntua.ifouk.*"%>
<%@page import="pack.MyUtil"%>
```

```
<%@page import="java.util.*"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-
1">
<title>e-voting system</title>
</head>
<body>
<%
      String choice = request.getParameter("party");
      String pass = (String) session.getAttribute("password");
      String user = (String) session.getAttribute("username");
      if (choice != null) {
            Object[] msg = new Object[1];
            msg[0] = choice;
            SynchCall myCall = new SynchCall("receiveVote", msg);

            String region = ("http://localhost:8080");
            URL u = new URL(region + "/soap/servlet/rpcrouter");

            AgentFilter filt = new AgentFilter();
            filt.setConstraint("agent.name", "Crypto2");

            SearchWorker sw = new SearchWorker(u, filt);

            while (!sw.isResultReady())
                  ;
            if (sw.getResult() instanceof String) {
%>
<%=(String) sw.getResult()%>
<%
            }

            if (sw.getResult() instanceof
gr.ntua.ifouk.agent.helpers.Result) {
                  Result res = (Result) sw.getResult();
                  Vector v = res.getEntries();
                  if (v == null) {
%>
<center>No agents found.</center>
<%
                  }
                  Enumeration e = v.elements();
                  while (e.hasMoreElements()) {
            ResultEntry entry = (ResultEntry) e.nextElement();
            CallResponse response1 = MyUtil.makeCall(entry
                        .getAgency(), entry.getId(), myCall);

                  }
                  System.out.println(v.size() + " agents found.");
%>


<font size=3> Thank you for voting, your vote has been successfully
received : <br>
<br>

You have chosen <%=choice%> <%
```

```
 } else {
 %> <br />
<br />
<h1>You have to vote first</h1>
Go <a href="voting.jsp">here</a> to vote <%
      }
      }
%>

</body>
</html>
```

Κώδικας 8. confirm.jsp

```
<%@ page isErrorPage="true" import="java.io.*"%>
<%

      response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR
,
                  exception.toString());
%>
<%
response.setHeader("customerror", exception.toString());
%>
<html>
<head>
<title>Exception Even Occurred!</title>
</head>
<body>
<%-- Exception Handler --%>
<font size="-1" face="Verdana, Arial, Helvetica, sans-serif"
color="red"> <b><%=exception.toString()%></b><br />
</font>
<%
            if (exception.toString().indexOf("OutOfMemoryError") > 0)
{
            out.println("<br/>Switch to CFUPARSER, by adding the
following to your JSP :<br/>");
            out.println("&lt;jsp:setProperty name=\"upBean\"
property=\"parser\" value=\"&lt;%= MultipartFormDataRequest.CFUPARSER
%&gt;\"/&gt;<br/>");
            out.println("&lt;jsp:setProperty name=\"upBean\"
property=\"parsertmpdir\" value=\"c:/temp\"/&gt;<br/>");
      } else if (exception.toString().indexOf("JasperException") > 0)
{
            out.println("<br/>Check UploadBean properties in your
JSP.<br/> For folder store check : ");
            out.println(" &lt;jsp:setProperty name=\"upBean\"
property=\"folderstore\" value=\"c:/uploads\" /&gt;<br/>");
      }
      out.println("<!--");
      StringWriter sw = new StringWriter();
      PrintWriter pw = new PrintWriter(sw);
      exception.printStackTrace(pw);
      out.print(sw);
      sw.close();
      pw.close();
      out.println("-->");
%>
</body>
```

```
</html>
```

Κώδικας 9. ExceptionHandler.jsp

```
<%@ page import="java.io.*"%>
<%
	int counter1 = 0;
	int counter2 = 0;
	int counter3 = 0;
	String thisLine;

	try {
		FileInputStream fin = new FileInputStream(
		"C:\\election_results.txt");
		BufferedReader myInput = new BufferedReader(
		new InputStreamReader(fin));

		while ((thisLine = myInput.readLine()) != null) {
			if (thisLine.equals("A")) {

		counter1++;
			} else if (thisLine.equals("B")) {
		counter2++;
			} else {
		counter3++;
			}
		}
		System.out.println("A   : " + counter1);
		System.out.println("B   : " + counter2);
		System.out.println("C   : " + counter3);

