Διπλωματικη Εργασια

(Diploma Thesis)

Κωνσταντίνου Μπίτσικα

Α.Μ.: 09104225

Κρυπτογραφηση Μηνυματων SMS σε Android

Επιβλέπων Καθηγητής:

Πέτρος Στεφανέας, Επίκουρος Καθηγητής

Επιτροπή:

Χρήστος Κουκουβίνος, Καθηγητής

Πέτρος Καβάσαλης, Αναπληρωτής Καθηγητής Παν.Αιγαίου

Σεπτέμβριος 2016

Abstract

Στις μέρες μας, με τον ταχύτατο ρυθμό ανάπτυξης της τεχνολογίας και την καθημερινή χρήση των κινητών, το SMS (Short Message Service) έχει εδραιωθεί ως ένας από τους κλασσικότερους τρόπους επικοινωνίας σε προσωπικό και επαγγελματικό επίπεδο. Παρόλα αυτά, οι υπάρχουσες εφαρμογές δεν παρέχουν ιδιαίτερη προστασία στα δεδομένα και τις επικοινωνίες, ενώ και το δίκτυο GSM που μεταφέρει τα μηνύματα μεταξύ των συσκευών έχει αρκετά τρωτά σημεία που αποδυναμώνουν την ασφάλεια των επικοινωνιών. Στην εργασία μας θα περιγράψουμε τον τρόπο λειτουργίας του δικτύου, θα αναδείξουμε τα τρωτά του σημεία και θα περιγράψουμε τις συνηθέστερες μορφές επιθέσεων. Τέλος, το κυρίως μέρος της εργασίας αφιερώνεται στην εφαρμογή που αναπτύξαμε για τη λύση αυτών των προβλημάτων. Η εφαρμογή μας προσφέρει κρυπτογράφηση των μηνυμάτων SMS από τέλος προς τέλος, σε συνδυασμό με ένα γνωστό και φιλικό προς τον χρήστη περιβάλλον.

Περιεχόμενα

Κατάλογος Σχημάτων

1. Εισαγωγή

Στις μέρες μας παρατηρείται μια στροφή από τους προσωπικούς υπολογιστές (desktops) και τα laptops προς τα κινητά τηλέφωνα (smartphones) και τα tablets και αυτό δεν είναι περίεργο. Τα smartphones και τα tablets είναι ευκολότερα στη μεταφορά από τα laptops, είναι μοντέρνα, ακόμη λειτουργούν και ως καταξίωση 'lifestyle'. Περιέχουν ενδιαφέροντα και χρήσιμα χαρακτηριστικά και εφαρμογές οι οποίες ελκύουν τον κόσμο, όπως οθόνες αφής, κινητά δίκτυα, WiFi, Bluetooth, GPS πλοήγηση, φωτογραφική μηχανή, μερικές εκ των οποίων είναι τελευταίας τεχνολογίας, αναγνώριση φωνής και αναπαραγωγή μουσικής καθώς και χωρητικότητα που μπορεί να φτάσει αρκετά gigabytes, όλα αυτά χωρίς να υστερούν σε υπολογιστική δύναμη. Επίσης, είναι ευκολότερα στη χρήση τους από τα desktops και τα laptops για τον μη επαγγελματία χρήστη.

Στον τομέα των κινητών τηλεφώνων και των tablet, υπάρχουν τρεις εταιρίες που πρωτοπορούν, η Google, η Apple και η Microsoft, αν και αυτήν τη στιγμή η Microsoft έρχεται τρίτη με μεγάλη διαφορά από τις άλλες δύο. Οι αντίστοιχες πλατφόρμες τους είναι το Android, το iOS και τα Windows. Η Google έχει απελευθερώσει προς το κοινό το μεγαλύτερο μέρος του λογισμικού του Android και έτσι είναι πολύ ευκολότερο σε προγραμματιστές να το χρησιμοποιήσουν και να δημιουργήσουν τις δικές τους εφαρμογές από ότι το εσώκλειστο iOS.

Η εφαρμογή που παρουσιάζεται στην παρούσα εργασία είναι βασισμένη στο λειτουργικό σύστημα (operating system, OS) Android, επομένως θα επικεντρωθούμε σε αυτό και θα αναφέρουμε προς το παρόν μερικές γενικές πληροφορίες (περισσότερες πληροφορίες στο κεφάλαιο 2.3).

