ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Μηχανισμοί ασφάλειας και υπηρεσίες προστασίας δεδομένων σε μη σχεσιακές βάσεις δεδομένων (Δημιουργία της Fork Crypt-MongoDB)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πενταράκης Ν. Εμμανουήλ

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2017

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Μηχανισμοί ασφάλειας και υπηρεσίες προστασίας δεδομένων σε μη σχεσιακές βάσεις δεδομένων (Δημιουργία της Fork Crypt-MongoDB)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πενταράκης Ν. Εμμανουήλ

Επιβλέπων : Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Σεπτεμβρίου 2017.

....

Αθήνα, Σεπτέμβριος 2017

.....
Πενταράκης Ν. Εμμανουήλ

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

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι να μελετήσει και να παρουσιάσει με ένα σύντομο αλλά ολοκληρωμένο τρόπο τους μηχανισμούς ασφάλειας και τις υπηρεσίες προστασίας που πρέπει να παρέχει ένα σύστημα πληροφορικής για να θεωρηθεί ασφαλές. Το σύστημα πρέπει να υλοποιεί κάποια διεθνή πρότυπα για την προστασία των δεδομένων και να είναι ταυτόχρονα χρηστικό και γρήγορο.

Αρχικά θα μελετηθούν όλες οι πτυχές που αφορούν τον κλάδο της ασφάλειας και τους μηχανισμούς προστασίας τον δεδομένων τόσο σε θεωρητικό, όσο και σε πρακτικό επίπεδο. Τέτοιοι μηχανισμοί είναι για παράδειγμα η προστασία των δεδομένων από υποκλοπή, αλλοίωση και μη διαθεσιμότητα, η παράνομη και μη επιθυμητή πρόσβαση τρίτων σε κάποια προσωπικά δεδομένα, η μη αποποίηση των ευθυνών όταν κάποιος γράφει δεδομένα σε μια βάση, κτλ. Η χρήση μιας μη σχεσιακής βάσης δεδομένων κρίθηκε απαραίτητη, τόσο για να συμβαδίζουμε με τις νέες τάσεις της τεχνολογίας, αλλά και λόγω της φύσης του προβλήματος.

Στη συνέχεια θα εξετασθούν διάφορες θεωρητικές έννοιες στον τομέα της ασφάλειας δεδομένων, όπως η εξουσιοδότηση χρηστών, η εμπιστευτικότητα, η ακεραιότητα και η διαθεσιμότητα των δεδομένων. Η υλοποίηση των παραπάνω εννοιών με συγκεκριμένες τεχνικές και μεθόδους είναι επίσης αντικείμενο αυτής της διπλωματικής.

Σχεδιάστηκε επίσης μια διεπαφή προγραμματισμού εφαρμογών και μια πρότυπη εφαρμογή που να χρησιμοποιεί την προηγούμενη διεπαφή. Η διεπαφή αυτή περικλείει τους μηχανισμούς ασφάλειας δεδομένων που μελετήθηκαν προηγουμένως και υλοποιεί τους κατάλληλους αλγόριθμους για να επιτύχει υψηλή προστασία δεδομένων.

Λέξεις Κλειδιά

Μηχανισμοί ασφάλειας, προστασία δεδομένων, μεγάλα δεδομένα, μη-σχεσιακή βάση δεδομένων, MongoDB, πιστοποίηση, εξουσιοδότηση, εμπιστευτικότητα, ακεραιότητα, διαθεσιμότητα, μη-αποποίηση ευθυνών

Abstract

The purpose of this diploma thesis is to research and showcase all the different security mechanisms and services an IT system must implement to be considered secure. A system like this should follow some international security standards for data protection and it should be fast and easy to use.

All the aspects and mechanisms of data security will initially be researched in both theoretical and practical levels. Such mechanisms include protection from data loss, corruption and theft, protection from illegal and unwanted access of third parties to personal and sensitive data, conserving non-repudiation in the digital environment, etc. The use of a non-relational database (NoSQL) was deemed necessary because of the nature of the problem and the new trends in the database environment.

Additional security concepts will also be studied in depth, like user authentication and authorization, data confidentiality, data integrity and availability. The implementation of the above theoretical concepts with practical and technical methods and algorithms was crucial to composing a secure and safe database system.

Designing and developing an application programming interface (API) and a prototype application, which uses the previous API, was also a very important part of the thesis. This API includes all the security services and mechanisms that were analyzed before and achieves high levels of security in a NoSQL database.

Keywords

Security mechanisms, data protection, big data, NoSQL database, MongoDB, authentication, authorization, confidentiality, integrity, availability, non-repudiation

Ευχαριστίες

Η εκπόνηση της παρούσας διπλωματικής εργασίας έγινε στο εργαστήριο Distributed Knowledge and Media Systems Group της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών και με τη συγγραφή της κλείνει ο πενταετής κύκλος φοίτησής μου.

Με την ευκαιρία αυτή θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια Θεοδώρα Βαρβαρίγου, καθώς και τον υποψήφιο διδάκτορα Βρεττό Μουλό, για το θέμα που μου εμπιστεύτηκαν και για τη βοήθεια και την καθοδήγηση που μου έδωσαν κατά τη διάρκεια της δημιουργίας της εργασίας αυτής.

Ευχαριστώ επίσης τους συμφοιτητές και φίλους μου για την πολύτιμη βοήθεια και συνεργασία που είχαμε καθ'όλη τη διάρκεια της φοίτησής μας. Ευχαριστώ ακόμα τον φίλο μου Σάμι για τον υλικό εξοπλισμό που μου παρείχε και διευκόλυνε την ολοκλήρωση της παρούσας εργασίας.

Τέλος θα ήθελα να ευχαριστήσω και εσένα αναγνώστη, που είσαι από τους λίγους ανθρώπους που θα διαβάσουν αυτή τη διπλωματική εργασία. Να έχεις μια *όμορφη* μέρα.

Μανώλης "NoLee" Πενταράκης

Αθήνα, Αύγουστος 2017

Πίνακας περιεχομένων

ΚΕΦΑΛΑΙΟ 4

ΚΕΦΑΛΑΙΟ 5

ΚΕΦΑΛΑΙΟ 6

Πίνακας διαγραμμάτων

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

Στη σημερινή εποχή της πληροφορίας η ανταλλαγή δεδομένων αποτελεί σημαντικό και αναπόσπαστο κομμάτι της καθημερινής ζωής. Αυτό γίνεται καλύτερα σαφές αν σκεφτούμε την εξάπλωση του διαδικτύου και το μεγάλο όγκο δεδομένων που κυκλοφορούν (Big Data). Μέσα σε όλη αυτή την πληροφορία που διανέμεται υπάρχουν πολλά ευαίσθητα και προσωπικά δεδομένα και η πρόσβαση σε αυτά πρέπει να γίνεται με μεγάλη προσοχή και ασφάλεια.

Μία περίπτωση που φαίνεται καθαρά το προηγούμενο πρόβλημα της προστασίας των δεδομένων είναι όταν επιχειρήσεις συνεργάζονται με τρίτους οργανισμούς και άτομα για τη δημιουργία ενός συγκεκριμένου έργου, για την κάλυψη υπηρεσιών της εταιρίας ή όταν τα δεδομένα της επιχειρήσης τα χρησιμοποιούν διάφοροι χρήστες, χωρίς να ξέρουμε τις προθέσεις του καθενός.

Όταν κάποιος ακούει για ασφάλεια δεδομένων, η σκέψη του πηγαίνει κατευθείαν στην κρυπτογράφηση. Η εξίσωση της έννοιας προστασίας των δεδομένων με την κρυπτογράφηση αποτελεί μια λανθασμένη αντίληψη πολλών ανθρώπων, ακόμα και αυτών που ασχολούνται σε θέματα πάνω στην επιστήμη και την τεχνολογία των υπολογιστών. Σκοπός της διπλωματικής αυτής είναι να παρουσιάσει με ένα σύντομο αλλά ολοκληρωμένο τρόπο πως μπορεί ένα σύστημα πληροφορικής να θεωρηθεί ασφαλές, να υλοποιεί κάποια διεθνή στάνταρτ για την προστασία των δεδομένων και να είναι ταυτόχρονα χρηστικό και γρήγορο.

Θα μελετηθούν δηλαδή όλες οι πτυχές που αφορούν τον κλάδο και τους μηχανισμούς προστασίας τον δεδομένων τόσο σε πρακτικό, όσο και σε θεωρητικό επίπεδο. Τέτοιοι μηχανισμοί είναι για παράδειγμα η προστασία από υποκλοπή, αλλοίωση και μη διαθεσιμότητα δεδομένων, η παράνομη και μη επιθυμητή πρόσβαση τρίτων σε κάποια προσωπικά δεδομένα, η μη αποποίηση των ευθυνών όταν κάποιος γράψει κάποια δεδομένα κτλ.

1.1 Οργάνωση της διπλωματικής

Η παρούσα διπλωματική απαρτίζεται από 6 κεφάλαια που δομούνται με τον παρακάτω τρόπο.

Στο *κεφάλαιο 2* γίνεται παρουσίαση των διάφορων θεωρητικών και τεχνολογικών εννοιών που πραγματεύεται η διπλωματική. Αναλύονται τα διάφορα είδη βάσεων δεδομένων μαζί με τα πλεονεκτήματα και μειονεκτήματά τους καθώς και η βάση που χρησιμοποιήθηκε για την επίλυση του προβλήματος, η MongoDB.

Στο *κεφάλαιο 3* γίνεται η ανάλυση του προβλήματος που χρήζει αντιμετώπισης και παρουσιάζονται όλοι οι μηχανισμοί ασφαλείας που πρέπει να εξεταστούν (εξουσιοδότηση και πιστοποίηση χρηστών, διασφάλιση εμπιστευτικότητας – ακεραιότητας – διαθεσιμότητας δεδομένων, μη αποκήρυξη ευθυνών, κτλ). Ως πρόβλημα θεωρήσαμε την προστασία των δεδομένων που αποθηκεύονται σε έναν ιατρικό οργανισμό, γνωρίζοντας ότι τα δεδομένα που διαχειρίζεται είναι ευαίσθητα και πρέπει να προστατεύονται από οποιαδήποτε απειλή.

Στο *κεφάλαιο 4* αναλύονται τεχνικά οι μηχανισμοί που αναφέρθηκαν στα προηγούμενα κεφάλαια και παρουσιάζονται συγκεκριμένοι αλγόριθμοι και τεχνικές που χρησιμοποιήθηκαν για την επίλυση κάθε ζητήματος στον τομέα της ασφάλειας. Ενδεικτικά χρησιμοποιήθηκαν το πρωτόκολλο Kerberos, η κρυπτογράφηση δημοσίου κλειδιού RSA, οι ψηφιακές υπογραφές, replication δεδομένων και διάφορες άλλες τεχνικές.

Στο *κεφάλαιο 5* παρουσιάζεται με ακρίβεια η αρχιτεκτονική του συστήματος, καθώς και η διεπαφή που δημιουργήθηκε ως fork της MongoDB (Crypt-MongoDB). Αναλύονται με ακρίβεια οι μέθοδοι που παρέχει το API που κατασκευάστηκε και οι λειτουργικές απαιτήσεις της εφαρμογής που δημιουργήθηκε. Η δημιουργία μιας εφαρμογής με γραφικό περιβάλλον έγινε για την παρουσίαση των αποτελεσμάτων, έλεγχο του API και εξέταση των προηγούμενων μηχανισμών και τεχνικών.

Τέλος, στο *κεφάλαιο 6* γίνεται η αξιολόγηση της διπλωματικής εργασίας και της λύσης που προτάθηκε, καθώς επίσης αναφέρονται μελλοντικές επεκτάσεις για περαιτέρω βελτίωση της πλατφόρμας.

ΚΕΦΑΛΑΙΟ 2

Τεχνολογίες και θεωρητικές έννοιες

Όπως αναφέραμε και στην εισαγωγή, το πλήθος των δεδομένων που δημιουργούνται αυξάνεται με εκθετικούς ρυθμούς τα τελευταία χρόνια. Για την αποτελεσματική διαχείριση και αποθήκευσή τους δεν αρκούν οι συμβατικές τεχνικές που χρησιμοποιούσαμε παλαιότερα, καθώς δεν ήταν σχεδιασμένες για την επεξεργασία τόσο μεγάλου όγκου δεδομένων.

Σε αυτό το κεφάλαιο αναλύουμε σύντομα τι είδους αλλαγές φέρνει η εποχή αυτή στη διαχείριση των δεδομένων, πως και με τι εργαλεία μπορούμε να επιλύσουμε τα προβλήματα που προκύπτουν και πως συνδέονται όλα αυτά με το σύνολο των μηχανισμών ασφάλειας που θέλουμε να υλοποιήσουμε.

2.1 Νέα εποχή, wearable devices και Internet of Things

Οι ηλεκτρονικές συσκευές και γενικότερα η τεχνολογία μπαίνει όλο και περισσότερο στις ζωές των ανθρώπων τα τελευταία χρόνια. Έξυπνα κινητά, τηλεοράσεις, αυτοκίνητα και σπίτια είναι μερικά από τα καθημερινά αντικείμενα που έχουν μεταβληθεί από την επιρροή της τεχνολογίας των υπολογιστών.

Οι συσκευές αυτές, καθώς και πολλές ακόμα, έχουν τη δυνατότητα να επικοινωνούν μεταξύ τους, να ανταλλάσσουν δεδομένα, να τα επεξεργάζονται σε συνεργασία με άλλες εφαρμογές και να τα αποθηκεύουν σε ένα απομακρυσμένο και ασφαλές περιβάλλον που είναι πάντα διαθέσιμο και προσβάσιμο. Η επικοινωνία αυτή μεταξύ διαφορετικών συσκευών ονομάζεται Διαδίκτυο των Πραγμάτων (Internet of Things – IoT).

Τα δεδομένα που δημιουργούν οι παραπάνω συσκευές είναι πάρα πολλά αν σκεφτεί κανείς τις υπηρεσίες που παρέχουν. Μερικά παραδείγματα είναι ένας ηλεκτρονικός θερμοστάτης που μετρά τη θερμοκρασία του σπιτιού κάθε δευτερόλεπτο, ένα κινητό που μετρά τα βήματα που κάνει ένας άνθρωπος κάθε μέρα ή μια συσκευή πάνω ή μέσα από το

δέρμα που μπορεί να μετρά τους παλμούς και άλλα ανθρώπινα βιολογικά χαρακτηριστικά συνεχώς. Και οι τρεις προηγούμενες συσκευές μπορούν να επικοινωνούν μεταξύ τους και να αλλάζουν για παράδειγμα τη θερμοκρασία του σπιτιού ανάλογα με την κατάσταση του ανθρώπου (αν έχει τρέξει 10 χιλιόμετρα και έχει κουραστεί, αν είναι άρρωστος κτλ).

Κάποια από τα προηγούμενα μηχανήματα, όπως τα ρολόγια ή οι ηλεκτρονικοί μετρητές μέσα στο σώμα, χρησιμοποιούνται σήμερα και σε πολλούς τομείς της υγείας καθώς και από την αστυνομία για μεγαλύτερη ασφάλεια των ανθρώπων. Η χρήση έξυπνων καρτών σε υπηρεσίες περίθαλψης είναι διαδεδομένη, καθώς αποθηκεύονται σε αυτές κρίσιμα δεδομένα για τον κάτοχο και λειτουργούν σαν ταυτότητες του ψηφιακού κόσμου. Ο μεγάλος όγκος των δεδομένων που δημιουργούν ονομάζεται big data και η τοποθεσία που συνήθως τα αποθηκεύουν ονομάζεται cloud. Στη συνέχεια δίνονται περισσότερες λεπτομέρειες για αυτές τις ορολογίες.

2.2 Big data και Cloud computing

Το πρόβλημα των Big data ξεκίνησε στις αρχές του 21^{ου} αιώνα μαζί με την εξάπλωση της ψηφιακής τεχνολογίας. Από τους πρώτους κλάδους που ανέδειξαν το πρόβλημα αυτό ήταν οι μηχανές αναζήτησης και τα κοινωνικά δίκτυα. Για να καταλάβουμε καλύτερα την έκταση του προβλήματος αρκεί να σκεφτούμε ότι η Google αναλύει mouseclicks, links και περιεχόμενο 1.5 τρισεκατομμυρίου σελίδων κάθε μέρα και παρουσιάζει τα αποτελέσματα αυτής της ανάλυσης σε εκατοστά του δευτερολέπτου.

Για τον ορισμό της έννοιας των Big data χρησιμοποιούμε τον χαρακτηρισμό των 7"V"s (Volume, Velocity, Variety, Value τα σημαντικότερα και Veracity, Visualization, Variability τα δευτερεύοντα)[12]. Για να κατανοήσουμε αν ένα πρόβλημα σχετίζεται με τα Big data, αρκεί να δούμε αν περιέχει τις παρακάτω παραμέτρους.

Όγκος (Volume)

Αφορά την επεξεργασία μεγάλου όγκου δεδομένων χαμηλής πυκνότητας και τη μετατροπή τους σε δεδομένα υψηλής πυκνότητας, δηλαδή σε δεδομένα που έχουν κάποια αξία για τους ανθρώπους. Παραδείγματα τέτοιων δεδομένων είναι τα mouseclicks σε μια

σελίδα, η κίνηση δεδομένων σε ένα δίκτυο και η καταγραφή δεδομένων από αισθητήρες κάθε χιλιοστό του δευτερολέπτου. Γενικά αναφερόμαστε σε τάξεις μεγέθους από μερικά terabyte ως και εκατοντάδες petabyte.

Ταχύτητα (Velocity)

Αφορά την ταχύτητα με την οποία δεχόμαστε τα δεδομένα και την ανάγκη για την επεξεργασία τους σε πραγματικό χρόνο. Εφαρμογές κινητής τηλεφωνίας που λαμβάνουν δεδομένα σε πραγματικό χρόνο, τα επεξεργάζονται και δίνουν αποτελέσματα καθώς και δεδομένα από αισθητήρες, όπου η άμεση ανταπόκριση είναι πολύ σημαντική, αποτελούν μερικά παραδείγματα.

Ποικιλία (Variety)

Αφορά μη δομημένους ή ημιδομημένους τύπους δεδομένων, όπως κείμενο, ήχος, βίντεο ή συνδυασμός αυτών, που χρειάζονται ανάλυση διαφορετική από τους δομημένους τύπους δεδομένων. Πολλές φορές στο πλαίσιο μιας γνωστής πηγής πληροφοριών μπορεί ο τύπος των δεδομένων να αλλάξει χωρίς προειδοποίηση και έτσι να παρουσιαστεί μεγαλύτερη πολυπλοκότητα.

Αξία (Value)

Αφορά το γεγονός ότι τα δεδομένα έχουν κρυφή αξία και πρέπει αυτή να ανακαλυφθεί μέσα από την επεξεργασία τους. Από όλα αυτά τα δεδομένα που συλλέγονται πρέπει να βρεθεί για παράδειγμα το εξάρτημα ενός συστήματος που πρόκειται να χαλάσει ή να γίνει μια πρόταση για τον καταναλωτή με βάση τα συναισθήματα ή τη συμπεριφορά του. Η μεγάλη πρόκληση για τα Big data είναι η αναγνώριση μοτίβων, η πρόβλεψη συμπεριφοράς και η λήψη συνειδητών υποθέσεων.

Έκτος από τα παραπάνω, για να ορίσουμε την έννοια των Big data αξίζει να αναφερθούμε και στα χαρακτηριστικά της μεταβλητότητας (Variability) που αφορά τη διαφορετική σημασιολογία παρόμοιων δεδομένων, την ακρίβεια (Veracity) των δεδομένων και την επεξεργασία πρέπει να περάσουν για να είναι χρήσιμα, καθώς και την απεικόνιση

(Visualization) των δεδομένων σε μια μορφή που να είναι αναγνώσιμη από τους ανθρώπους και να τους δίνει χρήσιμες πληροφορίες.

Σχήμα 1: Κύρια χαρακτηριστικά Μεγάλων δεδομένων (Big data)

Παράλληλα με την εξάπλωση των Big data έγινε και η διάδοση του Cloud computing. Όλα τα δεδομένα που αναφέραμε προηγουμένως χρειάζεται να είναι συνεχώς διαθέσιμα και να είναι προσβάσιμα από οπουδήποτε. Αυτό γίνεται με την αποθήκευση των δεδομένων στο διαδίκτυο χρησιμοποιώντας υπηρεσίες νέφους. Οι πάροχοι υπηρεσιών Cloud computing προσφέρουν τεράστιους χώρους αποθήκευσης δεδομένων τόσο σε μεμονωμένα άτομα, όσο και σε μεγάλες επιχειρήσεις.

Εκτός από τα προφανή θετικά χαρακτηριστικά του Cloud (διαθεσιμότητα δεδομένων, αποθήκευση στο διαδίκτυο, ταχύτητα επεξεργασίας κ.τ.λ.) υπάρχουν και κάποιες αρνητικές πτυχές, η σημαντικότερη των οποίων είναι η ασφάλεια των δεδομένων[19].

Η ανησυχία για την ασφάλεια των προσωπικών δεδομένων που είναι αποθηκευμένα στο Cloud πηγάζει από το γεγονός ότι δε γνωρίζουμε ακριβώς σε ποιο (φυσικό) σημείο βρίσκονται τα δεδομένα αυτά. Αυτό έχει μεγάλη σημασία, καθώς τα δεδομένα μπορεί να βρίσκονται σε πολλούς servers και σε διαφορετικές χώρες και η νομοθεσία προστασίας των δεδομένων να είναι διαφορετική από την αναμενόμενη.

Χωρίς λοιπόν να γνωρίζουμε που είναι αποθηκευμένα τα δεδομένα, δεν υπάρχει και έλεγχος για το ποιος έχει δικαίωμα να τα δει και να τα επεξεργαστεί. Πολλές φορές χωρίς να γίνεται αντιληπτό οι πάροχοι cloud υπηρεσιών προσφέρουν τα δεδομένα που έχουν στη διάθεση τους σε τρίτες επιχειρήσεις για διάφορους σκοπούς[20], όπως το marketing. Πρέπει λοιπόν να διασφαλιστεί ότι δε γίνεται κατάχρηση με οποιοδήποτε τρόπο.

Όπως είναι αναμενόμενο, οι υπηρεσίες Cloud χρησιμοποιούνται κατά κόρον για θέματα outsourcing. Σημαντικό παράδειγμα αποτελεί το γεγονός ότι πολλές επιχειρήσεις από τις Ηνωμένες Πολιτείες Αμερικής βασίζονται σε εξωτερικούς συνεργάτες στην Ευρώπη για την προστασία των δεδομένων τους[14]. Αυτό συμβαίνει καθώς στην Ευρώπη οι νόμοι αλλά και η καλύτερη τεχνογνωσία πάνω στον τομέα καθιστούν την προστασία και την αποθήκευση των δεδομένων στο cloud πολύ ασφαλέστερη.

Η εξάπλωση αυτή των Big data και του Cloud computing έχει οδηγήσει σε διάδοση δυο θεμελιωδών τεχνολογιών για την αποθήκευση και την επεξεργασία των δεδομένων, το framework Apache Hadoop και τις μη σχεσιακές βάσεις δεδομένων (NoSQL). Οι τεχνολογίες αυτές δεν είναι απαραίτητα διαφορετικές και συχνά χρησιμοποιούνται μαζί για την επίλυση διάφορων προβλημάτων. Εμείς θα εστιάσουμε περισσότερο την προσοχή μας στις μη-σχεσιακές βάσεις δεδομένων και δε θα ασχοληθούμε με τις υπηρεσίες που παρέχει το Apache Hadoop.