	} catch (Exception e) {
		e.printStackTrace();
	}
%>

<span
	style="color: black; font-family: 'Times New Roman'; font-size:
large; font-style: normal; font-weight: bold; text-align: center;
text-decoration: underline">E-VOTING
SYSTEM : FINAL RESULTS</span>
<br />
<br />
<FORM METHOD="POST" ACTION="http://www.cs.tut.fi/cgi-
bin/run/~jkorpela/echo.cgi">
<TABLE BORDER="1">
	<TR>
		<TD>A</TD>
		<TD><INPUT TYPE="TEXT" NAME="name" VALUE=<%=counter1%>
SIZE="20"></TD>
	</TR>
	<TR>
		<TD>B</TD>
		<TD><INPUT TYPE="TEXT" NAME="name" VALUE=<%=counter2%>
SIZE="20"></TD>
	</TR>
	<TR>
		<TD>C</TD>
```

```
            <TD><INPUT TYPE="TEXT" NAME="name" VALUE=<%=counter3%>
SIZE="20"></td>
      </tr>
</table>
</FORM>
```

Κώδικας 10. results.jsp

```
import gr.ntua.ifouk.agent.MobileAgent;
import gr.ntua.ifouk.agent.MoveException;
import gr.ntua.ifouk.agent.ThreadInterrupt;
import java.security.InvalidKeyException;
import java.security.KeyPairGenerator;
import java.security.KeyPair;
import java.security.NoSuchAlgorithmException;
import java.security.PublicKey;
import java.security.PrivateKey;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.NoSuchPaddingException;
import java.security.*;
import java.io.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import java.security.*;
import java.security.spec.*;
import sun.misc.*;


public class CryptoAgent extends MobileAgent {

      private static final int START = 0;

      private static final int REMOTE = 1;

      private static final int BACK_TO_HOME = 2;

      private boolean hasVote = false;

      private String vote = null;

      transient BASE64Encoder base64Encoder;

      transient BASE64Decoder base64Decoder;

      String myName = "CryptoAgent";

      String xform = "RSA";

      int state = START;

      String encodedEncryptedVoteString;


      byte[] encryptedVoteBytes;

      byte[] sigBytes;

      PublicKey pubk = null;
```

```
      PrivateKey prvk = null;

      public void init() {

            System.out.println("CryptoAgent: A new agent is
born!!!");

            myName = "CryptoAgent";
            this.getInfo().setName(myName);

      }

      public void start() throws ThreadInterrupt {

            try {

                  Thread.sleep(1000);

                  switch (state) {

                  case START:
                        //perimene mexri na labeis psifo
                        while (!hasVote) {

                              try {

                                    Thread.sleep(4000);

                              } catch (Exception e) {

                              }

                        }
                        //kryptografisi psifou
                        PublicKey pbk_cipher = (PublicKey)
read("C:\\publickey_cipher.dat");
                        SecureRandom random = new SecureRandom();
                        Cipher cipher = Cipher.getInstance(xform);

                        cipher.init(Cipher.ENCRYPT_MODE, pbk_cipher,
random);

                        base64Encoder = new BASE64Encoder();
                        base64Decoder = new BASE64Decoder();

                        byte[] voteBytes = vote.getBytes();
                        encryptedVoteBytes =
cipher.doFinal(voteBytes);
                        this.encodedEncryptedVoteString =
base64Encoder
                                    .encode(encryptedVoteBytes);

      System.out.println(this.encodedEncryptedVoteString);


//ypografi

                        KeyPair kp = (KeyPair)
read("C:\\KeyPair.dat");
```

```
                        Signature signature =
Signature.getInstance("SHA1withRSA");
                        signature.initSign(kp.getPrivate(), new
SecureRandom());

                        signature.update(encryptedVoteBytes);
                        sigBytes = signature.sign();
                        System.out.println(sigBytes);

                        System.out.println("EIMASTE AKOMA EDO");

                        state = REMOTE;

                        move("http://193.92.82.161:8080", "Default
Place");

                        break;

                  case REMOTE://gia na ftanoume edw exei ginei
metakinhsh se remote agency



      System.out.println("====================================");

                        System.out.println("ftasame");

                        System.out.println("State is " + state);


      System.out.println("====================================");