Το Android κυκλοφόρησε επίσημα στην αγορά τον Σεπτέμβριο του 2008 στο κινητό τηλέφωνο HTC Dream με την έκδοση 1.0 από την Google, σε συνεργασία με αρκετές μεγάλες εταιρίες όπως την Intel, την Dell, την Sony, την Nvidia, την LG Electronics και την HTC, με τις οποίες σχηματίζουν την Open Handset Alliance (OHA), μία συμμαχία

με σκοπό την ανάπτυξη ανοιχτού λογισμικού για φορητές συσκευές (open standard for mobile devices). Στην αρχική κυκλοφορία του και στις εκδόσεις πριν τη 2.0, το Android δεν ήταν συμβατό με άλλες πλατφόρμες όπως tablets και PDAs. Αυτό έγινε με τη συνεχή υποστήριξη της Google και γενικότερα της OHA. Το Android είναι ένα λειτουργικό σύστημα βασισμένο σε Linux και αποτελεί φυσικά το λειτουργικό σύστημα που χρησιμοποιούνται στα προϊόντα των εταιριών-μέλη της OHA, τα οποία αριθμούν πλέον σε 84. Το 2012 το Android ηγούταν της αγοράς για λειτουργικά συστήματα κινητών συσκευών κρατώντας το 69.3% και συνέχισε να μεγαλώνει σε ένα κυριαρχικό 82.9% της αγοράς το 2015, σύμφωνα με τα δεδομένα της International Data Corporation (IDC) [2].

Υπάρχει πληθώρα εφαρμογών (applications) Android που είναι διαθέσιμες αυτήν τη στιγμή, από IM (instant messaging) όπως το WhatsApp και το Viber, σε web banking, σε SMS (short message service), σε χάρτες όπως το κλασσικό Google maps, σε αναγνώριση προσώπων ή ακόμα και αναγνώριση τραγουδιών όπως το Shazam, καθώς και αμέτρητα παιχνίδια. Φυσικά κάθε χρήστης δημιουργεί μια πολύ προσωπική εμπειρία χρησιμοποιώντας τις εφαρμογές που ο ίδιος επιλέγει και με τον τρόπο που ο ίδιος επιλέγει και πολλές από τις εφαρμογές αυτές διατηρούν δεδομένα και ιδιωτικές πληροφορίες του χρήστη, καθιστώντας έτσι τις συσκευές εξαιρετικά προσωπικές και σημαντικές για τον ιδιοκτήτη τους. Αυτός είναι και ο λόγος για τον οποίο υπάρχει μεγάλη ανάγκη για προστασία όσον αφορά τις εφαρμογές που αποθηκεύουν προσωπικά δεδομένα.

Εξαιτίας της ραγδαίας εξάπλωσης των φορητών συσκευών και της ανάπτυξης της τεχνολογίας οι περισσότεροι χρήστες έχουν τη δυνατότητα να συνδεθούν στο διαδίκτυο μέσω των τηλεφώνων και tablet τους όποτε θέλουν. Αυτό έχει επηρεάσει τη χρήση του SMS καθώς όλο και περισσότεροι χρήστες τείνουν να χρησιμοποιούν προγράμματα ανταλλαγής άμεσων μηνυμάτων (instant messaging, IM), μιας και δεν έχουν όριο στον αριθμό χαρακτήρων που μπορούν να στείλουν σε ένα μήνυμα και επιπλέον δεν κοστίζουν τίποτα. Παρόλα αυτά το SMS είναι ακόμα πάρα πολύ διαδεδομένο και εξακολουθεί να είναι από τους συνηθέστερους τρόπους επικοινωνίας από άνθρωπο σε άνθρωπο τόσο για κοινωνικούς όσο, ακόμα περισσότερο και για επαγγελματικούς σκοπούς. Είναι γεγονός πως αυτή τη στιγμή ο

όγκος μηνυμάτων από IM εφαρμογές είναι μεγαλύτερος από αυτόν των SMS, ωστόσο οι IM εφαρμογές χρειάζονται smartphone ή tablet για να τρέξουν, ενώ το SMS λειτουργεί σε όλα τα κινητά τηλέφωνα, ακόμα και σε αυτά πολύ παλιάς τεχνολογίας. Έτσι, παρόλη την εξάπλωση των smartphones το SMS καλύπτει σχεδόν όλο τον κόσμο και όσο εύκολο και να είναι να συνδεθεί κάποιος πλέον στο διαδίκτυο είναι ακόμα ευκολότερο να στείλει ένα μήνυμα μέσω του ασύρματου δικτύου της κινητής τηλεφωνίας.

2. Περιγραφή της Εφαρμογής

Το θέμα αυτής της εργασίας είναι η κρυπτογράφηση των μηνυμάτων SMS (Short Messaging Service) έτσι ώστε να μπορούν να διαβαστούν μόνο από τον παραλήπτη τους ή για την ακρίβεια μόνο από τον άνθρωπο για τον οποίο προορίζονται.