2.3 Βάσεις Δεδομένων

Στις προηγούμενες σελίδες αναφερθήκαμε αρκετά σε δεδομένα και στην ανάγκη για την αποτελεσματική αποθήκευση και προστασία τους. Η αποθήκευση των δεδομένων στους προσωπικούς υπολογιστές γίνεται με πολλούς τρόπους, όμως σε μεγαλύτερη κλίμακα που χρειάζεται καλύτερη οργάνωση και χρηστικότητα, τα δεδομένα αποθηκεύονται σε βάσεις δεδομένων.

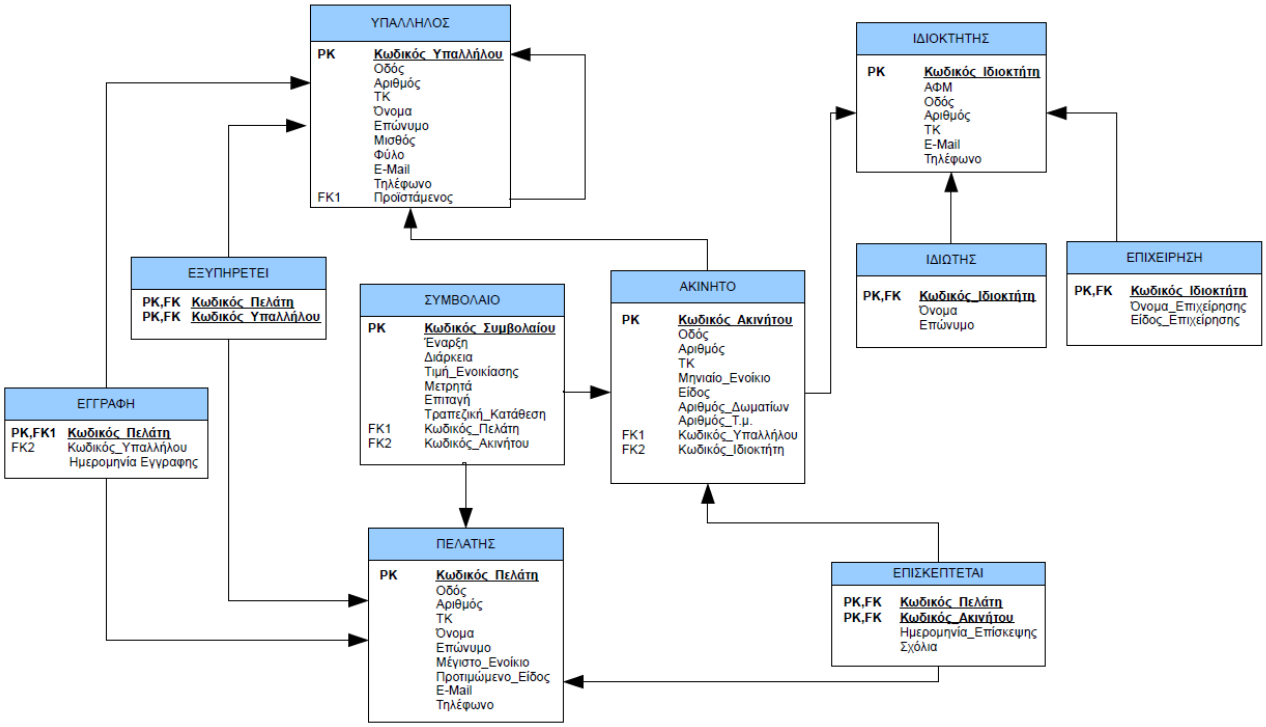
Βάση δεδομένων ονομάζεται μια συλλογή από δεδομένα που είναι οργανωμένη έτσι ώστε να γίνεται ευκολότερη η πρόσβαση, η διαχείριση και η ενημέρωσή τους[5]. Τα δεδομένα αυτά οργανώνονται με τη χρήση πινάκων (tables), ερωτημάτων (queries), όψεων (views), σχημάτων (schemas) της βάσης και άλλων αντικειμένων.

2.3.1 Σχεσιακές Βάσεις δεδομένων

Από τις πρώτες και πιο διαδεδομένες προσεγγίσεις είναι η σχεσιακή βάση δεδομένων (relational database), μια πινακοειδή βάση δεδομένων στην οποία τα δεδομένα ορίζονται έτσι ώστε να μπορούν να αναδιοργανωθούν και να προσεγγιστούν με έναν αριθμό διαφορετικών τρόπων. Οι πίνακες στις σχεσιακές βάσεις δεδομένων αποτελούνται από γραμμές και από στήλες. Οι στήλες έχουν συγκεκριμένο όνομα σε κάθε πίνακα και είναι καθορισμένου πλήθους, ενώ οι γραμμές μπορεί να είναι απεριόριστες και εκεί βρίσκονται τα δεδομένα.

Για τη διαχείριση των πινάκων στις σχεσιακές βάσεις δεδομένων χρησιμοποιείται μια εφαρμογή λογισμικού σχεδιασμένη να αλληλοεπιδρά με το χρήστη, με άλλες εφαρμογής και την ίδια τη βάση.

Αυτό το σύστημα διαχείρισης βάσεων δεδομένων (DBMS) είναι σχεδιασμένο να εκτελεί βασικές εργασίες για τα δεδομένα της βάσης, όπως η δημιουργία (create), η ανάγνωση (read), η ενημέρωση (update) και η διαγραφή (delete) δεδομένων, δηλαδή των CRUD λειτουργιών.



Σχήμα 2: Παράδειγμα σχήματος (schema) σχεσιακής βάσης δεδομένων. Το κάθε κουτί αποτελεί ένα πίνακα της βάσης που περιέχει τα αναγραφόμενα δεδομένα και τα βελάκια αποτελούν τις σχέσεις μεταξύ των πινάκων.

Για να επεξεργαζόμαστε τα δεδομένα των βάσεων δεδομένων υπάρχει μια προγραμματιστική γλώσσα ειδικού τύπου, η SQL. Βασισμένη στη σχεσιακή άλγεβρα και στο σχεσιακό λογισμό, με τη γλώσσα SQL μπορούμε να εκτελέσουμε όλες τις CRUD λειτουργίες με τη χρήση ερωτημάτων (queries) προς τη βάση. Φυσικά η SQL δίνει τη δυνατότητα για πολλές ακόμα λειτουργίες εκτός των βασικών που αναφέραμε.

Με τις αυξανόμενες ανάγκες για αποτελεσματική και γρήγορη διαχείριση των δεδομένων που έφερε η εποχή των Big data, οι σχεσιακές βάσεις άρχισαν να απαρχαιώνονται και να αντικαθίστανται από της μη-σχεσιακές NoSQL βάσεις δεδομένων. Οι σχεσιακές βάσεις είναι ακόμα αναγκαίες και χρησιμοποιούνται εκτενέστερα, όμως ανάλογα με το πρόβλημα και τις απαιτήσεις υπάρχουν πια και εναλλακτικές επιλογές, αντίθετα με παλαιότερα όπου οι SQL βάσεις αποτελούσαν μονόδρομο.

2.3.2 NoSQL Βάσεις δεδομένων

Το όνομα αυτών των βάσεων δεδομένων προέρχεται από συντόμευση της φράσης “not only SQL”, δηλαδή δεν αντικαθιστούν την SQL αλλά τη συμπληρώνουν και την επεκτείνουν.

Οι NoSQL βάσεις δεδομένων μπορούν να χωριστούν σε 4 μεγάλες κατηγορίες ανάλογα με το μοντέλο των δεδομένων στο οποίο στηρίζονται[2]. Μελετώντας τις ιδιαιτερότητες κάθε κατηγορίας μπορούμε να επιλέξουμε το βέλτιστο τύπο βάσης δεδομένων για κάποιο συγκεκριμένο πρόβλημα.

Μοντέλο Κλειδιού - Τιμής (Key - Value Store)

Το απλούστερο μοντέλο των NoSQL βάσεων δεδομένων. Τα δεδομένα αποτελούνται από δύο κομμάτια, μια συμβολοσειρά που αναπαριστά το “κλειδί” και τα χρήσιμα δεδομένα που αναπαριστούν την “τιμή”, δημιουργώντας έτσι ένα ζευγάρι κλειδιού - τιμής. Αυτή η μέθοδος αποθήκευσης μοιάζει αρκετά με τα hash tables των σχεσιακών βάσεων, όμως εδώ το κλειδί χρησιμοποιείται ως ευρετήριο και έτσι η επεξεργασία των δεδομένων γίνεται πολύ γρηγορότερα. Όλα τα κλειδιά είναι συγκεντρωμένα σε ένα χάρτη και ο χρήστης μπορεί να αναζητήσει δεδομένα σύμφωνα με το κατάλληλο κλειδί.

Λόγω της σχεδίασης αυτής μερικά χαρακτηριστικά όπως ένωση (join) πινάκων ή συναρτήσεις συγκεντρωτικών αποτελεσμάτων (aggregate functions) δεν μπορούν να υλοποιηθούν. Επίσης υπάρχει έλλειψη σχήματος της βάσης δεδομένων και έτσι είναι δύσκολο να κατασκευαστούν προσαρμοσμένες όψεις.

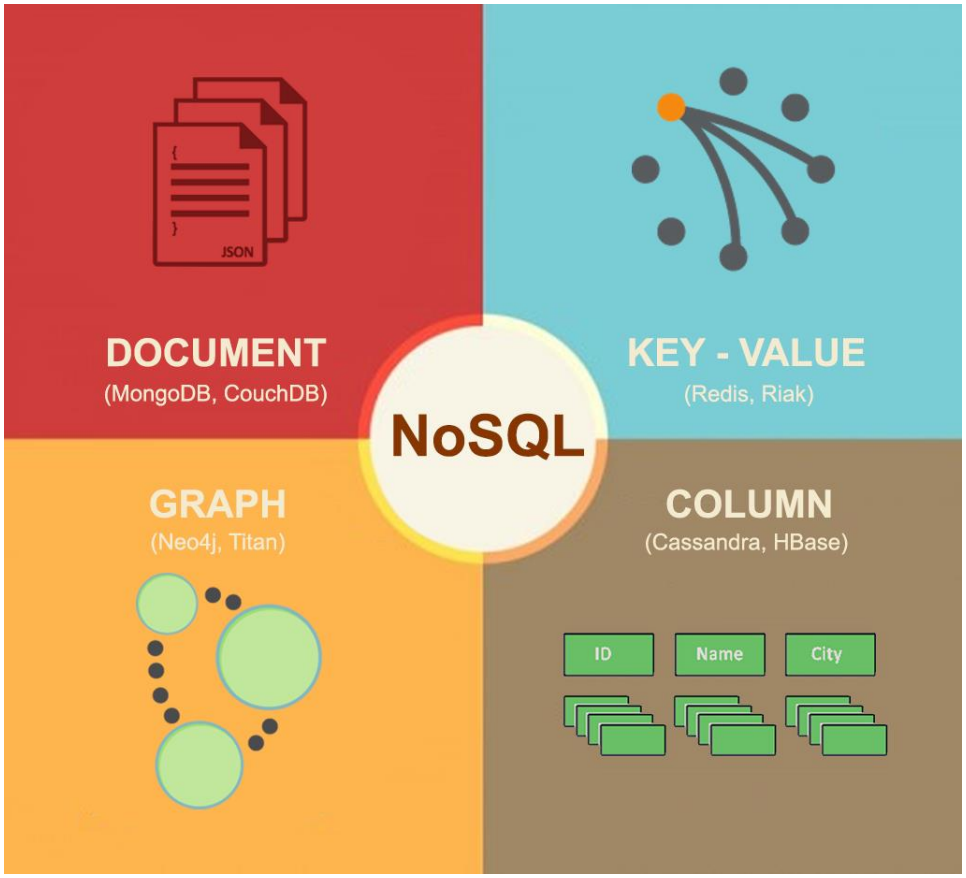
Κατάλληλες εφαρμογές του μοντέλου αυτού είναι όταν χρειάζεται αποθήκευση του session ή του καλαθιού αγορών ενός χρήστη ή σε καταστάσεις όπου χρειαζόμαστε υψηλή διαθεσιμότητα σε συνεχώς μεταβαλλόμενα δεδομένα, όπως οι τιμές των μετοχών.

Μοντέλο Αποθήκευσης κατά Στήλες (Column Family Store)

Τα δεδομένα αποθηκεύονται σε κελία που ομαδοποιούνται σε στήλες αντί για γραμμές και η ανάγνωση-εγγραφή των δεδομένων γίνεται από τις στήλες σε αντίθεση με τις σχεσιακές βάσεις δεδομένων. Οι στήλες ομαδοποιούνται σε οικογένειες στηλών και μπορούν να

περιέχουν απεριόριστο αριθμό στηλών. Κάθε οικογένεια στηλών είναι ανεξάρτητη από τις υπόλοιπες και έτσι η αποθήκευση μπορεί να γίνει κατανεμημένα.

Το μοντέλο αυτό χρησιμοποιείται κυρίως σε εφαρμογές όπου η εξόρυξη (data mining) και οι αναλύσεις (analytics) δεδομένων είναι πολύ σημαντικές.



Σχήμα 3: Μοντέλα NoSQL Βάσεων δεδομένων

Μοντέλο Εγγραφο-κεντρικής Αποθήκευσης (Document Store)

Τα δεδομένα αποθηκεύονται σε μορφή εγγράφων που είναι παρόμοια με τα records στις σχεσιακές βάσεις δεδομένων, όμως παρουσιάζουν μεγαλύτερη ευελιξία αφού δεν έχουν συγκεκριμένο σχήμα. Το μοντέλο αυτό παρουσιάζει αρκετές ομοιότητες με το μοντέλο κλειδιού - τιμής, αφού κλειδί αποτελεί το όνομα του αρχείου και τιμή είναι το περιεχόμενό του (που είναι πιο πολύπλοκο από τις key - value βάσεις δεδομένων)

Οι εγγραφο-κεντρικές βάσεις δεδομένων είναι καλύτερο να χρησιμοποιούνται για εφαρμογές όπου τα δεδομένα χρειάζεται να αποθηκευτούν σε έγγραφα με συγκεκριμένα

χαρακτηριστικά που ορίζονται από το χρήστη. Τα έγγραφα αυτά είναι συνήθως γραμμένα σε μορφές αρχείων XML, JSON, BSON κ.τ.λ.

Μοντέλο Γράφου (Graph Store)

Τα δεδομένα αποθηκεύονται σε μορφή γράφου όπου οι κορυφές αποτελούν αντικείμενα που έχουν συγκεκριμένες ιδιότητες και οι ακμές τις σχέσεις μεταξύ τους. Αφού αποθηκευτούν τα δεδομένα το μόνο που αλλάζει είναι οι σχέσεις μεταξύ τους, κάτι που δίνει ευελιξία στο σχήμα της βάσης. Η μετάβαση από κορυφή σε κορυφή μέσω τον ακμών είναι πολύ γρήγορη και μας δίνει πληροφορίες που δε χρειάζεται να υπολογίζονται κάθε φορά που γίνεται ένα ερώτημα.

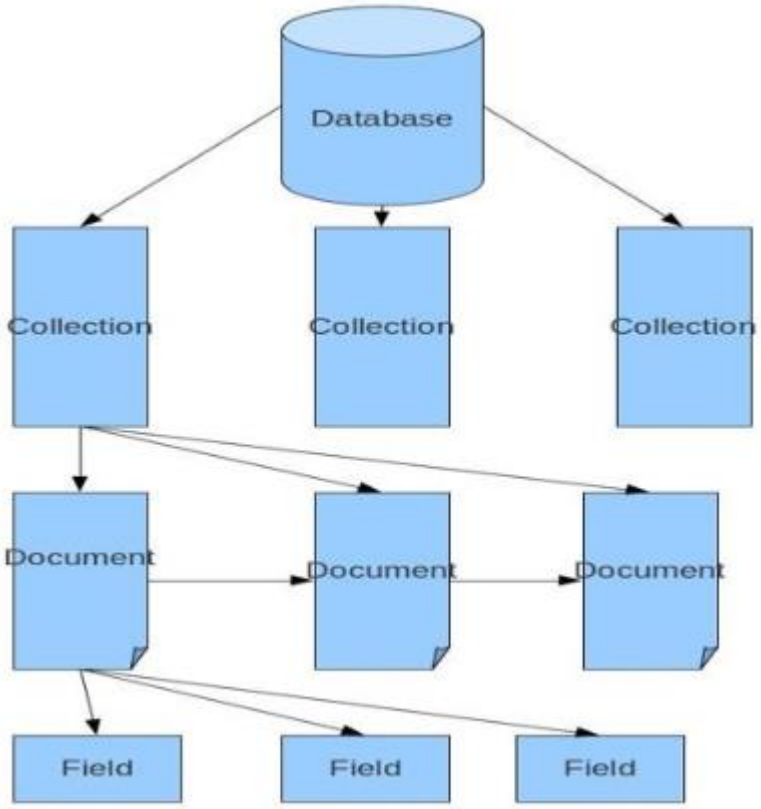
Οι βάσεις δεδομένων δομημένες ως γράφοι χρησιμοποιούνται σε εφαρμογές κοινωνικών δικτύων, διαχείριση cloud και σε περιπτώσεις ελέγχου - ασφάλειας πρόσβασης (security and access control).

Αξίζει να σημειωθεί ότι υπάρχουν και NoSQL βάσεις δεδομένων που συνδυάζουν τα παραπάνω μοντέλα-τύπους ώστε να παρέχουν μεγαλύτερη ευελιξία (Multi-model Databases). Χαρακτηριστικά παραδείγματα τέτοιων βάσεων είναι η MarkLogic που συνδυάζει εγγραφο-κεντρικές βάσεις με γράφους και η OrientDB που αποτελεί ένα υβριδικό συνδυασμό από document, graph και key-value μοντέλα.

2.3.3 MongoDB

Η MongoDB είναι μια ελεύθερη και open-source NoSQL βάση δεδομένων τύπου document store που άρχισε να αναπτύσσεται το 2009 από την εταιρία MongoDB Inc. Για την υλοποίησή της χρησιμοποιήθηκε η αντικειμενοστραφής γλώσσα προγραμματισμού C++ και κύριος στόχος της είναι η αποδοτική και γρήγορη αποθήκευση και επεξεργασία μεγάλου όγκου δεδομένων. Σήμερα χρησιμοποιείται από εκατομμύρια χρήστες και τεράστιους οργανισμούς για την αποθήκευση των δεδομένων τους.

Η δομή της MongoDB μοιάζει αρκετά με τη δομή των σχεσιακών βάσεων δεδομένων. Κάθε βάση δεδομένων (database) περιέχει πολλές διαφορετικές συλλογές (collections). Η έννοια της συλλογής μπορεί να θεωρηθεί παρόμοια με αυτή των πινάκων στις σχεσιακές βάσεις δεδομένων. Πιο βαθιά στην ιεραρχία της MongoDB , οι συλλογές περιέχουν έγγραφα (documents) που μέσα σε αυτά τα δεδομένα αποθηκεύονται ως ένα ζεύγος κλειδιού –τιμής (field – value).



Σχήμα 4: Δομή βάσεων της MongoDB

Το σχήμα της MongoDB είναι δυναμικό και όχι σταθερό όπως στις σχεσιακές βάσεις δεδομένων. Αυτό φαίνεται από το γεγονός ότι έγγραφα που ανήκουν στην ίδια συλλογή μπορεί να έχουν διαφορετικά ζεύγη κλειδιού-τιμής το καθένα και να έχουν εντελώς διαφορετική δομή. Στην πράξη συνήθως μια συλλογή περιέχει έγγραφα παρόμοιου τύπου για ευκολία στη χρήση της βάσης, όμως δεν είναι απαγορευτικό να έχουν διαφορετική δομή αν αυτό είναι επιθυμητό.

Τα έγγραφα αποθηκεύονται στο σκληρό δίσκο σε μορφή BSON ενώ στον χρήστη εμφανίζονται σε μορφή JSON. Σημαντικό είναι το γεγονός ότι ο μόνος περιορισμός που επιβάλλει η MongoDB είναι ότι κάθε έγγραφο πρέπει απαραίτητα να έχει ένα πεδίο με το όνομα _id και τιμή τύπου ObjectID. Το πεδίο αυτό έχει μοναδική τιμή για κάθε εγγραφή και ουσιαστικά παίζει το ρόλο του primary key, διαχωρίζοντας τα δεδομένα μεταξύ τους, ανεξάρτητα από τα πεδία που περιέχουν. Ένα αντιπροσωπευτικό έγγραφο της MongoDB παρουσιάζεται στο παρακάτω σχήμα.

Σχήμα 5: Μορφή ενός εγγράφου στη MongoDB

Μερικά από τα πλεονεκτήματα που μας οδήγησαν να χρησιμοποιήσουμε την MongoDB είναι το πλούσιο documentation που διαθέτει καθώς και αντίστοιχη υποστήριξη από την κοινότητα και τους δημιουργούς της. Αξίζει να σημειωθεί ότι η MongoDB παρέχει οδηγούς (drivers) για πολλές γλώσσες προγραμματισμού όπως C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala και πολλές άλλες. Επίσης πολλές μεγάλες ιστοσελίδες συνεργάζονται και έχουν ενσωματώσει στο API τους λειτουργίες ειδικά για την επικοινωνία με την MongoDB.

Η έκδοση που χρησιμοποιήθηκε είναι η MongoDB Enterprise v3.4, που είναι και η πιο πρόσφατη τη στιγμή της συγγραφής της παρούσας διπλωματικής.

ΚΕΦΑΛΑΙΟ 3

Ανάλυση προβλήματος

Ένας οργανισμός ο οποίος διατηρεί δεδομένα από διάφορους φορείς χρειάζεται να έχει ένα σύστημα που να διαχειρίζεται αποτελεσματικά και με ασφάλεια τα δεδομένα αυτά. Πολλές φορές τα δεδομένα είναι ιδιωτικά και πρέπει να μείνουν κρυφά από τρίτους που προσπαθούν να τα υποκλέψουν ή να τα αλλοιώσουν, ενώ άλλες φορές η ίδια η επιχείρηση παρέχει κάποια δεδομένα ελεύθερα με σκοπό την διευκόλυνσή της ίδιας αλλά και των χρηστών της.

Η δημιουργία ενός ενιαίου συστήματος που θα ικανοποιεί όλες τις διαφορετικές ανάγκες που υπάρχουν για την ασφάλεια των δεδομένων του οργανισμού είναι ο σκοπός της παρούσας διπλωματικής εργασίας. Το σύστημα αυτό αποτελεί συνδυασμό διάφορων τεχνολογιών και τεχνικών στον τομέα της ασφάλειας, και σύμφωνα με τις ανάγκες του εκάστοτε προβλήματος μπορούν να προστεθούν ή να αφαιρεθούν κάποιες λειτουργίες.

Η προστασία ενός συστήματος βάσης δεδομένων απαιτεί αρχικά μια μελέτη των απειλών που είναι πιθανόν αντιμετωπίσει η βάση. Είναι αναγκαίο να υπάρχει ένας μηχανισμός ασφάλειας για κάθε τύπου απειλή που θα αντιμετωπίσει η βάση, καθώς η ασφάλεια του συστήματος είναι τόσο δυνατή όσο είναι και η πιο αδύναμη συνιστώσα του. Ανάλογα με το πόσο σημαντική θεωρείται η κάθε απειλή και τι δεδομένα χρειάζεται να προστατευτούν, τόσο μεγαλύτερη σημασία δίνεται στην κάθε συνιστώσα.

Η επιλογή μιας NoSQL βάσης έγινε για να ωφεληθούμε από τις νέες δυνατότητες που αυτές προσφέρουν όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο. Η βάση αυτή είναι στην περίπτωσή μας η MongoDB. Το σύστημα που υλοποιήθηκε έχει κατασκευαστεί στο στρώμα ακριβώς πάνω από τη βάση και κάτω από το application layer, έτσι ώστε με κάποιες μικρές τροποποιήσεις να μπορεί να εφαρμοστεί πάνω σε οποιαδήποτε βάση δεδομένων.