                        //elegxos ypografis

                        KeyPair kp1 = (KeyPair)
read("C:\\KeyPair.dat");
                        Signature signature1 =
Signature.getInstance("SHA1withRSA");
                        signature1.initVerify(kp1.getPublic());
                        signature1.update(encryptedVoteBytes);
                        if (signature1.verify(sigBytes)) {
                              System.out.println("signature
verification succeeded");
                        } else {
                              System.out.println("failed");
                        }

                        KeyPair kp_cipher1 = (KeyPair)
read("C:\\KeyPair_cipher.dat");

                        //apokryptografisi psifou
                        Cipher decipher = Cipher.getInstance(xform);
                        decipher.init(Cipher.DECRYPT_MODE,
kp_cipher1.getPrivate());
                        base64Decoder = new BASE64Decoder();
                        byte[] encryptedVoteBytes1 = base64Decoder

      .decodeBuffer(this.encodedEncryptedVoteString);
```

```
                        byte[] voteBytes1 =
decipher.doFinal(encryptedVoteBytes1);
                        String recoveredVoteString = new
String(voteBytes1);
                        System.out.println(recoveredVoteString);
                        append(recoveredVoteString);

                        state = BACK_TO_HOME;

                        move(this.getInfo().getHomeHost(),
this.getInfo()
                                    .getHomePlace());

                        break;

                  }

            }

            catch (MoveException me) {

                  System.out.println("****************************");

                  System.out.println("Move exception:");

                  System.out.println("error moving agent: " +
me.getMessage());

                  System.out.println("****************************");

            } catch (Exception e) {

                  if (e instanceof ThreadInterrupt)

                        throw (ThreadInterrupt) e;

                  System.out.println("****************************");

                  System.out.println("error moving agent: " +
e.getMessage());

                  System.out.println("****************************");

                  e.printStackTrace();

            }

      }

      public void destroy() {

            System.out.println(" Bye bye!!!");

      }

      public void afterMove() {

            }

      public void beforeMove() {
```

```
	}

	public String receiveVote(String vote) {

		this.vote = vote;
		this.hasVote = true;
		return this.vote;
	}

	private static Object read(String filename) throws
ClassNotFoundException,
			IOException {
		ObjectInputStream in = null;

		try {
			in = new ObjectInputStream(new
FileInputStream(filename));

			return in.readObject();
		} finally {
			if (in != null) {
				try {
					in.close();
				} catch (IOException exception) {
				}
			}
		}

	}

	public static void append(String sn) {
		try {
			String filename = "C:\\election_results.txt";
			boolean append = true;
			FileWriter fw = new FileWriter(filename, append);

			fw.write(sn + "\n");//appends the string to the
file
			fw.close();
		}

		catch (IOException ioe) {
			System.err.println("IOException: " +
ioe.getMessage());
		}

	}

	public String getName() {

		return myName;

	}
}
```

Κώδικας 11. CryptoAgent.java

```
import java.security.*;
import java.io.*;

public class GenerateKeys {
      public static void main(String[] args) throws
NoSuchAlgorithmException,
                  IOException {
            KeyPairGenerator keyGen =
KeyPairGenerator.getInstance("RSA");
            keyGen.initialize(1024, new SecureRandom());
            KeyPair keyPair = keyGen.generateKeyPair();
            PublicKey pbk = keyPair.getPublic();
            write(pbk, "C:\\publickey_cipher.dat");

      }

      private static void write(Serializable object, String filename)
                  throws IOException {
            ObjectOutputStream out = null;

            try {
                  out = new ObjectOutputStream(new
FileOutputStream(filename));

                  out.writeObject(object);
                  out.flush();
            } finally {
                  if (out != null) {
                        try {
                              out.close();
                        } catch (IOException exception) {
                        }
                  }
            }
      }
}
```

Κώδικας 12. GenerateKeys.java

```
package pack;

import gr.ntua.ifouk.agent.rpc.*;
import gr.ntua.ifouk.agent.*;

public class MyUtil {

      public static CallResponse makeCall(String agency, AgentID id,
Call call) {

            //Callback calls

            if (call instanceof CallBackCall) {
```

```
                  return new CallResponse(agency, id, (CallBackCall)
call);

            }

            //Synch call

            if (call instanceof SynchCall) {

                  CallResponse response = new CallResponse(agency,
id, call);

                  while (!response.hasFinished()) {

                        try {

                              java.lang.Thread.sleep(500);

                        }

                        catch (Exception e) {

                              System.err.println("Thread interrupted
while waiting for result...");