3.1 Ιατρικά δεδομένα και προστασία

Ένας πολύ σημαντικό κλάδος που χρειάζεται να προστατεύει τα δεδομένα που αποθηκεύει είναι αυτός των υπηρεσιών υγείας. Επειδή τα νοσοκομεία και τα φαρμακεία διαχειρίζονται κάποια πολύ προσωπικά δεδομένα, έχουν διαμορφωθεί κάποια στάνταρτ για τον τρόπο αποθήκευσης και διανομής των δεδομένων.

Το HIPAA (Health Insurance Portability and Accountability Act) είναι το πιο διαδεδομένο στάνταρτ που θεσπίστηκε από της Ηνωμένες Πολιτείες της Αμερικής το 1996[11]. Είναι ουσιαστικά ένα σύνολο κανόνων και ρυθμίσεων για την προστασία ευαίσθητων δεδομένων των ασθενών σε όλες τις υπηρεσίες υγείας. Οποιοσδήποτε οργανισμός αποθηκεύει, διανέμει ή έχει πρόσβαση σε τέτοια δεδομένα οφείλει να ακολουθεί αυτό το στάνταρτ και να εξασφαλίζει τόσο φυσική όσο και διαδικτυακή ασφάλεια.

Το ιδανικότερο σενάριο είναι όλα τα ιατρικά δεδομένα να τα αποθηκεύει ένας οργανισμός και όλοι οι υπόλοιποι να μπορούν να τα επεξεργαστούν και να τα τροποποιήσουν. Ιδιωτικά και δημόσια νοσοκομεία επεξεργάζονται τους φακέλους των ασθενών για να εισάγουν νέα δεδομένα ανάλογα με την κατάσταση του ασθενή, πόσο χρόνο χρειάζεται να νοσηλευτεί, σε ποια κλινική κτλ. Φαρμακεία μπορούν να δουν λεπτομέρειες σχετικά με τον ασθενή, όπως τι φαρμακευτική αγωγή πρέπει να ακολουθήσει, αν έχει αλλεργία σε κάποια συγκεκριμένη ουσία κτλ. Ακόμα και ο ίδιος ο ασθενής - χρήστης μπορεί να έχει πρόσβαση στον προσωπικό του φάκελο και να δει τις πληροφορίες που είναι αποθηκευμένες, να επεξεργαστεί κάποιες από αυτές αλλά και να προσθέσει νέες. Καινούργιες πληροφορίες μπορεί να είναι για παράδειγμα κάποιες μετρήσεις για την πίεση, ζάχαρο ή οποιαδήποτε άλλο στοιχείο που θα λαμβάνονται από κάποια έξυπνη συσκευή, όπως ένα ηλεκτρονικό τσιπ κάτω από το δέρμα ή ένα smartwatch.

Οι NoSQL βάσεις δεδομένων παρέχουν πολλές φορές έτοιμα εργαλεία για τη διευκόλυνση της δημιουργίας εφαρμογών πάνω σε ιατρικά δεδομένα. Τα δεδομένα που αφορούν ένα ασθενή είναι πολλά και σε διαφορετικές μορφές (κείμενο, φωτογραφίες, ήχος, ακτινογραφίες κτλ) και έτσι η ευελιξία του σχήματος των NoSQL βάσεων βοηθάει στη διαχείρισή τους. Κάποιες βάσεις ενσωματώνουν εργαλεία ανάλυσης των ιατρικών δεδομένων

και βγάζουν δημογραφικά αποτελέσματα όσων αφορά την εξάπλωση των ασθενειών, ποιοι πληθυσμοί είναι πιο επιρρεπής, σχέσεις αλληλεπίδρασης μεταξύ φαρμάκων – ασθενειών και τις αλληλεπιδράσεις που μπορεί ακόμα να μην έχουν βρεθεί από τους συμβατικούς τρόπους.

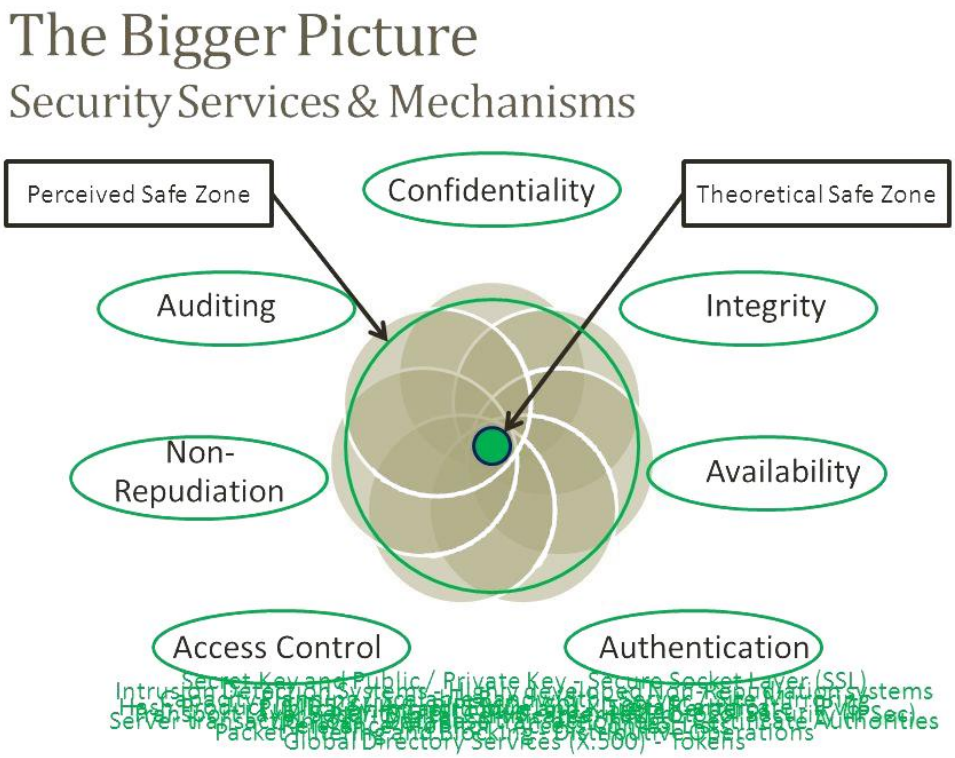
Από όλα αυτά τα θετικά χαρακτηριστικά των NoSQL βάσεων στον τομέας της υγείας εμείς θα εστιάσουμε περισσότερο στην ασφάλεια των δεδομένων χρησιμοποιώντας τα εργαλεία που δίνουν οι βάσεις αυτές σε συνδυασμό με κάποια που είναι ευρέως διαδεδομένα και χρησιμοποιούνται εκτενέστατα.

3.2 Απαιτήσεις και μηχανισμοί συστήματος ασφαλείας

Στη συνέχεια θα αναλύσουμε τις απαιτήσεις ενός συστήματος ασφαλείας πάνω σε ιατρικά δεδομένα. Οι απαιτήσεις αυτές φυσικά μπορούν εύκολα να επεκταθούν και σε άλλου τύπου δεδομένα που χρειάζονται προστασία, καθώς το απόρρητο και η ασφάλεια των ιατρικών δεδομένων θεωρείται από τις σημαντικότερες στη σημερινή εποχή.

Σε γενικές γραμμές, ένα σύστημα προστασίας των δεδομένων χρειάζεται να[9] :

- Περιορίζει την πρόσβαση χρηστών στα δεδομένα μέσω προκαθορισμένων ρυθμίσεων και επιπέδων ασφαλείας.
- Λαμβάνει μέτρα για την προστασία από τυχαία ή κακόβουλη απώλεια, καταστροφή, ζημιά και αποκάλυψη ευαίσθητων προσωπικών δεδομένων.
- Εξασφαλίζει ότι τα δεδομένα θα είναι διαθέσιμα κάθε στιγμή για όποιον χρειαστεί να έχει πρόσβαση σε αυτά.
- Διαχωρίζει τα καθήκοντα κατά την εκτέλεση εφαρμογών ή πρόσβαση των δεδομένων.
- Καταγράφει τη δραστηριότητα των χρηστών, του προσωπικού και των εφαρμογών στα δεδομένα έτσι ώστε να μπορεί με ακρίβεια να βρει την αιτία κάποιου σφάλματος.



Σχήμα 6: Συνδυασμός μηχανισμών και υπηρεσιών ασφαλείας

Κάθε μια από τις παραπάνω απαιτήσεις θα αναπτυχθεί σε βάθος στη συνέχεια, και αφού κατανοηθεί η σημασία και η ανάγκη τους θα αναφερθούμε και με ποια τεχνικά μέσα υλοποιήθηκαν στο σύστημά μας.

3.2.1 Πιστοποίηση χρηστών και εξουσιοδότηση (Authentication - Authorization)

Όταν τα δεδομένα μιας εφαρμογής τα διαχειρίζονται πολλοί χρήστες είναι αναγκαία η επίβλεψη και ο περιορισμός των δυνατοτήτων τους. Ένας διαχειριστής μιας βάσης δεδομένων για παράδειγμα πρέπει να έχει διαφορετικές δυνατότητες από ένα χρήστη που εισάγει ή απλά βλέπει τα δεδομένα της βάσης. Αυτό είναι ακόμα σημαντικότερο αν πρόκειται για δεδομένα που πρέπει να ακολουθούν κάποια στάνταρτ και περιέχουν ευαίσθητες πληροφορίες.

Για την επίτευξη των παραπάνω λειτουργιών εφαρμόζονται κάποιες τεχνικές ελέγχου εισόδου και καταγραφής της δραστηριότητας των χρηστών. Αυτές οι λειτουργίες αποδίδουν ένα επιπλέον στρώμα ασφάλειας των δεδομένων και μπορούν να εφαρμοστούν σε επίπεδο βάσεων δεδομένων αλλά και σε πιο κοντινό στο χρήστη επίπεδο, όπως με τη χρήση κατάλληλου λογισμικού πάνω από τη βάση.

Το access control, όπως ονομάζεται, χωρίζεται σε 2 κατηγορίες. Η μια είναι η πιστοποίηση των χρηστών (authentication), δηλαδή να γνωρίζουμε με ακρίβεια ότι ο χρήστης είναι αυτός που υποστηρίζει ότι είναι για να του δώσουμε τις κατάλληλες δικαιοδοσίες για να πάρει δεδομένα από τη βάση. Εφαρμόζεται δημιουργώντας λογαριασμούς χρηστών προστατευμένους με συνθηματικά που μόνο εκείνοι γνωρίζουν. Οι χρήστες μπορεί να είναι απλοί άνθρωποι, εφαρμογές λογισμικού που χρησιμοποιούν τα δεδομένα, διαχειριστές της βάσης ή και ακόμα λογικοί και φυσικοί κόμβοι στους οποίους τρέχει η βάση δεδομένων. Για να γίνει η πιστοποίηση ενός χρήστη χρειάζεται κάποιος *τρίτος έμπιστος φορέας* για να επιβεβαιώσει την ταυτότητα του χρήστη.

Αφού γίνει η πιστοποίηση του χρήστη και έχει συνδεθεί στο σύστημα, σε κάθε λογαριασμό δίνονται συγκεκριμένες δυνατότητες και δικαιοδοσίες έτσι ώστε να προστατεύονται τα δεδομένα από κακόβουλη ή λανθασμένη διαχείρισή τους (authorization). Ο διαχειριστής του συστήματος βάσης δεδομένων δημιουργεί κάποιες πολιτικές και πρότυπα χρήστη και έτσι κατά τη δημιουργία ενός νέου χρήστη αυτός κατατάσσεται στο κατάλληλο πρότυπο με την κατάλληλη πολιτική. Όταν για παράδειγμα έχουμε χρήστες που μηχανογραφούν δεδομένα στη βάση, οι μόνες δυνατότητες που χρειάζονται είναι για write και όλες οι άλλες λειτουργίες απενεργοποιούνται. Ένας γιατρός χρειάζεται να έχει δικαιώματα και για εισαγωγή νέων δεδομένων, αλλά και για αλλαγή και διαγραφή δεδομένων από το φάκελο ενός ασθενή. Φυσικά υπάρχουν και πιο σύνθετα κριτήρια, όπως για παράδειγμα ένας γιατρός να μην μπορεί να επεξεργαστεί μια πληροφορία που έγραψε ένας άλλος γιατρός (περισσότερες λεπτομέρειες αναφέρονται στη παράγραφο 4.2.4 με τις ψηφιακές υπογραφές). Σε άλλες περιπτώσεις είναι δυνατόν να δίνουμε πρόσβαση σε ένα μόνο υποσύνολο των δεδομένων της βάσης ώστε να προστατευτούν τα ευαίσθητα δεδομένα.

Role	Create	Read	Update	Delete	Index (Maintenance)
Physician	✔	✔	✔		
Billing Associate		✔			
Patient System Administrator		✔		✔	✔
Outsource Employee	✔				✔

Σχήμα 7: Παράδειγμα δικαιωμάτων χρηστών σε μια ιατρική βάση δεδομένων

Εφόσον ελεγχθεί ποιοι χρήστες μπορούν να επεξεργαστούν τα δεδομένα μιας βάσης είναι αναγκαίο να γίνει και καταγραφή των ενεργειών που εκτελούν όσο βρίσκονται εντός του συστήματος. Με τη χρήση ελεγκτικών (auditing) εργαλείων όλες οι δραστηριότητες των χρηστών γράφονται σε log files, είτε πρόκειται για αλλαγές δεδομένων είτε για αλλαγές στη διαμόρφωση της βάσης δεδομένων. Με αυτόν τον τρόπο μπορούν οι διαχειριστές να παρακολουθούν οποιαδήποτε ύποπτη ενέργεια από τους χρήστες.

3.2.2 Εμπιστευτικότητα και ακεραιότητα (Confidentiality - Integrity)

Με τον όρο εμπιστευτικότητα δεδομένων αναφερόμαστε σε προστασία των δεδομένων έτσι ώστε να μη διαρρεύσουν σε τρίτους τα ευαίσθητα δεδομένα της βάσης. Η εμπιστευτικότητα είναι ακόμα περισσότερο σημαντική για τα δεδομένα μια ιατρικής βάσης δεδομένων, καθώς οποιαδήποτε μη επιθυμητή διαρροή μπορεί να προκαλέσει σημαντικά προβλήματα στους χρήστες των υπηρεσιών υγείας.

Για να γίνουν κατανοητές οι συνέπειές από μια τέτοια διαρροή, αρκεί κανείς να σκεφτεί ένα σενάριο που διαρρέει η φυσική ή η ψυχική κατάσταση ενός ασθενή. Για παράδειγμα αν γίνει γνωστό ότι κάποιος πάσχει από μια θανατηφόρα ασθένεια ή ότι έχει ψυχολογικά προβλήματα, τότε τράπεζες, εργοδότες ή ακόμα και οι υπόλοιποι άνθρωποι είναι πιθανό να συμπεριφερθούν διαφορετικά προς τον ασθενή και να μην τον αντιμετωπίσουν με ισότητα. Αυτονόητη είναι η σπουδαιότητα διαρροής αριθμών πιστωτικών καρτών ή λεπτομερειών που αφορούν τις οικονομικές πληροφορίες ενός χρήστη.

Η κρυπτογράφηση αποτελεί το σημαντικότερο εργαλείο για τη διασφάλιση της εμπιστευτικότητας. Η κρυπτογράφηση διασφαλίζει ότι μόνο οι κατάλληλοι άνθρωποι (αυτοί δηλαδή που γνωρίζουν το ιδιωτικό κλειδί της κρυπτογράφησης) μπορούν να διαβάσουν τις πληροφορίες. Είναι ευρέως διαδεδομένη στο σημερινό περιβάλλον και χρησιμοποιείται σχεδόν σε κάθε σημαντικό πρωτόκολλο ασφαλείας για επικοινωνίες μέσω του διαδικτύου, όπως το SSL/TLS.

Ένας άλλος τρόπος που επιτυγχάνεται η εμπιστευτικότητα είναι με τον προηγούμενο τρόπο της πιστοποίησης χρηστών, καθώς μόνο εξουσιοδοτημένοι χρήστες έχουν πρόσβαση σε δεδομένα του οργανισμού και όλες οι ενέργειες που πραγματοποιούν παρακολουθούνται από τα ελεγκτικά εργαλεία.

Η κρυπτογράφηση εκτός από την εμπιστευτικότητα διασφαλίζει και την ακεραιότητα (integrity) των δεδομένων όσων αφορά την αλλοίωση από κακόβουλη επίθεση. Τα δεδομένα έχουν αξία μόνο αν είναι ακριβή και συνεπή, καθώς μια αλλοίωση των δεδομένων μπορεί να προβεί κοστοβόρα. Για παράδειγμα αν αλλάξει (κακόβουλα, από λάθος στην πληκτρολόγηση ή ακόμα και από αστοχία υλικού) η δοσολογία ενός φαρμάκου του ασθενή, τότε μπορεί ακόμα και να του κοστίσει τη ζωή.

Για τη διασφάλιση της ακεραιότητας χρησιμοποιούνται επίσης κάποιοι περιορισμοί για τα δεδομένα που μπορούν να εισαχθούν στη βάση δεδομένων (integrity constraints). Για παράδειγμα θέτουμε ότι το κελί “δοσολογία” δεν μπορεί να έχει ως δεδομένα μια συμβολοσειρά, αλλά πρέπει να είναι πραγματικός αριθμός μεγαλύτερος του 0. Με παρόμοιο τρόπο πρέπει να ελεγχθεί ότι μια σχέση μεταξύ δυο δεδομένων είναι σωστή και δεν έχει αλλοιωθεί (referential integrity).

Στις NoSQL βάσεις συνήθως δεν υπάρχουν πολλοί από τους προηγούμενους περιορισμούς, αφού υπάρχει μεγαλύτερη ελευθερία και ευέλικτο σχήμα, αφήνοντας τον προγραμματιστή να εξασφαλίσει την συνέπεια των δεδομένων με επιπλέον λογισμικό. Τέτοιο λογισμικό μπορεί να είναι ένα απλό σύστημα επικύρωσης της εισόδου των δεδομένων (input

validation), χρήση αθροισμάτων ελέγχου (checksums) ή χρήση ψηφιακών υπογραφών (περισσότερα στη παράγραφο 4.2.4).

Οι αστοχίες υλικού, όπως μια συντριβή του διακομιστή (server crash) ή μια φυσική καταστροφή των εγκαταστάσεων είναι γεγονότα που μπορούν να επηρεάσουν την ακεραιότητα των δεδομένων αλλά δεν θα μελετηθούν στο πλαίσιο της παρούσας διπλωματικής εργασίας.

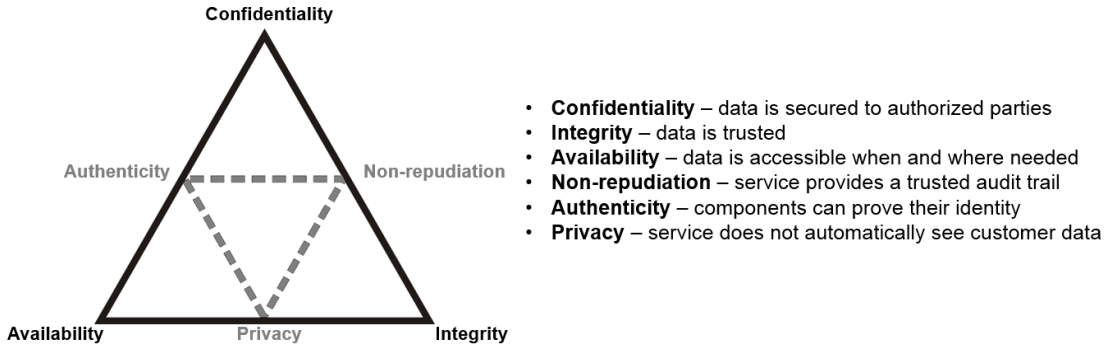
3.2.3 Διαθεσιμότητα (Availability)

Η διαθεσιμότητα ενός συστήματος εγγυάται ότι οι εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση στις πληροφορίες όταν χρειάζεται, δηλαδή κάθε αίτημα που γίνεται προς τη βάση να παίρνει απάντηση. Ένα σύστημα το οποίο δεν είναι ενεργό για μεγάλο χρονικό διάστημα και δεν προσφέρει τις υπηρεσίες για τις οποίες κατασκευάστηκε είναι κατ' επέκταση άχρηστο.

Τα τελευταία χρόνια οι επιθέσεις που έχουν ως στόχο την άρνηση πρόσβασης στις υπηρεσίες ενός συστήματος έχουν αυξηθεί δραματικά. Ο πρωταρχικός ρόλος αυτών των επιθέσεων (Denial of Service (DoS) attacks) είναι να μην επιτρέπεται στους χρήστες μια εφαρμογής να έχουν πρόσβαση στους πόρους της. Τέτοιες διακοπές μπορούν να είναι πολύ δαπανηρές, ιδιαίτερα αν πρόκειται για ιατρικά δεδομένα, καθώς κάθε στιγμή που δε γνωρίζουμε πληροφορίες για έναν ασθενή μπορεί να οδηγήσει σε επιδείνωση της κατάστασής του, όπως για παράδειγμα όταν γίνεται συνεχής ηλεκτρονική παρακολούθηση της υγείας ενός ασθενή με ειδικούς αισθητήρες που συνδέονται στο cloud.

Για να επιλυθεί το ζήτημα της διαθεσιμότητας είναι αναγκαίο να υπάρχουν δευτερεύοντες εξυπηρετητές και αντίγραφα ασφάλειας των δεδομένων σε περισσότερα από ένα υπολογιστικά κέντρα, έτσι ώστε αν γίνει οποιαδήποτε επίθεση ή ακόμα και κάποια φυσική καταστροφή το σύστημα να παραμένει ενεργό.

Μαζί με την εμπιστευτικότητα και την ακεραιότητα, η διαθεσιμότητα ολοκληρώνει την CIA τριάδα[3] που αποτελεί μια θεμελιώδη έννοια στην ασφάλεια συστημάτων βάσεων δεδομένων. Έχοντας προστατεύσει και τις τρεις αυτές πτυχές από ανεπιθύμητες συμπεριφορές έχουμε κάνει ένα σημαντικό βήμα στο σχεδιασμό ενός ασφαλούς συστήματος.



Σχήμα 8: Η τριάδα της εμπιστευτικότητας, ακεραιότητας και διαθεσιμότητας σε συνάρτηση με τα υπόλοιπα χαρακτηριστικά της ασφάλειας των δεδομένων

3.2.4 Μη αποκήρυξη ευθυνών (Non repudiation)

Με τη γενική έννοια, η μη αποκήρυξη συνεπάγεται τη συσχέτιση ενεργειών ή αλλαγών σε ένα μοναδικό άτομο. Στο φυσικό κόσμο η μη αποκήρυξη επιτυγχάνεται με την υπογραφή του ατόμου ότι έχει διαβάσει ή αλλάξει ένα έγγραφο και έτσι είμαστε σίγουροι ότι δεν μπορεί να αποποιηθεί των ευθυνών για την πράξη του, εκτός αν πρόκειται για πλαστογράφηση υπογραφής.

Στον ψηφιακό κόσμο είναι σημαντικό να υπάρχει ένας τρόπος που να δίνει πληροφορίες της προέλευσης ενός δεδομένου χωρίς αμφιβολία (repudiation of origin). Έχουν σχεδιαστεί και μελετηθεί αρκετές μέθοδοι για να πραγματοποιηθεί το προηγούμενο, οι περισσότερες βασισμένες σε κρυπτογραφικές και μαθηματικές τεχνικές. Εφόσον χρησιμοποιείται κρυπτογραφία τις περισσότερες φορές, διασφαλίζεται παράλληλα και η ακεραιότητα του δεδομένου, αφού γνωρίζουμε την προέλευσή του αλλά και ότι δεν έχει αλλοιωθεί.

Σε ιατρικό πλαίσιο, γνωρίζοντας την προέλευση ενός δεδομένου μπορεί να χρησιμεύσει σε πολλές εφαρμογές. Μια γνωμάτευση ασθένειας θα είναι υπογεγραμμένη από έναν γιατρό και έτσι θα είναι γνωστό ποιος την έκανε. Αντίστοιχα, η παροχή φαρμάκων και η συνταγογράφηση μπορεί να γίνει ευκολότερη καθώς θα είναι γνωστό αν ο γιατρός προέρχεται από δημόσιο ή ιδιωτικό φορέα, καθώς και ότι είναι πιστοποιημένα γιατρός έτσι ώστε να μην γίνεται παράνομη διανομή φαρμάκων.

Από νομικής άποψης, η χρήση μαθηματικών και κρυπτογραφικών μεθόδων για τη εξακρίβωση της προέλευσης ενός δεδομένου θεωρείται αδύναμη, καθώς σε μια δίκη λόγω γραφειοκρατικών μηχανισμών δύσκολα μπορεί να αποδειχθεί ότι ένας χρήστης με τη θέλησή του υπέγραψε ένα ψηφιακό δεδομένο, λόγου της παρεμβολής υπολογιστών[1] (σε αντίθεση με τη φυσική υπογραφή που ο χρήστης δε δρα μέσω ενός άλλου αντικειμένου). Αυτό όμως δεν έρχεται σε αντίθεση με το γεγονός ότι υπάρχουν τρόποι για μη αποκήρυξη ευθυνών σε ψηφιακά δεδομένα που είναι 100% ακριβής, χωρίς όμως να έχουν μεγάλη δύναμη σε νομικές αντιπαραθέσεις λόγω σφαλμάτων και κενών ασφάλειας που υπάρχουν στα ηλεκτρονικά υπολογιστικά συστήματα. Σκοπός εξάλλου δεν είναι να καλύψουμε τις νομικές ανάγκες ενός οργανισμού, αλλά τις πραγματικές που αφορούν τη χρηστικότητα, την επεξεργασία και την ασφαλή διακίνηση των δεδομένων του.

ΚΕΦΑΛΑΙΟ 4

Τεχνική ανάλυση προτεινόμενης λύσης

Όπως έχουμε ήδη αναφέρει, σκοπός μας είναι η κατασκευή ενός συστήματος που θα καλύπτει τις τεχνικές προδιαγραφές ασφάλειας που αναφέρθηκαν νωρίτερα έτσι ώστε τα δεδομένα, που στην περίπτωση που εξετάζουμε είναι ιατρικοί φάκελοι ατόμων, να είναι προστατευμένα από απειλές και να ικανοποιούν κάποια στάνταρτ ασφάλειας.

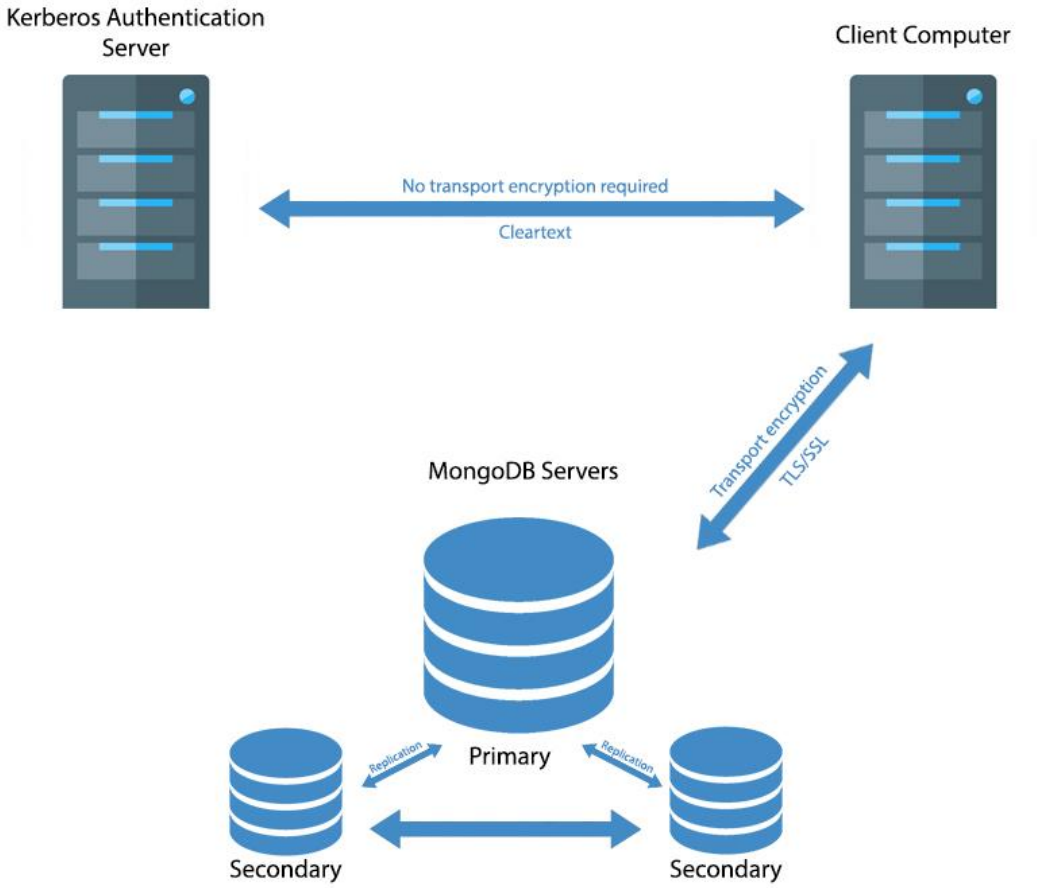
Στη συνέχεια παρουσιάζεται η αρχιτεκτονική του συστήματος που κατασκευάστηκε και αναλύονται τα επιμέρους συνδετικά χαρακτηριστικά του. Όλα μαζί συνεργάζονται έτσι ώστε να προκύψει το τελικό αποτέλεσμα.

4.1 Αρχιτεκτονική

Το σύστημα κατασκευάστηκε πάνω στη NoSQL βάση δεδομένων MongoDB, όμως μπορεί να ενσωματωθεί με μικρές αλλαγές πάνω στις περισσότερες βάσεις δεδομένων, όπως αναφέραμε και σε προηγούμενα κεφάλαια.

Ο έλεγχος της ταυτότητας των χρηστών έγινε με χρήση του πρωτοκόλλου πιστοποίησης Kerberos v5, ενώ το πρόγραμμα που τρέχει στον υπολογιστή του πελάτη είναι γραμμένο σε Java και περιέχει τις κατάλληλες βιβλιοθήκες για την επικοινωνία με την MongoDB αλλά και τους εξυπηρετητές του Κέρβερου.

Για την δοκιμή του συστήματος χρησιμοποιήθηκαν 3 υπολογιστικά συστήματα, ένα που έτρεχε τις υπηρεσίες του Κέρβερου, ένα που ήταν ο κεντρικός εξυπηρετητής της βάσης δεδομένων MongoDB και το τελευταίο ήταν ο δευτερεύον εξυπηρετητής της MongoDB σε περίπτωση που ο κεντρικός είχε βλάβη ή ήταν εκτός λειτουργίας. Για τον υπολογιστή – πελάτη μπορούσε να χρησιμοποιηθεί οποιοσδήποτε προσωπικός υπολογιστής, αρκεί να είχε το πρόγραμμα οδηγό που κατασκευάστηκε με Java και το κατάλληλο λογισμικό για την επικοινωνία με τον Κέρβερο.



Σχήμα 9: Αρχιτεκτονική συστήματος ασφάλειας

4.2 Λεπτομερή ανάλυση επιμέρους στοιχείων

Αναλύονται στη συνέχεια με κάθε λεπτομέρεια τα επιμέρους στοιχεία του συστήματος ασφαλείας που δημιουργήσαμε. Η κάθε υποενότητα συνδέεται με την αντίστοιχη υποενότητα του 3ου κεφαλαίου και αναλύει πως επιτεύχθηκαν οι θεωρητικές μέθοδοι που αναφέρθηκαν στο 3ο κεφάλαιο.

4.2.1 Πιστοποίηση χρηστών με τη χρήση του Kerberos v5

Όπως έχουμε αναφέρει ήδη πολλές φορές, το διαδίκτυο είναι ένας ανασφαλής χώρος και υποκλοπές των δεδομένων που μεταφέρονται είναι συχνό φαινόμενο. Πολλές φορές ο δράστης υποδύεται κάποιον άλλο χρήστη για να υποκλέψει εμπιστευτικές πληροφορίες ή

αποσπά τους κωδικούς του κατά τη μεταφορά τους στο διαδίκτυο και τους χρησιμοποιεί προς όφελός του.

Για την επίλυση των παραπάνω, το MIT σχεδίασε και υλοποίησε τη δεκαετία του ’90 το Kerberos Authentication Protocol για να προστατεύσει τις υπηρεσίες δικτύου του στα πλαίσια του Project Athena[8]. Αφορά ένα μηχανισμό πιστοποίησης μεταξύ ενός χρήστη και ενός διακομιστή ή μεταξύ δυο διαφορετικών διακομιστών. Η σημασία και η δύναμη του πρωτοκόλλου αυτού διαφαίνεται από το γεγονός ότι ακόμα και σήμερα, έπειτα από αρκετές αλλαγές και βελτιώσεις, το πρωτόκολλο Κέρβερος χρησιμοποιείται από μεγάλους οργανισμούς ενσωματωμένο σε συστήματα ασφάλειας.

Ο Κέρβερος λοιπόν σχεδιάστηκε έτσι ώστε να μπορεί με ασφάλεια να πιστοποιηθεί ότι ένας χρήστης είναι όντως αυτός που λέει ότι είναι και δεν είναι κάποιος παρείσακτος. Θεωρούμε ότι οι συναλλαγές μεταξύ των συμμετεχόντων στο πρωτόκολλο του Κέρβερου γίνονται σε ένα ανοικτό δίκτυο όπου τα πακέτα που διαδίδονται μπορούν να υποκλαπούν από οποιονδήποτε. Η ανάλυση των βασικών χαρακτηριστικών του πρωτοκόλλου για την κατανόησή του δίνεται παρακάτω.

Περιγραφή πρωτοκόλλου

Στο πρωτόκολλο ανταλλάσσουν μηνύματα 3 διαφορετικοί υπολογιστές, ο πελάτης, ο εξυπηρετητής του Κέρβερου (αποτελείται από τον Authentication Server (AS) και τον Ticket Granting Server (TGS)) και η υπηρεσία που θέλουμε να συνδεθούμε (εδώ η MongoDB ή αλλιώς Service Server (SS)).

Το πρωτόκολλο αποτελείται από 3 φάσεις και το περιγράφουμε με συντομία στη συνέχεια.

Πρώτη φάση

Ο πελάτης στέλνει στον AS το username του με cleartext και αν υπάρχει την βάση του Κέρβερου, τότε αυτός του στέλνει πίσω :

A) ένα TGS Session key κρυπτογραφημένο με το password του πελάτη που έχει ο AS στην βάση του

B) ένα Ticket-Granting-Ticket (TGT) κρυπτογραφημένο με το TGS Secret key.

Αν έβαλε σωστά το password ο χρήστης τότε ξέρει το TGS session key, ενώ το δεύτερο μήνυμα μόνο ο TGS μπορεί να το αποκρυπτογραφήσει.

Δεύτερη φάση (TGS Exchange)

Ο πελάτης θέλει κάποια υπηρεσία από τον SS, οπότε στέλνει στον TGS δύο μηνύματα:

C) αίτημα με το ID του service που θέλει και το TGT (μήνυμα **B**)

D) έναν authenticator (ID + timestamp) κρυπτογραφημένο με το TGS session key (που μόνο ο πελάτης γνωρίζει από την πρώτη φάση – μήνυμα **A**).

Ο TGS λαμβάνει τα 2 μηνύματα, αποκρυπτογραφεί το TGT από το **C** μήνυμα και βλέπει το TGS Session Key που περιέχει για να μπορεί να αποκρυπτογραφήσει το **D** μήνυμα.

Αν το Sender ID του **C** και **D** είναι ίδιο με αυτό από το authenticator του **D** (αρά τα έστειλε ο ίδιος πελάτης) τότε ο TGS στέλνει πίσω :

E) Client-to-Server ticket κρυπτογραφημένο με το Service secret key του SS

F) Client/Server Session Key κρυπτογραφημένο με το Client/TGS Session Key.

Τρίτη φάση (Client/Server (CS) Exchange)

Ο πελάτης λαμβάνει τα **E** και **F** (δε μπορεί να αποκρυπτογραφήσει το **E**, μόνο ο TGS και ο SS μπορούν). Αποκρυπτογραφεί όμως το **F** και παίρνει το Client/Server Session Key και με αυτό κατασκευάζει έναν authenticator για τον SS. Άρα στέλνει στον SS:

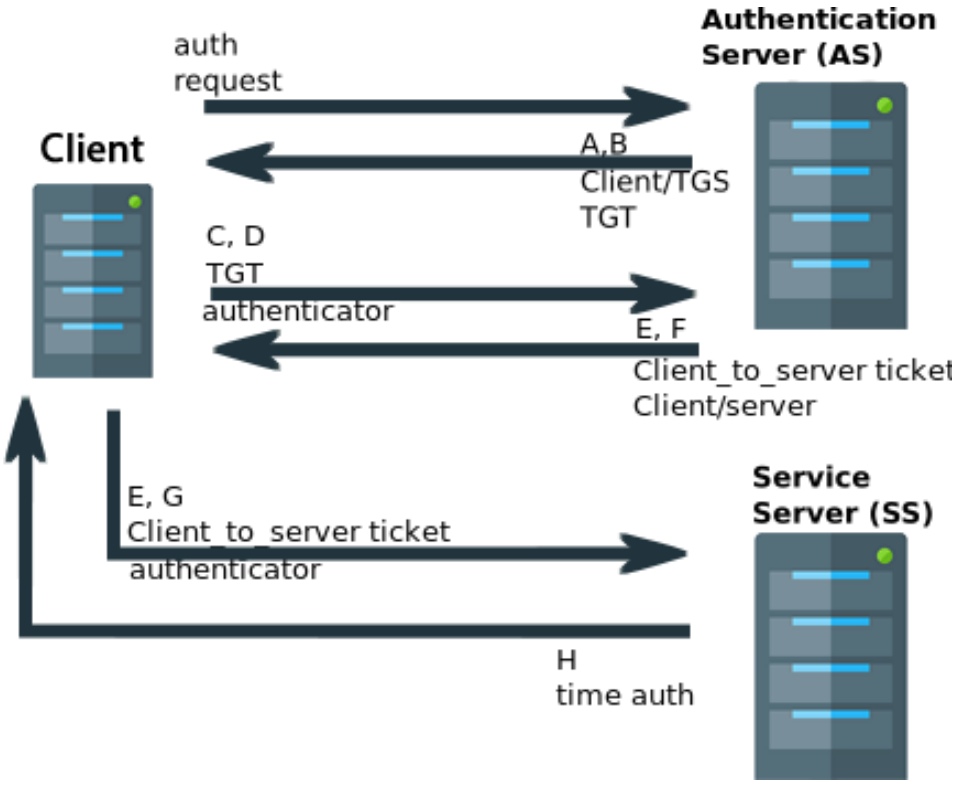
Το **E** μήνυμα αυτούσιο όπως το έλαβε από τον TGS.

G) Τον authenticator (ID+timestamp) κρυπτογραφημένο με το Client/Server Session Key

Ο SS αποκρυπτογραφεί το **E** με το secret key που έχει, βλέπει τα στοιχεία του πελάτη που θέλει κάποιο service, καθώς και το server session key για να αποκρυπτογραφήσει το **G**. Συγκρίνει τα ID του **E** και **G** με αυτό που έχει ο Authenticator και αν είναι ίδια τότε στέλνει πίσω στον πελάτη:

H) το timestamp που είχε ο authenticator κρυπτογραφημένο με το Client/Server Session Key.

Ο πελάτης τέλος αποκρυπτογραφεί το **H** και βλέπει αν το timestamp είναι ίδιο με αυτό που έστειλε νωρίτερα, οπότε γνωρίζει αν ο SS είναι ο αληθινός. Τότε ο SS και ο client αλληλοεμπιστεύονται οπότε ανταλλάσσουν services και δεδομένα.



Σχήμα 10: Ανταλλαγές μηνυμάτων στο πρωτόκολλο του Kerberos

Πλεονεκτήματα και μειονεκτήματα

Τα θετικά χαρακτηρίστηκα του Κέρβερου είναι αρκετά. Ο πελάτης – χρήστης δε χρειάζεται να στείλει τον κωδικό πρόσβασης του μέσω του διαδικτύου για να συνδεθεί στο σύστημα και έτσι υπάρχει μεγαλύτερη ασφάλεια. Φυσικά ο εξυπηρετητής του Κέρβερου πρέπει να ξέρει τον κωδικό του χρήστη και αυτός τον στέλνει μόνο την πρώτη φορά που θα εγγραφεί στο realm του Κέρβερου. Σημαντικό είναι το γεγονός ότι ο χρήστης δε χρειάζεται να πληκτρολογεί τον κωδικό του συνεχώς για να πάρει νέες υπηρεσίες, αλλά η ανταλλαγή πληροφοριών γίνεται με tickets μετά την πρώτη επικοινωνία (single sign-on).

Στα αρνητικά του πρωτοκόλλου βλέπουμε ότι υπάρχει ένα μόνο εξυπηρετητής για τον Κέρβερο και αυτό αποτελεί ένα single point of failure, που σημαίνει ότι αν πάθει κάτι ο κεντρικός εξυπηρετητής τότε δε θα μπορούμε να έχουμε πρόσβαση στις υπηρεσίες που χρειαζόμαστε. Αυτό όμως επιλύεται χρησιμοποιώντας επιπλέον εξυπηρετητές του Κέρβερου

και άλλους εφεδρικούς μηχανισμούς. Επίσης το πρωτόκολλο αυτό βασίζεται στο συγχρονισμό των ρολογιών μεταξύ των διαφορετικών συμμετεχόντων που ανταλλάσσουν μηνύματα και έτσι αν υπάρχουν μεγάλες αποκλίσεις τότε το πρωτόκολλο δεν λειτουργεί. Επειδή όμως χρησιμοποιούνται χρονοσφραγίδες το πρωτόκολλο αποκτά αντοχή σε replay attacks και υποκλοπές μηνυμάτων από τρίτους.

Σύνδεση με τη MongoDB

Ο Κέρβερος όπως αναφέραμε είναι ένα ανεξάρτητο σύστημα που χρησιμοποιείται για την πιστοποίηση των χρηστών. Η MongoDB περιλαμβάνει τις κατάλληλες υποδομές έτσι ώστε οι χρήστες να μπορούν να συνδεθούν μέσω του Κέρβερου και να εξουσιοδοτηθούν μέσα από την ίδια την MongoDB.

Η σύνδεση των δύο προηγουμένων γίνεται με τη δημιουργία κατάλληλων service principals καθώς και user principals στη βάση δεδομένων του Κέρβερου. Η MongoDB χρειάζεται επίσης να διαθέτει ένα keytab αρχείο που περιέχει τα κλειδιά πιστοποίησης του Κέρβερου έτσι ώστε να μπορεί να δεχτεί χρήστες που έχουν τα κατάλληλα tickets. Περισσότερες τεχνικές λεπτομερείς και οι κατάλληλες ρυθμίσεις δίνονται στο σχετικό παράρτημα.

Εφόσον γίνει η πιστοποίηση και ο χρήστης συνδεθεί με τη MongoDB, τότε ανάλογα με το username του η MongoDB του κατανέμει τις αντίστοιχες δικαιοδοσίες. Παράδειγμα δημιουργίας ενός user και των δυνατοτήτων του στη MongoDB φαίνεται παρακάτω:

```
db.createUser(
   {
     user: "NoLee@TELECOM.ECE.NTUA.GR",
     roles: [ { role: "read", db: "HealthCare" } ]
   }
)
```

4.2.2 Εμπιστευτικότητα και ακεραιότητα με χρήση κρυπτογράφησης

Για την επίτευξη της εμπιστευτικότητας και της ακεραιότητας των δεδομένων χρησιμοποιήθηκαν διάφορες τεχνικές κρυπτογράφησης σε διαφορά στρώματα κίνησης των δεδομένων, ακολουθώντας πάντα τους κανόνες ασφαλείας του HIPAA στάνταρντ.

Κρυπτογράφηση κατά τη διάρκεια μεταφοράς (Transport encryption)

Κρυπτογραφώντας τα δεδομένα όταν βρίσκονται ελεύθερα στο διαδίκτυο εξασφαλίζεται ότι το μήνυμα θα είναι αναγνώσιμο μόνο από τον επιδιωκόμενο χρήστη και όχι από οποιονδήποτε άλλο που κατάφερε να υποκλέψει το μήνυμα κατά τη μεταφορά του.

Η μέθοδος που χρησιμοποιήθηκε είναι η ευρέως διαδεδομένη TLS/SSL (Transport Layer Security / Secure Sockets Layer). Χρησιμοποιεί συμμετρική κρυπτογράφηση και συχνά η ταυτότητα των χρηστών επαληθεύεται χρησιμοποιώντας κρυπτογραφία δημοσίου κλειδιού. Η σύνδεση αυτή εξασφαλίζει επίσης ακεραιότητα δεδομένων, αφού κάθε μήνυμα περιλαμβάνει έναν έλεγχο ακεραιότητας (integrity check) χρησιμοποιώντας έναν κωδικό επαλήθευσης μηνύματος για αποφυγή απώλειας ή αλλοίωσης των δεδομένων. Η MongoDB χρησιμοποιεί τις βιβλιοθήκες OpenSSL για το πρωτόκολλο και επιτρέπει μόνο δυνατούς SSL κρυπτογραφικούς αλγόριθμους με ελάχιστο μήκος κλειδιού 128-bit.

Η λεπτομερής ανάλυση του αλγορίθμου γι’αυτό το πρωτόκολλο είναι εκτός τον ορίων της παρούσας διπλωματικής, όμως μπορεί να μελετηθεί σε βάθος από την αντίστοιχη βιβλιογραφία[16].

Κρυπτογράφηση κατά τη διάρκεια στασιμότητας (Encryption at rest)

Με την κρυπτογράφηση των δεδομένων της βάσης στο σκληρό δίσκο εξαλείφονται τα έξοδα διαχείρισης και διευκολύνεται το έργο των διαχειριστών της βάσης. Η κρυπτογράφηση γίνεται με διαφάνεια στο επίπεδο αποθήκευσης των δεδομένων, δηλαδή όλα τα αρχεία είναι πλήρως κρυπτογραφημένα από σκοπιά συστήματος, και μη κρυπτογραφημένα δεδομένα βρίσκονται μόνο στη μνήμη και κατά τη διάρκεια μεταφοράς τους.

Στη MongoDB η κρυπτογράφηση των στάσιμων δεδομένων γίνεται με τον αλγόριθμο AES-256 σε CBC mode χρησιμοποιώντας τις βιβλιοθήκες OpenSSL. Για την κρυπτογράφηση δημιουργούνται ιδιωτικά κλειδιά για κάθε βάση και αυτά τα κλειδιά κρυπτογραφούνται με το master κλειδί, που στη συνέχεια αποθηκεύεται σε έναν εξωτερικό εξυπηρετητή και απαιτεί εξωτερική διαχείριση, συνήθως χρησιμοποιώντας το KMIP πρωτόκολλο[13] (Key Management Interoperability Protocol).

Έλεγχος ενεργειών που γίνονται στη βάση (Auditing)

Με τα ελεγκτικά εργαλεία που παρέχει η MongoDB οι διαχειριστές μπορούν να παρακολουθούν τις ενέργειες των χρηστών όσο είναι συνδεδεμένοι στη βάση δεδομένων. Εκτός από όλες τις δραστηριότητές τους (CRUD operations), γράφονται σε ειδικά αρχεία καταγραφής (log files) και ποιοι χρήστες συνδέθηκαν στη βάση. Παρέχετε έτσι ένα επιπλέον στρώμα προστασίας και ελέγχου των δεδομένων.