                        }

                  }

                  return response;

            }

            //Asynch call

            return new CallResponse(agency, id, call);

      }

}
```

Κώδικας 13. MyUtil.java

```
package pack;

import java.util.*;
import java.io.*;
import java.sql.*;
import pack.FileInput;

public class myDatabaseChecker {

      // enhmerwsh endiamesou pinaka

      public static void main(String[] args) {

            java.util.Timer timer = new java.util.Timer();
            TimerTask update_localDatabase = new TimerTask() {
```

```
                  public void run() {

                        // The JDBC code
                        try {
                              String database = "evoting";
                              String username_db = "root";
                              String password_db = "240184";
                              String url =
"jdbc:mysql://212.251.127.50/evoting";

                              // Fortwsi toy Driver
                              try {

      Class.forName("com.mysql.jdbc.Driver").newInstance();
                              } catch (ClassNotFoundException cnfe) {
                                    System.err.println("Den
fortwthike swsta o JDBC driver!");
                                    cnfe.printStackTrace();
                                    System.exit(1);
                              } catch (InstantiationException e) {
                                    // TODO Auto-generated catch
block
                                    e.printStackTrace();
                              } catch (IllegalAccessException e) {
                                    // TODO Auto-generated catch
block
                                    e.printStackTrace();
                              }

                              // syndesi sti database
                              Connection connection = null;
                              try {
                                    connection =
DriverManager.getConnection(url,
                                                username_db,
password_db);
                              } catch (SQLException se) {
                                    System.out
                                                .println("Den
kataferame na syndethoume me to server.");
                                    se.printStackTrace();
                                    System.exit(1);
                              }// Dimiourgia Statement
                              try {
                                    Statement st1 =
connection.createStatement();
                                    String table = "CREATE TABLE
voter_table_endiameso(serial_num varchar(40), has_voted
tinyint(3),primary key(serial_num))";
                                    st1.executeUpdate(table);
                                    System.out.println("Table
creation process successfully!");

                                    st1.close();
                              } catch (SQLException ex) {
                                    System.out.println("exception: "
+ ex.getMessage());
                                    ex.printStackTrace();
                              }

                              ResultSet results = null;
```

```
                              try {

                                    Statement st2 =
connection.createStatement();
                                    String[] b =
FileInput.showVoters();
                                    for (int i = 0; i < 10; i++) {
                                          if (b[i] != "") {

      st2.executeUpdate("insert into voter_table_endiameso "

      + "values ('"

      + b[i]

      + "' ,1)  ");
                                          }
                                    }
                                    st2.close();

                              } catch (SQLException se1) {
                                    System.out.println("Lathos sto
SQL erwtima.");
                                    se1.printStackTrace();
                                    System.exit(1);

                              } catch (IOException e) {
                                    // TODO Auto-generated catch
block
                                    e.printStackTrace();
                              }
                              connection.close();
                        } catch (SQLException e) {
                        }

                  }
            };
            timer.schedule(update_localDatabase, 0, 50000);

      }

}
```

Κώδικας 14. myDatabaseChecker.java

```
package pack;

import java.io.*;
import java.util.*;

public class FileInput {
      public static void main(String[] args) throws IOException {
            String[] votersList = showVoters();
      }

      public static String[] showVoters() throws IOException {
            String[] voters;
            voters = new String[1000];
```

```
            for (int a = 0; a < 10; a++) {
                  voters[a] = "";
            }
            File file = new File("C:\\voters.txt");
            try {

                  String FStr = readFileAsString("C:\\voters.txt");
                  StringTokenizer pieces = new StringTokenizer(FStr);
                  String[] temp;
                  temp = new String[1000];
                  for (int a = 0; a < 10; a++) {
                        temp[a] = "";
                  }
                  while (pieces.hasMoreTokens()) {

                        for (int i = 0; i < temp.length; i++) {
                              temp[i] = pieces.nextToken();

                              voters[i] = temp[i];
                              // System.out.println(voters[i]);
                        }

                  }

            } catch (NoSuchElementException ee) {
            }

            finally {
                  file.delete();
            }
            for (int i = 0; i < voters.length; i++) {

                  System.out.println(voters[i]);