Οι τρεις παραπάνω ρυθμίσεις περιέχονται ενσωματωμένες στη MongoDB και αυτό που χρειάστηκε ήταν να τις ρυθμίσουμε κατάλληλα για να ικανοποιούν τις ανάγκες της εφαρμογής μας. Οι ρυθμίσεις αυτές αναφέρονται εκτενέστερα στο σχετικό παράρτημα.

Κρυπτογράφηση ευαίσθητων δεδομένων με δημόσιο κλειδί

Ο χρήστης έχει επίσης τη δυνατότητα να κρυπτογραφεί τα δεδομένα προτού καν τα στείλει στη βάση (και έτσι αποθηκεύονται κρυπτογραφημένα). Φυσικά χάνονται κάποιες λειτουργίες της βάσης, όπως η δυνατότητα για εξόρυξη δεδομένων ή προβολή στατιστικών και αναλυτικών στοιχείων και γι’αυτό δεν συνίσταται αυτή η πρακτική πάρα μόνο για πολύ ευαίσθητα δεδομένα.

Η τεχνική που υλοποιήθηκε για την κρυπτογράφηση των ευαίσθητων δεδομένων ακολουθεί την κρυπτογράφηση δημοσίου κλειδιού (public key cryptography) και συγκεκριμένα τον αλγόριθμο RSA[4]. Κατά την εισαγωγή ενός νέου χρήστη δημιουργείται ένα ζεύγος ιδιωτικού και δημοσίου κλειδιού. Το δημόσιο κλειδί αποθηκεύεται στην εγγραφή του χρήστη και μπορεί οποιοσδήποτε να το χρησιμοποιήσει για να καταχωρήσει στον χρήστη κάποια ευαίσθητη πληροφορία κρυπτογραφημένη με το δημόσιο αυτό κλειδί.

Το ιδιωτικό κλειδί *πρέπει* να το αποθηκεύσει με ασφάλεια ο χρήστης που τον αφορά. Αυτό μπορεί να γίνει με χρήση μιας Smart Card που περιέχει το ιδιωτικό κλειδί και μόνο με αυτή μπορούν να διαβαστούν τα δεδομένα. Παρόμοιο παράδειγμα αποτελούν οι πιστωτικές κάρτες. Είναι σημαντικό το δημόσιο κλειδί να είναι ψηφιακά υπογεγραμμένο από τον ίδιο τον χρήστη ή κάποιον έμπιστο χρήστη (όπως για παράδειγμα κάποιο διαχειριστή του συστήματος) ώστε να αποφευχθούν τυχόν αλλαγές στο δημόσιο κλειδί από κάποιον κακόβουλο με σκοπό την υποκλοπή ευαίσθητων δεδομένων.

Μέσω τις εφαρμογής που κατασκευάστηκε γίνεται επίσης και έλεγχος ορθότητας των εισαγόμενων δεδομένων (input validation) ώστε να διασφαλίζεται ακόμα περισσότερο η ακεραιότητα και η ορθότητα των δεδομένων.

4.2.3 Διαθεσιμότητα μέσω Replication

Για την επίτευξη της διαθεσιμότητας αλλά και για μεγαλύτερη συνέπεια, ασφάλεια δεδομένων και ανοχή σε σφάλματα χρησιμοποιήθηκε η μέθοδος του replication. Σε αντίθεση με τα αντίγραφα ασφαλείας που αποθηκεύονται εκτός δικτύου, όταν υπάρχουν κατανεμημένοι κόμβοι αποθηκεύονται στον καθένα αντίγραφα των δεδομένων από άλλους έτσι ώστε αν υπάρξει βλάβη σε κάποιους από αυτούς να έχουμε ακόμα πρόσβαση στα δεδομένα. Για παράδειγμα, αν ένα υπολογιστικό κέντρο σταματήσει να λειτουργεί λόγω κάποιας ηλεκτρολογικής βλάβης, τα δεδομένα θα είναι διαθέσιμα από άλλους κόμβους και έτσι υπάρχει συνεχή διαθεσιμότητα.

Στη MongoDB όταν ένας master (ή primary όπως αναφέρεται) κόμβος πάθει κάποιο σφάλμα, τότε τα ερωτήματα εξυπηρετούνται από τους κόμβους που έχουν τα replicas και γίνεται εκλογή ενός νέου primary από αυτούς τους κόμβους. Έτσι επιτυγχάνεται και replication αλλά και availability καθώς εξυπηρετούνται ερωτήματα από πολλούς κόμβους ταυτόχρονα.

Ο primary κόμβος δέχεται όλα τα write αιτήματα από τους χρήστες και καταγράφει όλες τις αλλαγές στα δεδομένα σε συγκεκριμένα log files. Οι δευτερεύοντες κόμβοι αντιγράφουν ασύγχρονα τις αλλαγές αυτές χρησιμοποιώντας τα log files.

Σε αντίθεση με τα write αιτήματα, οι δευτερεύοντες servers μπορούν να εξυπηρετούν read αιτήματα, πετυχαίνοντας ταχύτερη εξυπηρέτηση των χρηστών σε περιπτώσεις όπου υπάρχει μεγάλη ζήτηση δεδομένων και δεν μπορεί να ανταπεξέλθει ο primary εξυπηρετητής. Ανάλογα με τον τρόπο που γίνεται το replication υπάρχει η πιθανότητα τα δεδομένα που διαβάζονται από τους δευτερεύοντες κόμβους να μην είναι τα πιο πρόσφατα. Αν αυτό επηρεάζει αρνητικά την ορθή λειτουργία του συστήματος είναι δυνατόν να θυσιαστεί ταχύτητα έτσι ώστε όλα τα δεδομένα που διαβάζονται να έχουν πάρει την τελική τους τιμή.

Φυσικά, σύμφωνα με το CAP θεώρημα[17], δεν είναι δυνατόν να πετύχουμε και availability και consistency δεδομένων όταν υπάρχει διαχωρισμός μεταξύ των κόμβων του δικτύου (partitioning). Έτσι, ανάλογα με τα δεδομένα και τι σκοπό εξυπηρετούν, επιλέγεται είτε διαθεσιμότητα (και έτσι η τιμή του δεδομένου που διαβάσαμε μπορεί να μην είναι η πιο πρόσφατη) είτε consistency (που σημαίνει ότι δε θα παρέχουμε δεδομένα μέχρι να επισκευαστεί η βλάβη που προκάλεσε το partitioning).

4.2.4 Μη αποκήρυξη ευθυνών (non-repudiation) χρησιμοποιώντας ψηφιακές υπογραφές

Για να μπορούμε να γνωρίζουμε με ακρίβεια ποιος τροποποίησε ένα αρχείο στη βάση καθώς και για να γνωρίζουμε ότι δεν αλλοιώθηκε κακόβουλα, χρησιμοποιήσαμε την τεχνική των ψηφιακών υπογραφών [18].

Αρχικά, ο κάθε χρήστης δημιουργεί ένα ζεύγος κλειδιών, ένα δημόσιο και ένα ιδιωτικό. Το δημόσιο κλειδί το αποθηκεύουμε στη βάση δεδομένων μαζί με το αντίστοιχο όνομα χρήστη και με αυτό θα γίνεται στη συνέχεια η επικύρωση της υπογραφής στα δεδομένα. Είναι ευθύνη του χρήστη να αποθηκεύσει σε ασφαλές μέρος το ιδιωτικό του κλειδί και να μην το

χάσει, καθώς χωρίς αυτό δε θα μπορεί να εισάγει νέα δεδομένα ή να επεξεργαστεί υπάρχοντα από τη βάση δεδομένων. Το κάθε κλειδί έχει μήκος 1024-bit και δημιουργήθηκε με τον DSA αλγόριθμο[7], χρησιμοποιώντας μια συνάρτηση παραγωγής τυχαίων αριθμών από τις βιβλιοθήκες της Java.

Αφού ο χρήστης δημιουργήσει το ιδιωτικό και δημόσιο κλειδί του μπορεί πια να εισάγει δεδομένα στη βάση και να τα επεξεργάζεται. Σε κάθε εισαγωγή ή αλλαγή ενός δεδομένου, εκτός από το ίδιο το δεδομένο ο χρήστης στέλνει στη βάση και τη ψηφιακή υπογραφή του γι' αυτό το δεδομένο, που αποτελείται από το ίδιο το δεδομένο και από μια ψευδοτυχαία συμβολοσειρά προσαρτημένη στο δεδομένο. Η διαδικασία γίνεται ευκολότερα κατανοητή με τη βοήθεια του επόμενου παραδείγματος.

ΑΦΜ:	123456789
Όνομα:	Εμμανούηλ
Επίθετο:	Πενταράκης
Ηλικία:	24
Ασθένεια:	-

Έστω ότι έχουμε καταχωρήσει στη βάση το χρήστη "Πενταράκη Εμμανουήλ" με ΑΦΜ 123456789 (μοναδικός αριθμός για κάθε χρήστη εγγεγραμμένο στη βάση). Αν ο χρήστης "ΓΝΑ_Doctor1" θέλει να εισάγει μια νέα ασθένεια στον προηγούμενο χρήστη, τότε θα γίνει η παρακάτω διαδικασία :

Για την εισαγωγή της ασθένειας "Γρίπη", εισάγεται αρχικά κανονικά η ασθένεια στο κατάλληλο πεδίο "Ασθένεια". Για την κατασκευή της ψηφιακής υπογραφής για το δεδομένο "Γρίπη" συγκολλείται αρχικά ο ΑΦΜ στο δεδομένο ("Γρίπη:123456789"), μετά εφαρμόζεται η hash function SHA1 και τέλος κρυπτογραφείται με το ιδιωτικό κλειδί του χρήστη "ΓΝΑ_Doctor1".

Εισάγεται έτσι στη βάση δεδομένων και το πεδίο:

_Ασθένεια: { [BinData:(mH7dm…8gHx), "ΓΝΑ_Doctor1"]}

Στο παραπάνω πεδίο παρατηρούμε 2 τιμές. Η πρώτη είναι η ψηφιακή υπογραφή του δεδομένου σε δυαδική μορφή και η δεύτερη είναι ο χρήστης ο οποίος υπέγραψε το δεδομένο.

Ο φάκελος του χρήστη μετά την εισαγωγή της ασθένειας θα μοιάζει με τον παρακάτω:

ΑΦΜ:	123456789
Όνομα:	Εμμανούηλ
Επίθετο:	Πενταράκης
Ηλικία:	24
Ασθένεια:	Γρίπη
_Ασθένεια:	[BinData:(mH7dm..8gHx), "ΓΝΑ_Doctor1"]

Για να επαληθεύσουμε την ψηφιακή υπογραφή ακολουθείται η αντίστροφη διαδικασία, χρησιμοποιώντας για την αποκρυπτογράφηση το δημόσιο κλειδί του χρήστη που φαίνεται ότι υπέγραψε το δεδομένο. Το δημόσιο κλειδί βρίσκεται όπως αναφέραμε και προηγουμένως στη βάση δεδομένων και είναι προσβάσιμο από όλους τους χρήστες.

Η χρήση του ΑΦΜ πριν την εφαρμογή της hash function γίνεται με σκοπό να μην μπορεί κάποιος κακόβουλος χρήστης να αντιγράψει την ασθένεια μαζί με την ψηφιακή υπογραφή στον φάκελο κάποιου άλλου ασθενή. Αν γίνει κάτι τέτοιο, τότε η επαλήθευση της ψηφιακής υπογραφής θα αποτύχει, αφού θα χρησιμοποιηθεί διαφορετικός ΑΦΜ κατά την κατασκευή της συμβολοσειράς. Προστατεύουμε έτσι την ακεραιότητα και την εγκυρότητα των δεδομένων.

Με τη χρήση των ψηφιακών υπογραφών επιτυγχάνεται και η μη αποποίηση ευθυνών αλλά και η ακεραιότητα των δεδομένων, αφού οποιαδήποτε αλλαγή στο δεδομένων ή στην ψηφιακή υπογραφή θα οδηγήσει σε αποτυχία επαλήθευσης και έτσι το δεδομένο δε θα θεωρείται έγκυρο. Τη μη αποποίηση ευθυνών τη χρησιμοποιούμε για φαρμακευτικούς και νοσηλευτικούς σκοπούς κυρίως, αφού γνωρίζουμε με ακρίβεια ποιος έκανε μια διάγνωση ή χορήγησε το αντίστοιχο φάρμακο.

ΚΕΦΑΛΑΙΟ 5

Υλοποίηση κρυπτοδιεπαφής της εφαρμογής (API)

Οι τεχνικές και οι μέθοδοι που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής αναφέρονται εκτενέστερα παρακάτω. Επίσης περιγράφεται ένα σενάριο χρήσης της εφαρμογής καθώς και τα αντίστοιχα διαγράμματα όπου κρίθηκε αναγκαίο.

5.1 Απαιτήσεις λογισμικού

Στη συνέχεια αναφέρονται οι λειτουργικές απαιτήσεις της εφαρμογής που υλοποιήθηκε, καθώς και κάποιες σχεδιαστικές αποφάσεις και υποθέσεις που έγιναν ώστε καλυφθούν όλες οι πτυχές του προβλήματος.

5.1.1 Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις της εφαρμογής περιγράφουν τις δυνατότητες ή υπηρεσίες του συστήματος. Εξαρτώνται από τον τύπο του λογισμικού που χρησιμοποιείται αλλά και από τους αναμενόμενους χρήστες του λογισμικού.

Ένας χρήστης του συστήματος πρέπει να έχει τη δυνατότητα:

- Να κάνει εγγραφή στο σύστημα πιστοποίησης του Κέρβερου με το username και το password του.
- Να μπορεί να συνδεθεί μέσω της εφαρμογής στη βάση δεδομένων του συστήματος χρησιμοποιώντας τα προηγούμενα διαπιστευτήρια
- Να μπορεί κατά την πρώτη σύνδεση του στο σύστημα ή κατά την εγγραφή του να δημιουργήσει ένα ζεύγος ιδιωτικού – δημοσίου κλειδιού που θα το χρησιμοποιήσει για να υπογράφει ψηφιακά τα δεδομένα πριν τα στείλει στη βάση.
- Να μπορεί να εισάγει πληροφορίες ενός νέου ασθενή (ΑΦΜ, όνομα, επώνυμο, ηλικία, ασθένεια, κτλ)
- Να μπορεί να δει τις πληροφορίες ενός εγγεγραμμένου ασθενή, να τις επεξεργάζεται και να τις στέλνει πίσω στη βάση.

- Να μπορεί να κρυπτογραφεί οποιαδήποτε κρίσιμη πληροφορία για τον ασθενή με τη χρήση του δημοσίου κλειδιού του ασθενή πριν την αποστείλει στη βάση.
- Να μπορεί να αποκρυπτογραφεί τα κρυπτογραφημένα δεδομένα με χρήση της Έξυπνης Κάρτας του ασθενή, που μόνο αυτή περιέχει το ιδιωτικό κλειδί αποκρυπτογράφησης.
- Να μπορεί να επαληθεύσει την ψηφιακή υπογραφή σε κάθε δεδομένο, να δει ποιος το έχει υπογράψει και αν όντως το δεδομένο στο οποίο αναφέρεται δεν έχει αλλοιωθεί.

5.1.2 Κύριες σχεδιαστικές αποφάσεις και υποθέσεις

Η εφαρμογή δημιουργήθηκε ως ένα εκτελέσιμο αρχείο για προσωπικούς υπολογιστές με ένα απλό γραφικό περιβάλλον. Σε μεγαλύτερη έκταση οι μέθοδοι που υλοποιήθηκαν μπορούν πολύ εύκολα να μετατραπούν σε ένα API όπου οι προγραμματιστές θα χρησιμοποιήσουν για να κατασκευάσουν μια μεγαλύτερη εφαρμογή με περισσότερες δυνατότητες. Τα κύρια χαρακτηριστικά και οι αλγόριθμοι όμως παραμένουν τα ίδια.

Για τις ψηφιακές υπογραφές είναι αναγκαίο να υπάρχει μια υπηρεσία που θα αποθηκεύει τα δημόσια κλειδιά των χρηστών. Η υπηρεσία αυτή είναι ένας τρίτος έμπιστος οργανισμός (Trusted Certificate Authority) και χρησιμοποιείται για να αποφευχθούν οι επιθέσεις man-in-the-middle, όμως για λόγους απλότητας τα αποθηκεύουμε στη βάση δεδομένων που χρησιμοποιούμε. Παρόμοιο περιστατικό αποτελεί και η αποθήκευση του master key που αφορά το encryption at rest που παρέχει η MongoDB και χρειάζεται εξωτερική αποθήκευση με βάση το KMIP[13] πρωτόκολλο.

Για τη σωστή λειτουργεία του πρωτοκόλλου του Κέρβερου πρέπει να θεωρήσουμε κάποιες παραδοχές. Αρχικά πρέπει τα ρολόγια του εξυπηρετητή και του πελάτη να είναι συγχρονισμένα για να λειτουργήσει σωστά το πρωτόκολλο. Επίσης θεωρούμε ότι ο κύριος εξυπηρετητής του Κέρβερου ή ένας δευτερεύον είναι πάντα σε λειτουργία, διαφορετικά δε θα μπορούν οι χρήστες να συνδεθούν στο σύστημα.

Σε μερικά σημεία αναφέρθηκε η χρήση μιας Έξυπνης Κάρτας για αποθήκευση των ευαίσθητων πληροφοριών του χρήστη. Η υλοποίηση μιας τέτοιας τεχνολογίας παρέμεινε σε θεωρητικό επίπεδο και δεν αναλύθηκε εκτενώς, αλλά παρουσιάστηκε ως ένας ασφαλής

τρόπος αποθήκευσης των ευαίσθητων δεδομένων, όπως το ιδιωτικό κλειδί κρυπτογράφησης.

5.2 Ανάλυση της κρυπτοδιεπαφής (Crypt- MongoDB API)

Η εφαρμογή που υλοποιήσαμε αποτελείται από 3 μεγάλες κλάσεις που χρησιμοποιούνται για να ομαδοποιήσουν τις μεθόδους που χρησιμοποιεί το σύστημα για να επικοινωνεί με τον χρήστη και τις υπόλοιπες υπηρεσίες (Κέρβερο, MongoDB) όπως αναφέραμε σε προηγούμενες ενότητες.

5.2.1 Μέθοδοι και UML διαγράμματα της εφαρμογής

Οι μέθοδοι αυτοί μπορούν να λειτουργήσουν και αυτόνομα ως μέρος ενός API μιας web εφαρμογής και να χρησιμοποιηθούν από τους προγραμματιστές για την κατασκευή ενός μεγαλύτερου συστήματος, καθώς και μπορούν να επεκταθούν εύκολα με επιπλέον λειτουργίες και δυνατότητες.

Σχήμα 11: API - Class diagram της εφαρμογής

Για καθεμιά από τις παραπάνω κλάσεις και μεθόδους αναφέρουμε τη λειτουργία της και τα χαρακτηριστικά της.

*public class **MainGUI***

Περιέχει όλο το γραφικό περιβάλλον της εφαρμογής και καλεί κατάλληλα όλες τις υπόλοιπες μεθόδους ανάλογα με τις ενέργειες του χρήστη πάνω στα κουμπιά, στα πλαίσια κειμένου κτλ.

*public class **KeyManager***

Περιέχει όλες τις μεθόδους που αφορούν τη μεταχείριση των δημόσιων και ιδιωτικών κλειδιών, τόσο για τις ψηφιακές υπογραφές, όσο και για την κρυπτογράφηση δημοσίου κλειδιού. Τα κλειδιά αποθηκεύονται τοπικά στον σκληρό δίσκο και αν χρειάζεται να αποσταλούν στη βάση δεδομένων τότε χρησιμοποιούνται οι μέθοδοι της Database κλάσης.

createKeyPairs*()*

returns	KeyPair

Δημιουργεί ένα ζεύγος από δημόσιο - ιδιωτικό κλειδί για να χρησιμοποιηθεί από τον χρήστη. Το ζεύγος των κλειδιών αυτών μπορεί να είναι για ψηφιακές υπογραφές (αλγόριθμος DSA) ή για κρυπτογράφηση δημοσίου κλειδιού (αλγόριθμος RSA) και η διαδικασία είναι ελάχιστα διαφορετική ανάλογα με το είδος των κλειδιών που χρειάζεται να παραχθούν.

saveKeysToFile*(KeyPair kp, String uName)*

kp	Το ζεύγος ιδιωτικού – δημοσίου κλειδιού για αποθήκευση
uName	Το όνομα χρήστη στο οποίο αντιστοιχούν τα κλειδιά
returns	void

Αποθηκεύει το ζεύγος τον κλειδιών που παράχθηκαν από την προηγούμενη μέθοδο και τα αποθηκεύει *τοπικά*. Το δημόσιο κλειδί αποθηκεύεται και στη βάση δεδομένων (βλέπε παρακάτω), αλλά το ιδιωτικό κλειδί πρέπει να το αποθηκεύσει με ασφάλεια ο χρήστης που τον αφορά, είτε σε Smart Card είτε σε ασφαλές μέρος εκτός δικτύου.

getPublicKey*(String uName),* ***getPrivateKey****(String uName);*

uName	Το όνομα χρήστη στο οποίο αντιστοιχούν τα κλειδιά
returns	Το ιδιωτικό ή δημόσιο κλειδί του χρήστη

Επιστρέφει τα δημόσια και ιδιωτικά κλειδιά από τον *τοπικό* χώρο αποθήκευσής τους. Τα κλειδιά αφορούν τον αλγόριθμο DSA ή RSA, ανάλογα με το τι ζητάει ο χρήστης.

encryptRSA*(PublicKey pk, String data);*

pk	Το δημόσιο κλειδί με το οποίο θα γίνει η κρυπτογράφηση
data	Τα δεδομένα προς κρυπτογράφηση
returns	byte[]

Δέχεται ως είσοδο το δημόσιο κλειδί της κρυπτογράφησης και το δεδομένο που πρέπει να κρυπτογραφηθεί και επιστρέφει ως αποτέλεσμα το κρυπτογραφημένο δεδομένο σε μορφή ενός πίνακα από bytes. Η μέθοδος αυτή είναι βοηθητική και αυτόνομη και χρησιμοποιείται μόνο για τον αλγόριθμο κρυπτογράφησης δημοσίου κλειδιού RSA.

decryptRSA*(PrivateKey pk, byte[] encryptedData);*

pk	Το ιδιωτικό κλειδί με το οποίο θα γίνει η κρυπτογράφηση
encryptedData	Τα δεδομένα προς αποκρυπτογράφηση
returns	byte[]

Δέχεται ως είσοδο το ιδιωτικό κλειδί της κρυπτογράφησης και το κρυπτογραφημένο δεδομένο και αν το ιδιωτικό κλειδί είναι το κατάλληλο για την κρυπτογράφηση, τότε επιστρέφει ως αποτέλεσμα το ακρυπτογράφητο δεδομένο. Αν το ιδιωτικό κλειδί δεν είναι το σωστό, τότε η διαδικασία τερματίζεται με σφάλμα. Η μέθοδος αυτή χρησιμοποιείται μόνο για τον αλγόριθμο κρυπτογράφησης δημοσίου κλειδιού RSA.

*public class **Database***

Περιλαμβάνει όλες τις λειτουργίες που αφορούν τη σύνδεση και την αποστολή δεδομένων από και προς τη βάση δεδομένων MongoDB. Περιέχει το βασικότερο κομμάτι των μεθόδων που υλοποιήθηκαν, καθώς ενσωματώνει όλες τις τεχνολογίες που αναφέρθηκαν σε προηγούμενα κεφάλαια και τις παρέχει με κομψό και αποτελεσματικό τρόπο στον χρήστη.