            }
            return voters;
      }



      public static String readFileAsString(String filePath)
                  throws java.io.IOException {
            StringBuffer fileData = new StringBuffer(1000);
            BufferedReader reader = new BufferedReader(new
FileReader(filePath));
            char[] buf = new char[1024];
            int numRead = 0;
            while ((numRead = reader.read(buf)) != -1) {
                  fileData.append(buf, 0, numRead);
            }
            reader.close();
            return fileData.toString();
      }

}
```

Κώδικας 15. FileInput.java

```
import java.util.*;

import java.util.*;

import java.util.*;
import java.io.*;

import java.sql.*;

public class final_update {

	public static void main(String[] args) {

		java.util.Timer timer = new java.util.Timer();
		TimerTask update_central = new TimerTask() {

			public void run() {

				//The JDBC code
				//enhmerwsh endiamesou pinaka

				try {
					String database = "evoting";
					String username_db = "root";
					String password_db = "240184";
					String url =
"jdbc:mysql://localhost/evoting";

					//Fortwsi toy Driver
					try {

	Class.forName("com.mysql.jdbc.Driver").newInstance();
					} catch (ClassNotFoundException cnfe) {
						System.err.println("Den
fortwthike swsta o JDBC driver!");
						cnfe.printStackTrace();
						System.exit(1);
					} catch (InstantiationException e) {
						// TODO Auto-generated catch
block
						e.printStackTrace();
					} catch (IllegalAccessException e) {
						// TODO Auto-generated catch
block
						e.printStackTrace();
					}

					//syndesi sti database
					Connection connection = null;
					try {
						connection =
DriverManager.getConnection(url,username_db, password_db);
					} catch (SQLException se) {
						System.out.println("Den
kataferame na syndethoume me to server.");
						se.printStackTrace();
```

```
                                    System.exit(1);
                              }//Dimiourgia Statement
                              try {
                                    ResultSet results = null;
                                    Statement st1 =
connection.createStatement();
                                    results =st1.executeQuery("select
voter_table_kedriko.serial_num,voter_table_endiameso.serial_num "
                                                + "from
voter_table_kedriko,voter_table_endiameso "
                                                + "where
voter_table_kedriko.serial_num =  voter_table_endiameso.serial_num
");
                                    while (results.next()) {

                                          String sn = results

      .getString("voter_table_kedriko.serial_num");
                                          System.out.println(sn);
                                          append(sn);
                                          Statement st2 =
connection.createStatement();
                                          st2.executeUpdate("update
voter_table_kedriko set has_voted=1  where serial_num ='"
                                                                      +
sn + "' ");
                                          st2.close();
                                    }
                                    st1.close();
                                    try {

                                          Statement st3 =
connection.createStatement();

                                          st3.executeUpdate("drop
table voter_table_endiameso");
                                          System.out.println("Table
Deletion process is completed successfully!");
                                          st3.close();

                                    } catch (SQLException se1) {
                                          System.out.println("Lathos
sto SQL erwtima.");
                                          se1.printStackTrace();
                                          System.exit(1);

                                    }
                              } catch (SQLException ex) {
                                    System.out.println("exception: "
+ ex.getMessage());
                                    ex.printStackTrace();
                              }

                              connection.close();
                        } catch (SQLException el) {
                              System.out.println(el.getMessage());
                        }
                        //enimerwsi perifereikwn basewn
                        try {
                              String database = "evoting";
                              String username_db = "root";
```

```
                              String password_db = "240184";
                              String url =
"jdbc:mysql://147.102.9.13/evoting";

                              //Fortwsi toy Driver
                              try {

      Class.forName("com.mysql.jdbc.Driver").newInstance();
                              } catch (ClassNotFoundException cnfe) {
                                    System.err
                                                .println("Den
fortwthike swsta o JDBC driver!");
                                    cnfe.printStackTrace();
                                    System.exit(1);
                              } catch (InstantiationException e) {
                                    // TODO Auto-generated catch
block
                                    e.printStackTrace();
                              } catch (IllegalAccessException e) {
                                    // TODO Auto-generated catch
block
                                    e.printStackTrace();
                              }

                              //syndesi sti database
                              Connection connection = null;
                              try {
                                    connection =
DriverManager.getConnection(url,username_db, password_db);
                              } catch (SQLException se) {
                                    System.out.println("Den
kataferame na syndethoume me to server.");
                                    se.printStackTrace();
                                    System.exit(1);
                              }//Dimiourgia Statement
                              try {
                                    ResultSet results = null;
                                    /*Statement st1 =
connection.createStatement();
                                    results =st1.executeQuery("select
voter_table.serial_num,voter_table_endiameso.serial_num "
                                                + "from
voter_table,voter_table_endiameso "
                                                + "where
voter_table.serial_num =  voter_table_endiameso.serial_num ");
                                    while (results.next()) {

                                          String sn = results

      .getString("voter_table_kedriko.serial_num");
                                          System.out.println(sn);*/
                                    String[] b = showVoters();
                                    Statement st1 =
connection.createStatement();
                                    for (int i = 0; i < 10; i++) {
                                          if (b[i] != "") {

      st1.executeUpdate("update voter_table set has_voted=1  where
serial_num ='"

      + b[i] + "' ");
```

```
                                        }
                                  }
                                  st1.close();

                            } catch (SQLException ex) {
                                  System.out.println("exception: "
+ ex.getMessage());
                                  ex.printStackTrace();
                            } catch (IOException e) {
                                  // TODO Auto-generated catch
block
                                  e.printStackTrace();
                            }

                            connection.close();
                      } catch (SQLException e) {
                            System.out.println(e.getMessage());
                      }

                }
          };
          timer.schedule(update_central, 0, 60000);

    }

    public static void append(String sn) {
          try {
                String filename = "C:\\voters_kedriko.txt";
                boolean append = true;
                FileWriter fw = new FileWriter(filename, append);

                fw.write(sn + "\n");//appends the string to the
file
                fw.close();
          }

          catch (IOException ioe) {
                System.err.println("IOException: " +
ioe.getMessage());
          }

    }

    public static String[] showVoters() throws IOException {
          String[] voters;
          voters = new String[1000];
          for (int a = 0; a < 10; a++) {
                voters[a] = "";
          }
          File file = new File("C:\\voters_kedriko.txt");
          try {

                String FStr =
readFileAsString("C:\\voters_kedriko.txt");
                StringTokenizer pieces = new StringTokenizer(FStr);
                String[] temp;
                temp = new String[1000];
                for (int a = 0; a < 10; a++) {
                      temp[a] = "";
                }
                while (pieces.hasMoreTokens()) {
```

```
                        for (int i = 0; i < temp.length; i++) {
                              temp[i] = pieces.nextToken();

                              voters[i] = temp[i];
                              // System.out.println(voters[i]);
                        }

                  }

            } catch (NoSuchElementException ee) {
            }

            finally {
                  file.delete();
            }
            for (int i = 0; i < voters.length; i++) {

                  System.out.println(voters[i]);

            }
            return voters;
      }


      public static String readFileAsString(String filePath)
                  throws java.io.IOException {
            StringBuffer fileData = new StringBuffer(1000);
            BufferedReader reader = new BufferedReader(new
FileReader(filePath));
            char[] buf = new char[1024];
            int numRead = 0;
            while ((numRead = reader.read(buf)) != -1) {
                  fileData.append(buf, 0, numRead);
            }
            reader.close();
            // System.out.println(fileData.toString());
            return fileData.toString();
      }

}
```

Κώδικας 16. final_update.java

```
--
-- Create schema evoting
--

CREATE DATABASE IF NOT EXISTS evoting;
USE evoting;

--
-- Definition of table `voter_table`
--

DROP TABLE IF EXISTS `voter_table`;
CREATE TABLE `voter_table` (
  `serial_num` varchar(45) NOT NULL,
```

```
  `username` varchar(45) NOT NULL,
  `passwd` varchar(45) NOT NULL,
  `has_voted` tinyint(3) unsigned NOT NULL,
  `name` varchar(45) NOT NULL,
  `surname` varchar(45) NOT NULL,
  `father's name` varchar(45) NOT NULL,
  `birth_date` varchar(45) NOT NULL,
  `id_number` varchar(45) NOT NULL,
  PRIMARY KEY  (`serial_num`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;




--
-- Definition of table `voter_table_kedriko`
--

DROP TABLE IF EXISTS `voter_table_kedriko`;
CREATE TABLE `voter_table_kedriko` (
  `serial_num` varchar(40) NOT NULL,
  `has_voted` tinyint(3) unsigned NOT NULL,
  PRIMARY KEY  (`serial_num`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Κώδικας 17. Δημιουργία πινάκων SQL