***connect**(String uName)*

uName	Το όνομα χρήστη που θα συνδεθεί στη βάση δεδομένων
returns	MongoClient

Η μέθοδος αυτή χρησιμοποιείται για να συνδεθεί ο χρήστης με τις υπηρεσίες της MongoDB. Προϋποθέτει ότι ο χρήστης έχει λάβει από πιο πριν τα απαραίτητα πιστοποιητικά από τις υπηρεσίες του Κέρβερου. Αν περάσει τον έλεγχο πιστοποίησης και από τη MongoDB, τότε γίνεται η σύνδεση και επιστρέφεται στον χρήση ένα αντικείμενο MongoClient που χρησιμοποιείται για οποιαδήποτε άλλη επικοινωνία με τη βάση δεδομένων.

***storePK**(MongoClient mClient, PublicKey pk)*

mClient	Η ενεργή σύνδεση με την MongoDB
pk	Το δημόσιο κλειδί που θα αποθηκευτεί στη βάση δεδομένων
returns	void

Αποθηκεύει το δημόσιο κλειδί (είτε για τις ψηφιακές υπογραφές, είτε για την κρυπτογράφηση δημοσίου κλειδιού) στη κατάλληλη βάση δεδομένων και στην κατάλληλη συλλογή (collection).

***getPK**(MongoClient mClient, String uName)*

mClient	Η ενεργή σύνδεση με την MongoDB
uName	Το όνομα χρήστη του οποίου θέλουμε το δημόσιο κλειδί από τη βάση
returns	PublicKey

Ανακτά από τη βάση δεδομένων το δημόσιο κλειδί για τον αντίστοιχο αλγόριθμο κρυπτογράφησης. Το δημόσιο αυτό κλειδί χρησιμοποιείται είτε για επικύρωση της ψηφιακής υπογραφής είτε για κρυπτογράφηση ευαίσθητων πληροφοριών.

myInsert*(MongoClient mClient, String AFM, PrivateKey pk);*

myUpdate*(MongoClient mClient, String AFM, Document data, PrivateKey pk)*

mClient	Η ενεργή σύνδεση με την MongoDB
AFM	Ο ΑΦΜ που είναι μοναδικός για κάθε χρήστη, είτε για εισαγωγή ενός νέου, είτε για update ενός υπάρχοντος
data	Τα δεδομένα (field – value) που θα προστεθούν στο φάκελο του ασθενή
pk	Το ιδιωτικό κλειδί για να γίνει η διαδικασία της ψηφιακής υπογραφής
returns	void

Οι δύο αυτές μέθοδοι αντικαθιστούν τις βασικές προκαθορισμένες μεθόδους που παρέχει η MongoDB σε βιβλιοθήκες για την Java. Με αυτές τις νέες μεθόδους γίνεται η εισαγωγή ενός νέου δεδομένου ή η επεξεργασία ενός υπάρχοντος και παράλληλα γίνεται και η διαδικασία της ψηφιακής υπογραφής και αποστέλλονται στη βάση η υπογραφή μαζί με το αρχικό δεδομένο. Είναι πολύ σημαντικές καθώς δε γίνεται να εισάγουμε στη βάση δεδομένα χωρίς πρώτα να υπογραφούν και έτσι διασφαλίζεται η εγκυρότητα, η εμπιστευτικότητα και η ακεραιότητα των δεδομένων.

signData*(String data, PrivateKey pk, String salt)*

data	Τα δεδομένα (values) που θέλουμε να υπογραφτούν
pk	Το ιδιωτικό κλειδί για να γίνει η διαδικασία της ψηφιακής υπογραφής
salt	Η συμβολοσειρά που θα προστεθεί στα δεδομένα πριν γίνει το hashing
returns	byte[]

Αυτή η μέθοδος είναι συμπληρωματική και καλείται κυρίως από τις δύο προηγούμενες, myInsert() και myUpdate(), αλλά μπορεί να χρησιμοποιηθεί και αυτόνομα αν χρειάζεται από τον προγραμματιστή. Ως παράμετροι δίνονται το ιδιωτικό κλειδί της ψηφιακής υπογραφής, τα δεδομένα προς υπογραφή καθώς και ένα salt για ακόμη μεγαλύτερη προστασία των

δεδομένων. Στην εφαρμογή που υλοποιήσαμε το salt ήταν ο ΑΦΜ του κάθε χρήστη, όμως αυτό είναι ευέλικτο και μπορεί να χρησιμοποιηθεί οποιαδήποτε άλλη συμβολοσειρά. Η μέθοδος αυτή χρησιμοποιείται μόνο για τον αλγόριθμο ψηφιακών υπογραφών DSA.

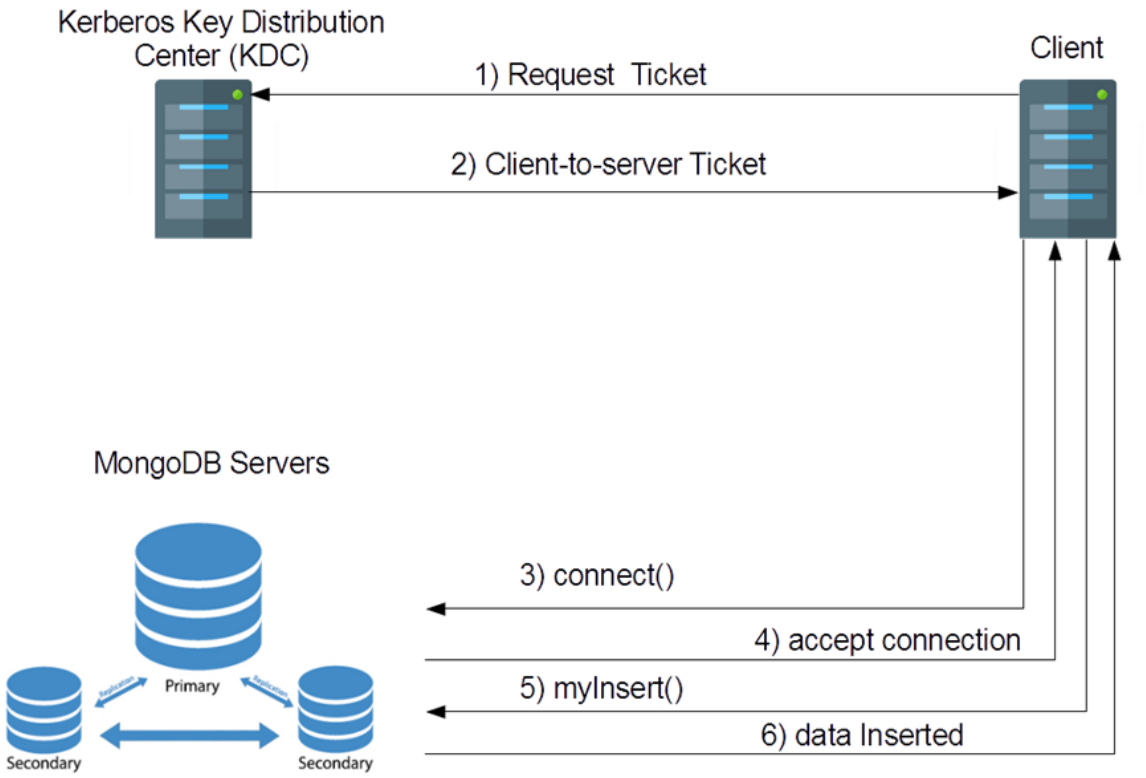
validate*(MongoClient mClient, String data, byte[] signature, String uName)*

mClient	Η ενεργή σύνδεση με την MongoDB
data	Τα δεδομένα για τα οποία θέλουμε να επικυρώσουμε την υπογραφή
signature	Η υπογραφή των προηγούμενων δεδομένων
uName	Ο χρήστης στον οποίο ανήκει η υπογραφή
returns	boolean

Για την επικύρωση των ψηφιακών υπογραφών χρησιμοποιείται αυτή η μέθοδος. Αφού δοθούν ως παράμετροι το δεδομένο, η ψηφιακή υπογραφή και ο χρήστης που το υπέγραψε, η μέθοδος ανακτά το δημόσιο κλειδί του χρήστη από τη βάση δεδομένων (getPK) και επαληθεύει την εγκυρότητα της υπογραφής. Εάν η υπογραφή δεν είναι γνήσια εμφανίζεται στον χρήστη κατάλληλο μήνυμα από το γραφικό περιβάλλον ώστε να προσέχει για ψευδή στοιχεία. Η μέθοδος αυτή χρησιμοποιείται μόνο για τον αλγόριθμο ψηφιακών υπογραφών DSA.

Διαδικασία αποστολής ενός αιτήματος myInsert()

Στο σχήμα που ακολουθεί παρουσιάζεται ένα communication diagram που δείχνει όλες τις αποστολές μηνυμάτων για να πραγματοποιηθεί ένα Insert ερώτημα στη βάση δεδομένων. Θεωρούμε ότι έχει γίνει η αρχική ανταλλαγή μηνυμάτων του χρήστη με τον Κέρβερο και ο χρήστης έχει παραλάβει το TGT του (λεπτομερείς αναλύθηκαν στη παράγραφο 4.2.1 στο Σχήμα 10: Ανταλλαγές μηνυμάτων στο πρωτόκολλο του Kerberos). Έχει δηλαδή γίνει η πρώτη φάση του πρωτοκόλλου του Κέρβερου και τώρα ζητούνται κάποιες υπηρεσίες από τη MongoDB και φυσικά αυτό γίνεται μέσω του ticketing system του Κέρβερου.



Σχήμα 12: Communication diagram ενός Insert

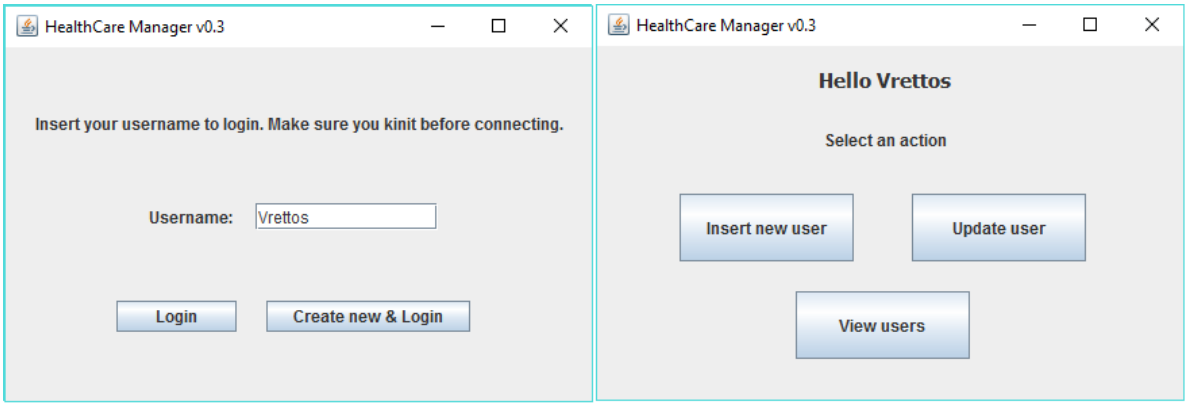
Αρχικά, στο **(1)** μήνυμα ο client ζητάει από τον Κέρβερο ένα ticket για να χρησιμοποιήσει κάποια υπηρεσία της MongoDB (στην παραπάνω περίπτωση θέλει να κάνει ένα Insert). Το αίτημα αποτελείται από το TGT που διαθέτει και από το όνομα της υπηρεσίας που ζητάει. Ο Κέρβερος, εφόσον είναι σωστό το TGT του χρήστη και υπάρχει η ζητούμενη υπηρεσία, στέλνει στον client ένα Client-to-Server ticket **(2)**. Στη συνέχεια ο χρήστης χρησιμοποιεί αυτό το ticket για να κάνει Connect **(3)** στη βάση δεδομένων. Η MongoDB ελέγχει το ticket και αποδέχεται ή απορρίπτει το αίτημα **(4)**. Τέλος ο χρήστης στέλνει τα δεδομένα που θέλει να εισάγει στη βάση **(5)** και τελειώνει η επικοινωνία με ένα επιβεβαιωτικό μήνυμα **(6)**.

5.3 Παράδειγμα λειτουργίας της εφαρμογής

Για τη δοκιμή και την παρουσίαση των προηγούμενων μεθόδων και υπηρεσιών κατασκευάστηκε μια εφαρμογή σε γραφικό περιβάλλον που αντικατοπτρίζει την ενδεδειγμένη λειτουργία του συστήματος .

Αρχικά ο χρήστης πρέπει να πάρει τα απαραίτητα πιστοποιητικά της ταυτότητάς του από την υπηρεσία του Κέρβερου. Αυτό γίνεται εξωτερικά της εφαρμογής από την κονσόλα.

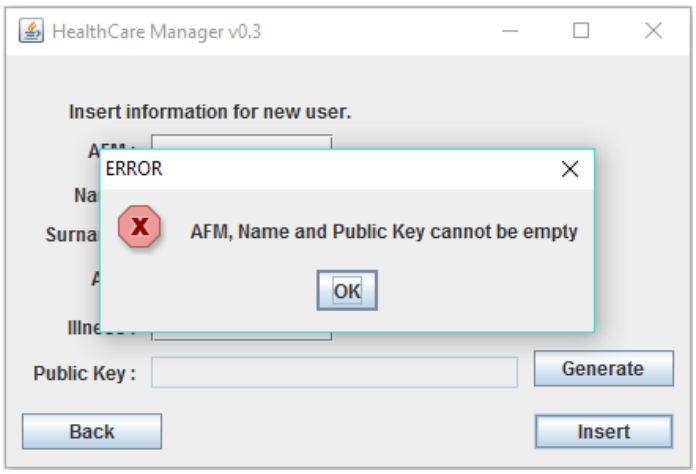
Στη συνέχεια ο χρήστης μπορεί μόνο με το username του να συνδεθεί στην βάση δεδομένων MongoDB. Αν δεν διαθέτει δημόσιο και ιδιωτικό κλειδί για την ψηφιακή υπογραφή του μπορεί να τα δημιουργήσει πριν συνδεθεί στο σύστημα, ώστε να μπορεί να το χρησιμοποιήσει χωρίς σφάλματα.



Σχήμα 13: Αρχική σελίδα εφαρμογής

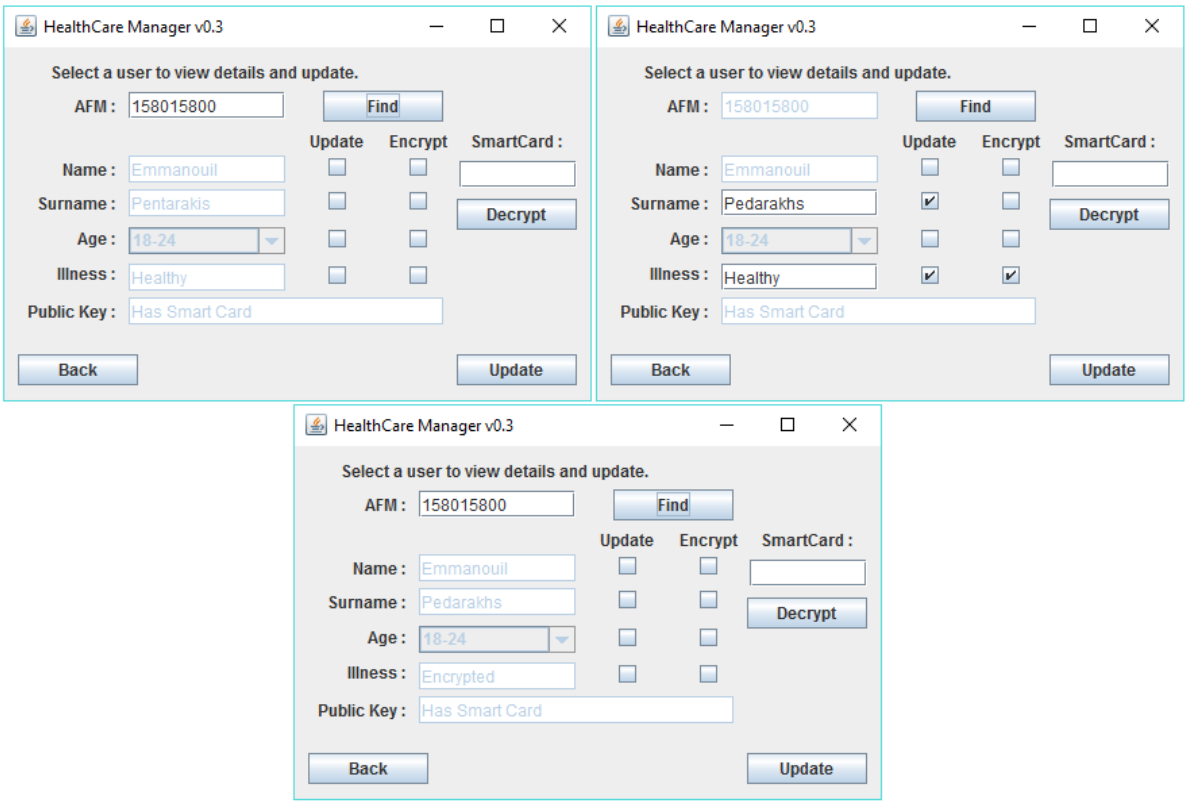
Αφού γίνει η πιστοποίηση και ο χρήστης συνδεθεί στο σύστημα, του δίνονται οι επιλογές για εισαγωγή νέου χρήστη, για ενημέρωση των πληροφοριών ενός εγγεγραμμένου χρήστη ή για προβολή των στοιχείων ενός χρήστη.

Κατά την εισαγωγή ενός νέου χρήστη είναι απαραίτητη η εισαγωγή του ΑΦΜ, του ονόματος καθώς και η δημιουργία ενός Δημοσίου κλειδιού (μαζί με το αντίστοιχο ιδιωτικό κλειδί που αποθηκεύεται στην Smart Card του χρήστη) για τυχόν κρυπτογράφηση των ευαίσθητων πληροφοριών του. Γίνεται επίσης έλεγχος εγκυρότητας των δεδομένων, όπως για παράδειγμα ο ΑΦΜ να αποτελείται μόνο από αριθμητικούς χαρακτήρες και να είναι μοναδικός για τον κάθε χρήστη (input validation).



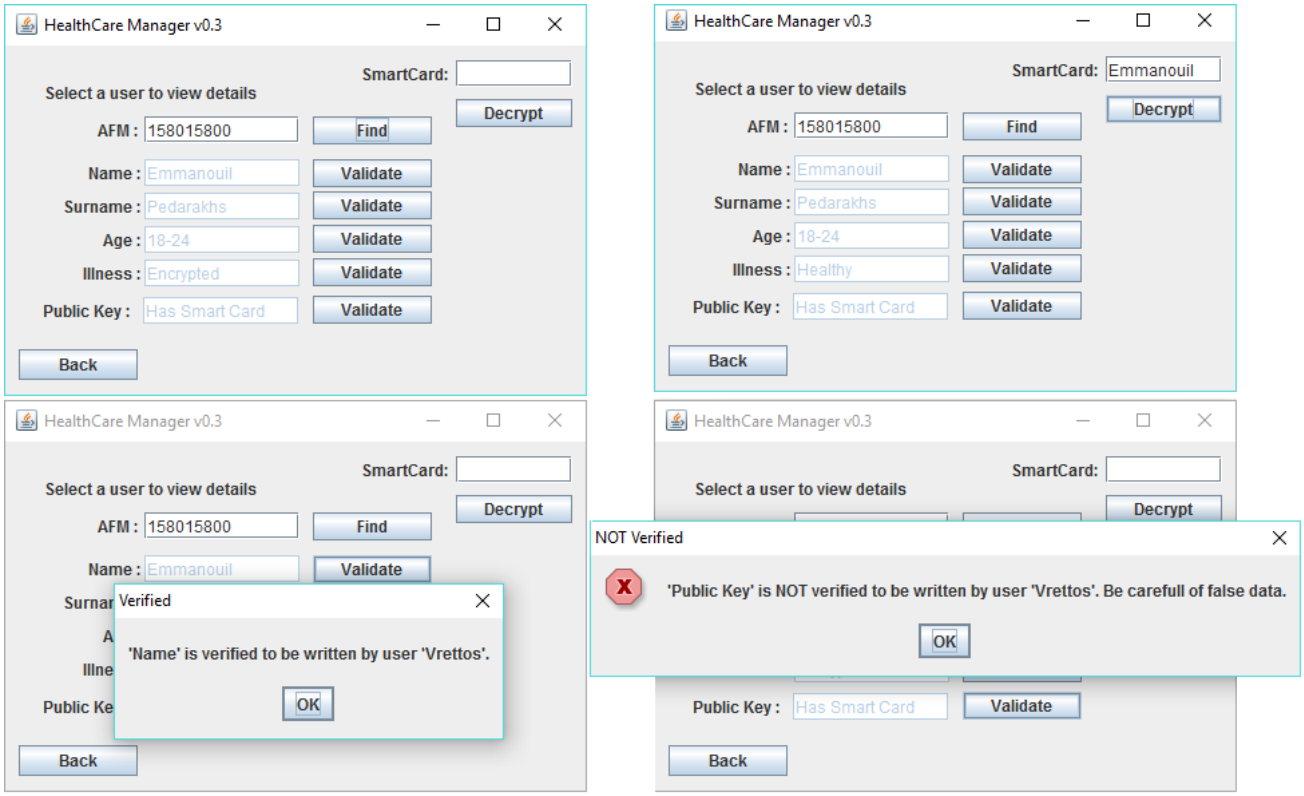
Σχήμα 14: Έλεγχος εγκυρότητας δεδομένων μέσα από την εφαρμογή

Κατά την επεξεργασία των πληροφοριών ενός χρήστη, αρχικά γίνεται εύρεση του χρήστη από το μοναδικό ΑΦΜ του και παρουσιάζονται οι πληροφορίες του. Στη συνέχεια μπορούμε να επεξεργαστούμε τις πληροφορίες αυτές και να τις αλλάξουμε ή να τις κρυπτογραφήσουμε. Δίνεται επίσης η δυνατότητα για αποκρυπτογράφηση των κρυπτογραφημένων δεδομένων με τη χρήση του κωδικού της Smart Card έτσι ώστε να γίνει στη συνέχεια η επεξεργασία και η αλλαγή των δεδομένων.



Σχήμα 15: Ενημέρωση ενός χρήστη και κρυπτογράφηση των ευαίσθητων πληροφοριών που τον αφορούν

Στην οθόνη προβολής των χρηστών ακολουθείται αρχικά ίδια διαδικασία με την προηγούμενη οθόνη. Γίνεται εισαγωγή του ΑΦΜ ενός χρήστη και εμφανίζονται οι λεπτομέρειες που αφορούν τον φάκελό του. Στη συνέχεια μπορούμε να αποκρυπτογραφήσουμε οποιαδήποτε πληροφορία είναι κρυπτογραφημένη αν έχουμε στη διάθεσή μας την Smart Cart του χρήστη.



Σχήμα 16: Εμφάνιση λεπτομερειών για ένα χρήστη και επικύρωση των ψηφιακών υπογραφών

Σε αυτή την οθόνη μπορεί επίσης να γίνει η επικύρωση τον πληροφοριών που έχουν γραφτεί στην καρτέλα του ασθενή. Με τη χρήση του κουμπιού “Validate” βλέπουμε αν η ψηφιακή υπογραφή είναι αληθής (επικυρώνεται το integrity και το non-repudiation) και το όνομα του χρήστη που υπέγραψε για αυτό το δεδομένο. Ακόμα και αν το δεδομένο είναι κρυπτογραφημένο μπορούμε να δούμε ποιος το έχει υπογράψει και αν είναι έγκυρη η υπογραφή.

Οι λειτουργίες που αναφέρθηκαν σε προηγούμενες παραγράφους (Kerberos authentication services, transport encryption, encryption at rest, replication κτλ) δεν είναι εμφανείς μέσα από την εφαρμογή και από τον χρήστη της εφαρμογής, αλλά βρίσκονται σε ένα χαμηλότερο στρώμα και γίνονται κρυφά χωρίς να αποσπούν την προσοχή του χρήστη.

ΚΕΦΑΛΑΙΟ 6

Αξιολόγηση

Στόχος της παρούσας διπλωματικής ήταν να εξεταστούν σε βάθος οι περισσότερες πτυχές που πρέπει να ελέγχει ένα ολοκληρωμένο σύστημα ασφαλείας πάνω σε μη σχεσιακές βάσεις δεδομένων, καθώς και να αναπτυχθεί ένα τέτοιο πρότυπο σύστημα ασφάλειας.

6.1 Αξιολόγηση της λύσης

Σημαντικό κομμάτι της εκπόνησης της εργασίας, το οποίο και δεν είναι εύκολο να μετρηθεί ποσοτικά, ήταν η μελέτη, κατανόηση και ανάλυση των απαιτήσεων και των χαρακτηριστικών ενός συστήματος ασφαλείας. Εξετάστηκαν σε θεωρητικό επίπεδο και υλοποιήθηκαν τεχνικά μέθοδοι για πιστοποίηση και εξουσιοδότηση χρηστών, για εμπιστευτικότητα, ακεραιότητα και διαθεσιμότητα των δεδομένων, για μη αποκήρυξη ευθυνών και γνώση της προέλευσης των δεδομένων καθώς και προστασία από κακόβουλες επιθέσεις πάνω στη βάση δεδομένων.

Η πιστοποίηση των χρηστών επιτεύχθηκε με τη χρήστη του πρωτοκόλλου Kerberos σε συνεργασία με την MongoDB. Για τη διασφάλιση της εμπιστευτικότητας των δεδομένων χρησιμοποιήθηκε η κρυπτογράφηση δημοσίου κλειδιού, ενώ για τη μη αποποίηση ευθυνών και καταγραφή της προέλευσης των δεδομένων χρησιμοποιήθηκε ο αλγόριθμος και η μέθοδος των ψηφιακών υπογραφών. Η ακεραιότητα των δεδομένων εξασφαλίστηκε από τις διάφορες προηγούμενες τεχνικές που χρησιμοποιήθηκαν, όπως η κρυπτογράφηση και οι ψηφιακές υπογραφές, και ακόμα μέσα από integrity checks και input validation λογισμικό. Τέλος η διαθεσιμότητα των δεδομένων διασφαλίστηκε με τη χρήστη της τεχνικής μεθόδου του replication που παρέχει η MongoDB.

Εκτός από την μελέτη των παραπάνω εννοιών και τεχνικών κατασκευάστηκε ένα API που παρέχει τους προηγούμενους μηχανισμούς ασφαλείας και τις υπηρεσίες οργανωμένες και συγκεντρωμένες. Κατασκευάστηκε επίσης μια πρότυπη εφαρμογή με γραφικό περιβάλλον που χρησιμοποιεί το παραπάνω API και συνδέεται με τη βάση δεδομένων μέσω του

συστήματος του Kerberos. Ο χρήστης με αυτή την εφαρμογή μπορεί να εκτελέσει διάφορες λειτουργίες που αναφέρθηκαν προηγουμένως, όπως εισαγωγή και ανανέωση των πληροφοριών ενός χρήστη, υπογραφή νέων δεδομένων και έλεγχος των υπαρχόντων ψηφιακών υπογραφών και κρυπτογράφηση - αποκρυπτογράφηση των δεδομένων αν έχει τα κατάλληλα εργαλεία (δικαιώματα ανάγνωσης, Smart Card, κτλ).

6.2 Μελλοντικές επεκτάσεις και στοιχεία προς έρευνα

Η υλοποίηση της παρούσας διπλωματικής εργασίας αποτελεί ικανοποιητική λύση για τη συνολική προστασία των δεδομένων με χρήση μιας μη σχεσιακής βάσης.

Θα μπορούσε ωστόσο να υπάρξει βελτιστοποίηση των τεχνικών που χρησιμοποιήθηκαν ώστε να επιτευχθούν μεγαλύτερες ταχύτητες επεξεργασίας των δεδομένων για μεγάλους όγκους δεδομένων και για ταυτόχρονα αιτήματα που είναι πιθανόν να γίνονται σε ένα τέτοιο σύστημα. Θα μπορούσε επίσης να γίνει επιπλέον έρευνα και ανάλυση σε μερικά δευτερεύοντα χαρακτηρίστηκα του συστήματος, όπως τα auditing εργαλεία και υπηρεσίες, το σύστημα access control της MongoDB, έρευνα για ασφαλείς μεθόδους για data mining και analytics, κτλ.

Αντικείμενο προς μελέτη αποτελεί επίσης η υλοποίηση μιας μεγαλύτερης και πιο οργανωμένης εφαρμογής και γραφικού περιβάλλοντος για διευκόλυνση των χρηστών του συστήματος, παρόλο που δημιουργήθηκε μια αντίστοιχη απλή πλατφόρμα που χρησιμοποιεί το API που κατασκευάστηκε.

Είναι επίσης ανοικτό το ζήτημα της επέκτασης του API με επιπλέον λειτουργίες, όπως η υλοποίηση της λειτουργίας διαγραφής δεδομένων από τη βάση. Η περαιτέρω μελέτη των διάφορων μεθόδων για ασφαλή αποθήκευση των μυστικών κλειδιών (όπου αυτά είναι απαραίτητα) αποτελεί επίσης ένα σημαντικό ερευνητικό κομμάτι. Τέτοιες περιπτώσεις αποτελεί η αποθήκευση των ιδιωτικών κλειδιών της κρυπτογράφησης, όπως για παράδειγμα η μελέτη των Έξυπνων καρτών (Smart Cards) που αναφέρθηκαν σε προηγούμενα κεφάλαια.

Βιβλιογραφία

[1] Adrian McCullagh, William Caelli. Non-Repudiation in the Digital Environment. *First Monday* 2000; 5(8).

[2] Ameya Nayak, Anil Poriya, Dikshay Poojary. Type of NOSQL Databases and its Comparison with Relational Databases. *International Journal of Applied Information Systems (IJAIS)* 2013; 5(4).

[3] Anthony Henderson. The CIA Triad: Confidentiality, Integrity, Availability. *Panmore Institute* 2017; *http://panmore.com/the-cia-triad-confidentiality-integrity-availability (accessed 28 June 2017).*

[4] Evgeny Milanov. *The RSA Algorithm*; 2009.

[5] Hector Garcia-Molina, Jeff Ullman, Jennifer Widom. *Database Systems - The Complete Book*, 2nd ed. New Jersey: Prentice Hall; 2008; p2-11, 239-241.

[6] Jay Runkel. *Making HIPAA Compliant Applications with MongoDB. https://www.mongodb.com/blog/post/making-hipaa-compliant-applications-mongodb (accessed 20 May 2017).*

[7] Johannes Buchmann. *The Digital Signature Algorithm (DSA)* ; 2001

[8] MIT Kerberos Team. *Kerberos: The Network Authentication Protocol. https://web.mit.edu/kerberos/ (accessed 29 June 2017).*

[9] MongoDB, Inc. MongoDB Security Architecture. *A MongoDB White Paper* 2016; p2-10. *https://webassets.mongodb.com/_com_assets/collateral/MongoDB_Security_Architecture_WP.pdf (accessed 27 June 2017).*

[10] MongoDB, Inc. *The MongoDB 3.4 Manual. https://docs.mongodb.com/manual/ (accessed 17 July 2017).*

[11] Office for Civil Rights. *Summary of the HIPAA Security Rule. https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html (accessed 27 June 2017).*

[12] Oracle. An Enterprise Architect's Guide to Big Data. *Oracle Enterprise Architecture White Paper* 2016; p7-10. *http://www.oracle.com/technetwork/topics/entarch/articles/oea-big-data-guide-1522052.pdf (accessed 18 February 2017).*

[13] Organization for the Advancement of Structured Information Standards (OASIS). *Key Management Interoperability Protocol (KMIP)*; 2009. http://xml.coverpages.org/KMIP/KMIP-WhitePaper.pdf (accessed 13 July 2017).

[14] Pablo Valerio. US Firms Looking to Europe for Data Protection. *Network Computing* 2016; *http://www.networkcomputing.com/cloud-infrastructure/us-firms-looking-europe-data-protection/129277031 (accessed 18 January 2017).*

[15] Ravi Sandhu, Sushil Jajodia. Integrity Mechanisms in Database Management Systems. 1990; *http://profsandhu.com/articles/infosec_collection/27.pdf (accessed 28 June 2017).*

[16] Rolf Oppliger. *SSL and TLS : Theory and Practice*. London: ARTECH HOUSE; 2009.

[17] Seth Gilbert, Nancy A. Lync. Perspectives on the CAP Theorem. *Computer* 2012; 45(2).

[18] Siegfried Herda. Non-repudiation: Constituting evidence and proof in digital cooperation. *Computer Standards & Interfaces* 1995; 17(1).

[19] Stéphane Guilloteau. Privacy in Cloud Computing. *ITU-T Technology Watch Report* 2012.

[20] Vic (J.R.) Winkler. Cloud Computing: Data Privacy in the Cloud. *TechNet Magazine* 2012; *https://technet.microsoft.com/en-us/library/jj554305.aspx (accessed 26 July 2017).*

[21] Vrettos Moulos. *Outsourcing Information Security Management*. London: Kings College of London; 2008.

[22] Warren C. Axelrod. *Outsourcing Information Security*; p27-69. Norwood: Artech House; 2004.

[23] Warwick Ashford. *Bad outsourcing decisions cause 63% of data breaches. http://www.computerweekly.com/news/2240178104/Bad-outsourcing-decisions-cause-63-of-data-breaches (accessed 9 February 2017).*

Παράρτημα

Α. Πηγαίος κώδικας εφαρμογής

Ακολουθεί ο πηγαίος κώδικας του γραφικού περιβάλλοντος της εφαρμογής, καθώς και το API σε μορφή κλάσεων και μεθόδων στη Java.

A.1 MainGUI.java

Η κλάση αυτή περιέχει όλο τον πηγαίο κώδικα που αφορά το γραφικό περιβάλλον της εφαρμογής. Παρακάτω παρουσιάζεται ένα κομμάτι από το γραφικό αυτό περιβάλλον (το παράθυρο που ο χρήστης μπορεί να κάνει login, το κεντρικό παράθυρο μετά την σύνδεσή του και το παράθυρο που γίνεται η προσθήκη ενός καινούργιου ασθενή στη βάση δεδομένων). Δεν παρατίθεται όλος ο πηγαίος κώδικας, καθώς δεν αφορά κεντρικό κομμάτι της διπλωματικής και της έρευνας που έγινε, αλλά έγινε μόνο για έλεγχο τον υπολοίπων στοιχείων (API – υπηρεσίες Κέρβερου και MongoDB).

```
/*Copyright (C) 2017  PENTARAKIS EMMANOUIL


Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. */


import java.awt.*;
import javax.swing.*;

import org.bson.Document;
import org.bson.types.Binary;
import org.eclipse.wb.swing.FocusTraversalOnArray;

import com.mongodb.MongoClient;
import com.mongodb.client.*;
```

```
public class MainGUI {

    private JFrame frame;
    private Database connector;
    private MongoClient mClient ;
    private String databaseName = "HealthCare";
    private KeyManager keyManager = new KeyManager();

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    MainGUI window = new MainGUI();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }



    /**
     * Create the application.
     */
    public MainGUI() {
        initialize();
    }



    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {

        //Create frame and login panel
        frame = new JFrame("HealthCare Manager v0.3");
        frame.setBounds(100, 100, 450, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        JPanel panel = new JPanel();
        panel.setBounds(0, 0, 434, 261);
        frame.getContentPane().add(panel);
        panel.setLayout(new CardLayout(0, 0));

        JPanel login_panel = new JPanel();
        panel.add(login_panel, "login_panel");
        login_panel.setLayout(null);

        JLabel inserunameLB = new JLabel("Insert your username to login. Make sure you
kinit before connecting.");
        inserunameLB.setHorizontalAlignment(SwingConstants.CENTER);
        inserunameLB.setBounds(10, 37, 414, 38);
        login_panel.add(inserunameLB);
```

```
JLabel unameLB = new JLabel("Username:");
unameLB.setHorizontalAlignment(SwingConstants.RIGHT);
unameLB.setBounds(93, 118, 75, 14);
login_panel.add(unameLB);

JTextField unameTXT = new JTextField();
unameTXT.setBounds(185, 115, 135, 20);
login_panel.add(unameTXT);
unameTXT.setColumns(10);
JLabel lblUser = new JLabel("hello ");

JButton btnLogin = new JButton("Login");
btnLogin.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        connector = new Database(unameTXT.getText(),databaseName);
        try {
            mClient = connector.Connect();
        } catch (Exception e1) {
            //Auto-generated catch block
            e1.printStackTrace();
        }
        //System.out.println(unameTXT.getText() + " " + databaseName);
        CardLayout card = (CardLayout)panel.getLayout();
        card.show(panel, "home_panel");
        //Set hello message for next panel
        lblUser.setText("Hello "+unameTXT.getText());
    }
});
btnLogin.setBounds(82, 187, 89, 23);
login_panel.add(btnLogin);

JButton btnCreateNew = new JButton("Create new & Login");
btnCreateNew.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        connector = new Database(unameTXT.getText(),databaseName);
        try {
            KeyPair keys = keyManager.createKeyPairs();
            keyManager.saveKeysToFile(keys, unameTXT.getText());
            mClient = connector.Connect();
            connector.storeSignaturePK(mClient, keys.getPublic());
        } catch (Exception e) {
            // Auto-generated catch block
            e.printStackTrace();
        }

        CardLayout card = (CardLayout)panel.getLayout();
        card.show(panel, "home_panel");
        //Set hello message for next panel
        lblUser.setText("Hello "+unameTXT.getText());

    }
});
btnCreateNew.setBounds(193, 187, 151, 23);
login_panel.add(btnCreateNew);

//Create home panel
JPanel home_panel = new JPanel();
panel.add(home_panel, "home_panel");
home_panel.setLayout(null);

JLabel lblPleaseSelectThe = new JLabel("Select an action");
```

```
        lblPleaseSelectThe.setHorizontalAlignment(SwingConstants.CENTER);
        lblPleaseSelectThe.setBounds(75, 62, 277, 14);
        home_panel.add(lblPleaseSelectThe);

        lblUser.setHorizontalAlignment(SwingConstants.CENTER);
        lblUser.setFont(new Font("Tahoma", Font.BOLD, 15));
        lblUser.setBounds(75, 0, 277, 50);
        home_panel.add(lblUser);

        JButton btnInsertNewUser = new JButton("Insert new user");
        btnInsertNewUser.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                CardLayout card = (CardLayout)panel.getLayout();
                card.show(panel, "insert_panel");
            }
        });
        btnInsertNewUser.setBounds(61, 109, 129, 50);
        home_panel.add(btnInsertNewUser);

        JButton btnUpdateUser = new JButton("Update user");
        btnUpdateUser.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                CardLayout card = (CardLayout)panel.getLayout();
                card.show(panel, "update_panel");
            }
        });
        btnUpdateUser.setBounds(233, 109, 129, 50);
        home_panel.add(btnUpdateUser);

        JButton btnViewUsers = new JButton("View users");
        btnViewUsers.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                CardLayout card = (CardLayout)panel.getLayout();
                card.show(panel, "users_panel");
            }
        });
        btnViewUsers.setBounds(147, 181, 129, 50);
        home_panel.add(btnViewUsers);

        //Create insert_user panel
        JPanel insert_panel = new JPanel();
        panel.add(insert_panel, "insert_panel");
        insert_panel.setLayout(null);

        JLabel label = new JLabel("Name :");
        label.setHorizontalAlignment(SwingConstants.RIGHT);
        label.setBounds(10, 81, 72, 14);
        insert_panel.add(label);

        JTextField insName = new JTextField();
        insName.setColumns(10);
        insName.setBounds(92, 78, 116, 20);
        insert_panel.add(insName);

        JComboBox<String> insAgeCB = new JComboBox<String>();
        insAgeCB.setModel(new DefaultComboBoxModel<String>(new String[] {"<12", "12-
18", "18-24", "24-30", "30-45", "45+"}));
        insAgeCB.setBounds(92, 131, 116, 20);
        insert_panel.add(insAgeCB);

        JLabel lblAge = new JLabel("Age :");
```

```
        lblAge.setHorizontalAlignment(SwingConstants.RIGHT);
        lblAge.setBounds(10, 132, 72, 19);
        insert_panel.add(lblAge);

        JTextField insIll = new JTextField();
        insIll.setColumns(10);
        insIll.setBounds(92, 162, 116, 20);
        insert_panel.add(insIll);

        JLabel lblIlness = new JLabel("Illness :");
        lblIlness.setHorizontalAlignment(SwingConstants.RIGHT);
        lblIlness.setBounds(10, 162, 72, 20);
        insert_panel.add(lblIlness);

        JLabel lblAfm = new JLabel("AFM :");
        lblAfm.setHorizontalAlignment(SwingConstants.RIGHT);
        lblAfm.setBounds(10, 53, 72, 14);
        insert_panel.add(lblAfm);

        JLabel lblSurname = new JLabel("Surname :");
        lblSurname.setHorizontalAlignment(SwingConstants.RIGHT);
        lblSurname.setBounds(10, 106, 72, 15);
        insert_panel.add(lblSurname);

        JTextField insSur = new JTextField();
        insSur.setColumns(10);
        insSur.setBounds(92, 103, 116, 20);
        insert_panel.add(insSur);

        JTextField insAFM = new JTextField();
        insAFM.setColumns(10);
        insAFM.setBounds(92, 50, 116, 20);
        insert_panel.add(insAFM);

        JLabel lblPublicKey = new JLabel("Public Key :");
        lblPublicKey.setHorizontalAlignment(SwingConstants.RIGHT);
        lblPublicKey.setBounds(10, 191, 72, 20);
        insert_panel.add(lblPublicKey);

        JTextField insPK = new JTextField();
        insPK.setEditable(false);
        insPK.setColumns(10);
        insPK.setBounds(92, 191, 233, 20);
        insert_panel.add(insPK);

        JButton btnInsert = new JButton("Insert");
        btnInsert.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                PrivateKey priv = null;

                //if AFM or public key is empty, then don't insert it
                if (insAFM.getText().isEmpty() || insPK.getText().isEmpty() )
                {
                    JOptionPane.showMessageDialog(frame,"AFM, Name and Public Key
cannot be empty","ERROR",JOptionPane.ERROR_MESSAGE);
                }
                else
                {
                    String AFM = insAFM.getText();
                    try {
                        priv = keyManager.getPrivateKey(unameTXT.getText());
```

```
                        //insert the data (user is logged in as uname)
                        connector.myInsert(mClient,AFM,priv);
                        //insert the rest of data with update commands
                        update(insAFM.getText(),insName.getText(), insSur.getText(),
insAgeCB.getSelectedItem().toString(),
insIll.getText(),keyManager.getPublicKeyRSA(insName.getText()),priv);

                        JOptionPane.showMessageDialog(frame,"User added
successfully.","User inserted",JOptionPane.PLAIN_MESSAGE);
                    } catch (Exception e1) {
                        e1.printStackTrace();
                    }
                }
            }
        });
        btnInsert.setBounds(335, 227, 89, 23);
        insert_panel.add(btnInsert);

        JButton back_admin = new JButton("Back");
        back_admin.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                insAFM.setText(null);
                insName.setText(null);
                insSur.setText(null);
                insAgeCB.setSelectedItem("<12");
                insIll.setText(null);
                insPK.setText(null);
                CardLayout card = (CardLayout)panel.getLayout();
                card.show(panel, "home_panel");
            }
        });
        back_admin.setBounds(10, 227, 89, 23);
        insert_panel.add(back_admin);

        JLabel lblInsertInformationFor = new JLabel("Insert information for new
user.");
        lblInsertInformationFor.setBounds(40, 28, 367, 14);
        insert_panel.add(lblInsertInformationFor);

        JButton btnGenerate = new JButton("Generate");
        btnGenerate.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                if (insName.getText().isEmpty())
                {
                    JOptionPane.showMessageDialog(frame,"Please insert a Name to
create his personal Public Key","ERROR",JOptionPane.ERROR_MESSAGE);
                }
                else
                {
                    try {
                        KeyPair kp = keyManager.createRSAkeyPair();
                        keyManager.saveKeysToFileRSA(kp, insName.getText());
                        insPK.setText("Generated and added to SmartCard");
                    } catch (IOException | NoSuchAlgorithmException |
NoSuchProviderException e1) {
                        e1.printStackTrace();
                    }
                }

            }
        });
```

```
        btnGenerate.setBounds(335, 187, 89, 23);
        insert_panel.add(btnGenerate);

       /*
        *
        *
        *
        (code continues with the rest of the GUI)
         not contained in this thesis for space saving
        *
        *
        *
       **/
}
```

A.2 KeyManager.java

```
import java.io.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.Cipher;
import com.mongodb.MongoClient;


public class KeyManager {

    private final String ALGORITHM = "RSA";

    public KeyPair createKeyPairs () throws Exception {

        KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA", "SUN");

        SecureRandom random = SecureRandom.getInstance("SHA1PRNG", "SUN");
        keyGen.initialize(1024, random);

        KeyPair pair = keyGen.generateKeyPair();
        return pair;
    }

    public KeyPair createRSAkeyPair() throws Exception {

        KeyPairGenerator keyGen = KeyPairGenerator.getInstance(ALGORITHM);

        SecureRandom random = SecureRandom.getInstance("SHA1PRNG", "SUN");

        // 1024 is keysize
        keyGen.initialize(1024, random);

        KeyPair generateKeyPair = keyGen.generateKeyPair();
        return generateKeyPair;
    }

    public void saveKeysToFile (KeyPair keys, String uname) throws IOException {

        PrivateKey priv = keys.getPrivate();
        PublicKey pub = keys.getPublic();

        /* save the public key in a file */
        byte[] key = pub.getEncoded();
        FileOutputStream keyfos = new FileOutputStream(uname+"Pub");
        keyfos.write(key);
        keyfos.close();

        /* save the private key in a file */
        byte[] key2 = priv.getEncoded();
        FileOutputStream key2fos = new FileOutputStream(uname+"Priv");
        key2fos.write(key2);
        key2fos.close();
    }


    public void saveKeysToFileRSA (KeyPair keys, String uname) throws IOException {

        PrivateKey priv = keys.getPrivate();
        PublicKey pub = keys.getPublic();
```

```
        /* save the public key in a file */
        byte[] key = pub.getEncoded();
        FileOutputStream keyfos = new FileOutputStream(uname+"RSAPub");
        keyfos.write(key);
        keyfos.close();

        /* save the private key in a file */
        byte[] key2 = priv.getEncoded();
        FileOutputStream key2fos = new FileOutputStream(uname+"RSAPriv");
        key2fos.write(key2);
        key2fos.close();
    }


    public PublicKey getPublicKey (String uname) throws Exception {

        FileInputStream keyfis = new FileInputStream(uname+"Pub");
        byte[] encKey = new byte[keyfis.available()];
        keyfis.read(encKey);
        keyfis.close();

        X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(encKey);

        KeyFactory keyFactory = KeyFactory.getInstance("DSA", "SUN");
        PublicKey pubKey = keyFactory.generatePublic(pubKeySpec);

        return pubKey;
    }



    public PrivateKey getPrivateKey (String uname) throws Exception {

        FileInputStream keyfis = new FileInputStream(uname+"Priv");
        byte[] encKey = new byte[keyfis.available()];
        keyfis.read(encKey);
        keyfis.close();

        PKCS8EncodedKeySpec privKeySpec = new PKCS8EncodedKeySpec(encKey);

        KeyFactory keyFactory = KeyFactory.getInstance("DSA");
        PrivateKey privKey = keyFactory.generatePrivate(privKeySpec);

        return privKey;
    }


    public PublicKey getPublicKeyRSA (String uname) throws Exception {

        FileInputStream keyfis = new FileInputStream(uname+"RSAPub");
        byte[] encKey = new byte[keyfis.available()];
        keyfis.read(encKey);
        keyfis.close();

        X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(encKey);

        KeyFactory keyFactory = KeyFactory.getInstance(ALGORITHM);
        PublicKey pubKey = keyFactory.generatePublic(pubKeySpec);

        return pubKey;
    }
```

```
    public PrivateKey getPrivateKeyRSA (String uname) throws Exception {

        FileInputStream keyfis = new FileInputStream(uname+"RSAPriv");
        byte[] encKey = new byte[keyfis.available()];
        keyfis.read(encKey);
        keyfis.close();

        PKCS8EncodedKeySpec privKeySpec = new PKCS8EncodedKeySpec(encKey);

        KeyFactory keyFactory = KeyFactory.getInstance(ALGORITHM);
        PrivateKey privKey = keyFactory.generatePrivate(privKeySpec);

        return privKey;
    }


    public byte[] encryptRSA(PublicKey key, String inputData) throws Exception {

        byte[] inputDataB = inputData.getBytes();

        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.PUBLIC_KEY, key);

        byte[] encryptedBytes = cipher.doFinal(inputDataB);

        return encryptedBytes;
    }

    public byte[] decryptRSA(PrivateKey key, byte[] inputData) throws Exception {

        Cipher cipher = Cipher.getInstance(ALGORITHM);
        cipher.init(Cipher.PRIVATE_KEY, key);

        byte[] decryptedBytes = cipher.doFinal(inputData);

        return decryptedBytes;
    }

}
```

A.3 Database.java

```
import java.security.*;
import java.security.spec.*;
import java.util.*;
import org.bson.Document;
import org.bson.types.Binary;
import com.mongodb.*;
import com.mongodb.client.*;



public class Database {


    private String uname = null;
    private String databaseName = null;


    public String getUname() {
        return uname;
    }


    public Database(String uname, String databaseName) {
        super();
        this.uname = uname;
        this.databaseName = databaseName;
    }



    /**
     * After obtaining credentials with "kinit" from command line, use Connect() to
open a connection to a MongoDB instance
     * @param name Username that has the Kerberosv5 credentials
     * @return MongoClient connection to database
     */
    public MongoClient Connect(String uname) throws Exception {
        //use kinit first to obtain credentials for kerberos
        MongoClient con = new MongoClient(new
MongoClientURI("mongodb://"+uname+"%40TELECOM.ECE.NTUA.GR@plrf20.telecom.ece.ntua.gr/
?authMechanism=GSSAPI&authSource=$external"));
        return con;
    }


    /**
     * Stores the given public key to database from the connected user (used for
DIGITAL SIGNATURES)
     * @param m Mongo client obtained by the "Connect" method
     * @param pk Public key to store in the database, used for Digital Signature
     */
    public void storeSignaturePK(MongoClient m, PublicKey pk) {
        MongoDatabase db = m.getDatabase(databaseName);
        MongoCollection<Document> c = db.getCollection("SignatureKeys");

        Document doc = new Document("Name", uname)
                .append("PublicKey", pk.getEncoded());
        c.insertOne(doc);
    }
```

```
    /**
     * Stores the given public key to database from the connected user (used for RSA
CRYPTOGRAPHY)
     * @param m Mongo client obtained by the "Connect" method
     * @param pk Public key to store in the database, used for RSA cryptography
     * @param AFM Unique number of person in the database
     * @param priv Private key used to digitally sign the data
     */
    public void storeRsaPK(MongoClient m, PublicKey pk, String AFM, PrivateKey priv)
throws Exception {
        MongoDatabase db = m.getDatabase(databaseName);
        MongoCollection<Document> c = db.getCollection( "BasicInfo");

        String key1 = "Public Key";
        Document doc = new Document(key1, pk.getEncoded());

        byte[] signature = signDataPK(pk.getEncoded(), priv, AFM);
        //Create the signature field and values
        //example "_Name" : ["diGiTaLS1gn4TuR3","NoLee"]
        doc.append("_"+key1, Arrays.asList(signature, uname));
        Document doc2 = new Document("$set",doc);
        //doc to query, we expect one result
        Document queryDoc = new Document("AFM", AFM);

        c.updateOne(queryDoc, doc2);
    }

    /**
     * Gets the public key for the desired username.
     * @param m Mongo client obtained by the "Connect" method
     * @param username Username of the author of the public key
     * @return Public key for the selected user from the database (Digital
signatures)
     */
    public PublicKey getSignaturePK (MongoClient m, String username) throws Exception
{
        MongoDatabase db = m.getDatabase(databaseName);
        MongoCollection<Document> c = db.getCollection( "SignatureKeys");

        //get the binary public key from Database, we expect one result
        Document doc = new Document("Name", username);
        MongoCursor<Document> cursor = c.find(doc).iterator();

        Binary binKey=null;
        try {
            binKey = (Binary) cursor.next().get("PublicKey");
        } finally {
            cursor.close();
        }

        //make a byte array from binary public key
        byte[] encKey = new byte[binKey.length()];
        encKey = binKey.getData();

        X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(encKey);

        KeyFactory keyFactory = KeyFactory.getInstance("DSA", "SUN");
        PublicKey pubKey = keyFactory.generatePublic(pubKeySpec);

        return pubKey;
    }
```

```
    /**
     * Gets the public key for the desired username.
     * @param m Mongo client obtained by the "Connect" method
     * @param username Username of the author of the public key
     * @return Public key for the selected user from the database (RSA cryptography)
     */
    public PublicKey getRsaPK (MongoClient m, String AFM) throws Exception {
        MongoDatabase db = m.getDatabase(databaseName);
        MongoCollection<Document> c = db.getCollection("BasicInfo");

        //get the binary public key from Database, we expect one result
        Document doc = new Document("AFM", AFM);
        MongoCursor<Document> cursor = c.find(doc).iterator();

        Binary binKey=null;
        try {
            binKey = (Binary) cursor.next().get("Public Key");
        } finally {
            cursor.close();
        }

        //make a byte array from binary public key
        byte[] encKey = new byte[binKey.length()];
        encKey = binKey.getData();

        X509EncodedKeySpec pubKeySpec = new X509EncodedKeySpec(encKey);

        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        PublicKey pubKey = keyFactory.generatePublic(pubKeySpec);

        return pubKey;
    }



    /**
     * This method is used to insert a new user to the database, must have AFM
     * @param m Mongo client obtained by the "Connect" method
     * @param AFM is the unique tax number for each person
     * @param priv Private key used for Digital Signature
     */
    public void myInsert(MongoClient m, String AFM, PrivateKey priv) throws Exception
{
        MongoDatabase db = m.getDatabase(databaseName);
        MongoCollection<Document> c = db.getCollection( "BasicInfo");

        Document doc = new Document("AFM", AFM);

        byte[] signature = signData(AFM,priv,AFM);
        //Create the signature field and values
        //example "_Name" : ["diGiTaLS1gn4TuR3","NoLee"]
        doc.append("_AFM", Arrays.asList(signature, uname));
        c.insertOne(doc);
    }
```

```
    /**
     * This method is used to update or insert into a document data to the selected
database.
     * @param AFM is the unique tax number for each person
     * @param m Mongo client obtained by the "Connect" method
     * @param doc Must contain one field and one value (Strings)
     * @param priv Private key used for Digital Signature
     */
    public void myUpdate(String AFM,MongoClient m, Document doc, PrivateKey priv)
throws Exception {
        MongoDatabase db = m.getDatabase(databaseName);
        MongoCollection<Document> c = db.getCollection( "BasicInfo");

        //Document has one key and one data
        //Get the key and data
        Set<String> set = doc.keySet();
        Iterator<String> iter = set.iterator();
        String key1 = iter.next();
        String data1 = doc.getString(key1);

        byte[] signature = signData(data1,priv,AFM);
        //Create the signature field and values
        //example "_Name" : ["diGiTaLS1gn4TuR3","NoLee"]
        doc.append("_"+key1, Arrays.asList(signature, uname));
        Document doc2 = new Document("$set",doc);
        //doc to query, we expect one result
        Document queryDoc = new Document("AFM", AFM);

        c.updateOne(queryDoc, doc2);
    }



    /**
     * Uses the provided private key to sign the provided data for the Digital
Signature procedure
     * @param data String with data to be signed
     * @param priv Private key used for Digital Signature
     * @param salt Salt used in Digital Signature so you cannot copy both signature
and data, usually is the AFM of the person the data concerns
     * @return a byte array containing the Digital Signature data
     */
    private byte[] signData (String data, PrivateKey priv, String salt) throws
Exception{

        Signature dsa = Signature.getInstance("SHA1withDSA", "SUN");
        dsa.initSign(priv);

        String saltedData = data + ":" + salt;

        byte[] dataB = saltedData.getBytes();

        dsa.update(dataB);

        byte[] realSig = dsa.sign();

        return realSig;
    }
```

```
    /**
     * Validates the Digital signature for provided data.
     * @param m Mongo client obtained by the "Connect" method
     * @param data String with data to be validated
     * @param signature Signature to be validated
     * @param uname Username of the person that signed the data
     * @return the result of the validation, true or false
     */
    public boolean validate (MongoClient m,String data,byte[] signature,String
username) throws Exception {
        /* get public key from database */
        PublicKey pubKey = getSignaturePK(m,username);

        /* create a Signature object and initialize it with the public key */
        Signature sig = Signature.getInstance("SHA1withDSA", "SUN");
        sig.initVerify(pubKey);

        /* Update and verify the data */
        sig.update(data.getBytes());

        boolean verifies = sig.verify(signature);

        System.out.println("signature verifies: " + verifies);
        return verifies;
    }
```

Β. Διαδικασία εγκατάστασης απαραίτητου λογισμικού

Παρακάτω δίνονται οι απαραίτητες βασικές πληροφορίες για την εγκατάσταση του απαραίτητου λογισμικού που χρησιμοποίησε το σύστημα για να λειτουργεί σωστά. Φυσικά για κάθε διαφορετικό υπολογιστή μπορεί να προκύψουν διαφορά προβλήματα, όπως εγκατάσταση κάποιων επιπλέον βιβλιοθηκών, αλλαγή των ονομάτων των εξυπηρετητών κτλ

Η εφαρμογή με το γραφικό περιβάλλον σχεδιάστηκε για να τρέχει σε windows, αλλά μπορεί με κάποιες αλλαγές στο γραφικό περιβάλλον να τρέξει και σε διαφορετικά λειτουργικά συστήματα. Για την εγκατάσταση αρκεί να γίνουν compile όλα τα αρχεία του πηγαίου κώδικα που δόθηκαν στο προηγούμενο παράρτημα.

Οι εντολές που είναι με έντονο χρώμα είναι αυτές που πληκτρολογεί ο χρήστης, ενώ χωρίς είναι οι αποκρίσεις της κονσόλας στις ενέργειες του χρήστη.

Β.1 Kerberos v5 σε Ubuntu 5.4.0-6ubuntu1~16.04.4

>sudo apt-get install krb5-{admin-server,kdc}

Default Kerberos version 5 realm? ***TELECOM.ECE.NTUA.GR***

Add locations of default Kerberos servers to /etc/krb5.conf? **Yes**

Kerberos servers for your realm: ***plrf21.telecom.ece.ntua.gr***

Administrative server for your Kerberos realm: ***plrf21.telecom.ece.ntua.gr***

Create the Kerberos KDC configuration automatically? **Yes**

Run the Kerberos V5 administration daemon (kadmind)? **Yes**

Δημιουργία ενός realm του Κέρβερου

>sudo krb5_newrealm

Enter KDC database master key: ***TYPE_YOUR_PASSWORD***

Re-enter KDC database master key to verify: ***TYPE_YOUR_PASSWORD***

Δημιουργία log files

```
>sudo mkdir /var/log/kerberos

>sudo touch /var/log/kerberos/{krb5kdc,kadmin,krb5lib}.log

>sudo chmod -R 750  /var/log/kerberos
```

Για εφαρμογή των αλλαγών κάνουμε restart τον εξυπηρετητή του Κέρβερου

```
>sudo invoke-rc.d krb5-admin-server restart

>sudo invoke-rc.d krb5-kdc restart
```

Ορισμός παραμέτρων, λογαριασμών χρηστών και υπηρεσιών στον Κέρβερο

```
>sudo kadmin.local

Authenticating as principal root/admin@TELECOM.ECE.NTUA.GR with password.

kadmin.local:  add_policy -minlength 8 -minclasses 3 admin

kadmin.local:  add_policy -minlength 8 -minclasses 4 host

kadmin.local:  add_policy -minlength 8 -minclasses 4 service

kadmin.local:  add_policy -minlength 8 -minclasses 2 user

kadmin.local:  addprinc -policy user Vrettos

Enter password for principal "Vrettos@TELECOM.ECE.NTUA.GR": PASSWORD

Re-enter password for principal " Vrettos@TELECOM.ECE.NTUA.GR ": PASSWORD

Principal " Vrettos@TELECOM.ECE.NTUA.GR " created.

kadmin.local:  addprinc -policy service -randkey
mongod/plrf20.telecom.ece.ntua.gr

Principal "mongod/plrf20.telecom.ece.ntua.gr@TELECOM.ECE.NTUA.GR" created.

kadmin.local:  ktadd -k /etc/krb5.keytab -norandkey
mongod/plrf20.telecom.ece.ntua.gr

kadmin.local:  addprinc -policy service -randkey
mongodb/plrf20.telecom.ece.ntua.gr

Principal "mongodb/plrf20.telecom.ece.ntua.gr@TELECOM.ECE.NTUA.GR "
created.

kadmin.local:  ktadd -k /etc/krb5.keytab -norandkey
mongodb/plrf20.telecom.ece.ntua.gr

kadmin.local:  addprinc -policy service -randkey
mongos/plrf20.telecom.ece.ntua.gr
```

```
Principal "mongos/plrf20.telecom.ece.ntua.gr@TELECOM.ECE.NTUA.GR "
created.

kadmin.local:  ktadd -k /etc/krb5.keytab -norandkey
mongos/plrf20.telecom.ece.ntua.gr

kadmin.local:  quit
```

Απόκτηση διαπιστευτηρίων (Kerberos service ticket) από τον Κέρβερο

```
>kinit Vrettos

Password for Vrettos@TELECOM.ECE.NTUA.GR: PASSWORD

>klist

Ticket cache:          FILE:/tmp/krb5cc_1000

Default principal:     Vrettos@TELECOM.ECE.NTUA.GR

Valid starting         07/17/2017 11:06:20

Expires                07/17/2017 21:06:20

Service principal      krbtgt/TELECOM.ECE.NTUA.GR@TELECOM.ECE.NTUA.GR
```

Απαραίτητα configuration files που πρέπει να βρίσκονται σε κάθε υπολογιστή που θέλει να συνδεθεί και να πιστοποιηθεί με τον Κέρβερο

```
/etc/krb5.conf

[libdefaults]

        default_realm = TELECOM.ECE.NTUA.GR

# The following krb5.conf variables are only for MIT Kerberos.

        krb4_config = /etc/krb.conf

        krb4_realms = /etc/krb.realms

        kdc_timesync = 1

        ccache_type = 4

        forwardable = true

        proxiable = true

[realms]

        TELECOM.ECE.NTUA.GR = {

                kdc = plrf21.telecom.ece.ntua.gr
```

```
                admin_server = plrf21.telecom.ece.ntua.gr
        }
[domain_realm]
        .telecom.ece.ntua.gr = TELECOM.ECE.NTUA.GR
        telecom.ece.ntua.gr = TELECOM.ECE.NTUA.GR
[login]
        krb4_convert = true
        krb4_get_tickets = false
[logging]
        kdc = FILE:/var/log/kerberos/krb5kdc.log
        admin_server = FILE:/var/log/kerberos/kadmin.log
        default = FILE:/var/log/kerberos/krb5lib.log
```

B.2 MongoDB Enterprise v3.4 σε Ubuntu 5.4.0-6ubuntu1~16.04.4

Για την εγκατάσταση της MongoDB Enterprise ακολουθούμε την παρακάτω διαδικασία

```
>sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
0C49F3730359A14518585931BC711F9BA15703C6
```

```
>echo "deb [ arch=amd64,arm64,ppc64el,s390x ]
http://repo.mongodb.com/apt/ubuntu xenial/mongodb-enterprise/3.4
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-enterprise.list
```

```
>sudo apt-get update
>sudo apt-get install -y mongodb-enterprise
>sudo service mongod start
```

Για την εκκίνηση του MongoDB server με τις παραμέτρους που θέλουμε πληκτρολογούμε

```
>env KRB5_KTNAME=/etc/krb5.keytab \
```

```
KRB5_TRACE=/logs/mongodb-kerberos.log \
mongod --config /etc/mongod.conf
```

Απαραίτητο αρχείο που περιέχει όλες τις ρυθμίσεις και παραμέτρους της MongoDB

/etc/mongod.conf

```
# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/
# Where and how to store data.
storage:
  dbPath: /data/newdb
  journal:
    enabled: true
#  engine:
#  mmapv1:
#  wiredTiger:

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# network interfaces
net:
  port: 27017
  bindIp: 0.0.0.0
  ssl:
     mode: requireSSL
     PEMKeyFile: /etc/ssl/mongodb.pem
```

```
#setParameters
setParameter:
  authenticationMechanisms: GSSAPI
#processManagement:

#security:
security:
  authorization: enabled
  enableEncryption: true
  encryptionKeyFile: /home/ntua/mongodb-keyfile

#operationProfiling:
replication:
  replSetName: rsHealthCare

#sharding:
## Enterprise-Only Options:
auditLog:
  destination: syslog

#snmp:
```

B.3 Δημιουργία σχήματος της βάσης και προσθήκη χρηστών

Για τη δημιουργία χρηστών της βάσης δεδομένων σε συνεργασία με την πιστοποίηση από τον Κέρβερο (πρέπει τα ονόματα χρηστών να είναι κοινά), ακολουθούμε την παρακάτω διαδικασία για κάθε χρήστη. Για αυτή τη διαδικασία χρειάζεται να συνδεθούμε στη βάση με έναν super user ή να έχουμε απενεργοποιήσει την υπηρεσία πιστοποίησης του Κέρβερου:

```
>mongo --host plrf20.telecom.ece.ntua.gr
MongoDB shell version v3.4.5
connecting to: mongodb://plrf20.telecom.ece.ntua.gr:27017/
```

```
MongoDB server version: 3.4.4

MongoDB Enterprise > use $external
MongoDB Enterprise > db.createUser(
                         {
                           user: "Vrettos@TELECOM.ECE.NTUA.GR ",
                           roles: [ { role: "readWrite", db: "HealthCare" } ]
                         }
                       )
```

Για σύνδεση με τη βάση από οποιαδήποτε τοποθεσία, αφού έχουμε λάβει τα απαραίτητα διαπιστευτήρια με την εντολή **kinit**, συνδεόμαστε με την εντολή:

```
>mongo --host plrf20.telecom.ece.ntua.gr --authenticationMechanism=GSSAPI
--authenticationDatabase='$external'--username Vrettos@TELECOM.ECE.NTUA.GR
```

Παρακάτω παρουσιάζονται κάποια παραδείγματα από τα δεδομένα που αποθηκεύονται στη βάση για να γίνει κατανοητό το σχήμα που χρησιμοποιήθηκε για την αποθήκευσή τους.

```
MongoDB Enterprise > use HealthCare
switched to db HealthCare
MongoDB Enterprise > show collections
BasicInfo
SignatureKeys
MongoDB Enterprise > db.SignatureKeys.find().pretty();
{
        "_id" : ObjectId("5931484d5fb49b18a0f1baa7"),
        "Name" : "Vrettos",
        "PublicKey" :
BinData(0,"MIIBuDCCASwGByqGSM44BAEwggEfAoGBAP1/U4EddRIpUt9KnC7s5Of2EbdSPO9
EAMMeP4C2USZpRV1AIlH7WT2NWPq/xfW6MPbLm1Vs14E7gB00b/JmYLdrmVClpJ+f6AR7ECLCT
7up1/63xhv4O1fnxqimFQ8E+4P208UewwI1VBNaFpEy9nXzrith1yrv8iIDGZ3RSAHHAhUAl2B
QjxUjC8yykrmCouuEC/BYHPUCgYEA9+GghdabPd7LvKtcNrhXuXmUr7v6OuqC+VdMCz0HgmdRW
VeOutRZT+ZxBxCBgLRJFnEj6EwoFhO3zwkyjMim4TwWeotUfI0o4KOuHiuzpnWRbqN/C/ohNWL
x+2J6ASQ7zKTxvqhRkImog9/hWuWfBpKLZl6Ae1UlZAFMO/7PSSoDgYUAAoGBALzEXd6cEQqIr
```

```
R9pi0QNYFwJXN22auKwsd1NSCeXRM032NUMZ+qeCRKNr9iQn+Q/no8+3N7A+vB6OSgJkLfU9DA
7x7qPQ14OV4qNTY8UbotIIPd8dep/brItIZy7CmZE6/cWA1hHt2p1kqErDa5+My50zNxLiyfiY
/HoCMAoZRtu")

}


MongoDB Enterprise > db.BasicInfo.find().pretty();

{
        "_id" : ObjectId("596b27f302e83f113466c922"),
        "AFM" : "158015800",
        "_AFM" : [
BinData(0,"MCwCFB1B+wQm4gMtAtKl0rD+Id+GIFKEAhRYWP/cRDKC28z0iXCd9mcbQm3y6w=
="),"NoLee"],


        "Name" : "Emmanouil",
        "_Name" : [
BinData(0,"MCwCFBpnElECVGf0CkTm/pSyOdDw+I0SAhQQDCkRyHZztzQv6yJ0utptTFNQbw=
="), "Vrettos"],


        "Surname" : "Pedarakhs",
        "_Surname" : [
BinData(0,"MCwCFG7mbFsUK1C22EPw2pfxopMsFjdaAhQ8ukyQ20yJzT6kFxKSjKFgdhEuug=
="),"NoLee"],


        "Age" : "18-24",
        "_Age" :
[BinData(0,"MCwCFAPUISCgDOb8fdYy0pVDY31dVBssAhRSdSw2hRXpyQ8+6hxH5n0o11Mnog
=="),"NoLee" ],


        "Illness" :
BinData(0,"i4eVJh9zVSOwIWPqegK6Xjs6PuzUOda49B1ppksjYn6k3SIi6leURxvfm0oZa6Q
sPrIqIzt/keO5tfmGci48SIB8CpQ+CBCNWgI0KTEzJgMNp/ij/UW6RSEK203c/faxOViYPMW/M
nYMkikKPPKROdqmNNwLjGvO2iokLp/quRU="),
        "_Illness" :
[BinData(0,"MCwCFFhZPLTSavZ9ytmfQUUFCHEksYykAhR6YGfh2VvU7NiGctG2bq86pr4uqg
=="),"NoLee"],


        "Public Key" :
BinData(0,"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCkWWPkvNavUTwh+4Pwi2ox1ND
Y+7wCXml+CTl2hixJIN8QPZUwUsegxRvyxpwsPXeMBSxzLiQLmzZhpvqo9nSLsIsrjImblcyZO
cFinVRzda1e+2lG87DBskoQaNkbPAZ9Qg/W0mCdFX28cy/vYQJ8+ZVS9GGog3317gGieP+uVwI
DAQAB"),
        "_Public Key" :
[BinData(0,"MCwCFBJXLVe6FKOKd2JOuIIGxsQj3X/HAhRSCG9u4mvgyD7qIaaE+xoqQE+MoQ
=="), "Vrettos"]
}
```