Η πλατφόρμα Android επιλέχτηκε για την εφαρμογή διότι όπως προαναφέραμε το Android είναι πολύ πιο διαδεδομένο από τις υπόλοιπες πλατφόρμες, είναι πιο προσιτό στους προγραμματιστές επειδή το μεγαλύτερο κομμάτι του είναι Open Sourced, ενώ οι εφαρμογές που τρέχουν σε Android είναι γραμμένες σε Java η οποία επίσης χρησιμοποιείται ευρύτατα. Η Java class η οποία εμπεριέχεται στο πρόγραμμα και που εκτελεί την κρυπτογράφηση, μπορεί να χρησιμοποιηθεί από άλλους προγραμματιστές Android για την κρυπτογράφηση οποιουδήποτε είδους κειμένου δημιουργώντας έτσι πιο ασφαλείς εφαρμογές.

2.1. Περιγραφή SMS και Δικτύου GSM

Τα SMS μεταδίδονται μέσω του δικτύου GSM, επομένως θα περιγράψουμε παρακάτω τα στοιχεία τους, ώστε να αναδείξουμε τα αδύνατα σημεία τους, τα οποία είναι αυτά που ενδιαφερόμαστε να καλύψουμε με την εφαρμογή μας.

2.1.1. SMS-Short Message Service (Υπηρεσία Σύντομου Μηνύματος)

Η ιδέα του SMS (Short Messaging Service) συνελήφθη για πρώτη φορά το 1984 στη Γαλλογερμανική εταιρία GSM από τους μηχανικούς Friedhelm Hillebrand και Bernard Ghillebaert. Η βασική ιδέα του SMS ήταν να χρησιμοποιήσει το δίκτυο του τηλεφώνου, το οποίο δεν είχε ιδιαίτερη κίνηση, για να μεταδίδει μηνύματα. Με αυτόν τον τρόπο θα χρησιμοποιούσε αχρησιμοποίητους πόρους του συστήματος ώστε να μεταφέρει μηνύματα με ελάχιστο κόστος. Έπρεπε όμως να περιοριστεί το μήκος των μηνυμάτων ώστε να μπορεί να μεταδοθεί από τις υπάρχουσες μορφές σημάτων.

Ωστόσο το πρώτο sms εστάλη το 1992 από τον προσωπικό υπολογιστή ενός προγραμματιστή της τότε SEMA Group (σήμερα Mavenir Systems) στη Vodafone στη Μεγάλη Βρετανία. Το μήνυμα ήταν «Καλά Χριστούγεννα».

Από την επόμενη χρονιά, το 1993, πρώτη η Σουηδική Nokia αρχίζει να προσφέρει στην αγορά συσκευές που υποστηρίζουν sms, ενώ το 1999 τα sms μπορούν πλέον να μεταδοθούν μέσω διαφορετικών δικτύων.

Το 2015 υπολογίζεται πως σε ολόκληρο τον κόσμο απεστάλησαν περισσότερα από 8,3 τρισεκατομμύρια sms.

Το SMS (Short Message Service) επιτρέπει τη μεταφορά μηνυμάτων μέσω ασύρματων δικτύων. Ένα τυπικό SMS αποτελείται από 140 bytes, δηλαδή 160 7-bit χαρακτήρες ή 140 8-bit χαρακτήρες, το οποίο τις περισσότερες φορές μεταφράζεται σε ένα μήνυμα 160 χαρακτήρων. Τα μηνύματα ταξιδεύουν από και προς τις συσκευές που επικοινωνούν. Η υπηρεσία SMS περιέχεται μέσα στο Global Standard for Mobile (GSM) δίκτυο, που είναι το επικρατέστερο ψηφιακό ασύρματο δίκτυο. Κάθε δίκτυο SMS περιέχει το Short Messaging Message Center (SMSC), τον μηχανισμό μέσα από τον οποίο τα μηνύματα αποθηκεύονται και προωθούνται στους συνδρομητές, ανεξάρτητα αν αυτοί είναι απενεργοποιημένοι τη στιγμή της αρχικής εκπομπής, απλά προωθώντας το μήνυμα όταν ο παραλήπτης είναι ξανά διαθέσιμος. Σε αυτήν την εργασία μελετάμε την ασφαλή μεταφορά μηνυμάτων από σημείο σε σημείο (point-to-point messaging), δηλαδή την μεταφορά μηνυμάτων μεταξύ δύο ατομικών συσκευών.

2.1.2. GSM – Global System for Mobile communication (Παγκόσμιο Σύστημα Κινητών Επικοινωνιών)

Το GSM είναι το καθιερωμένο ψηφιακό δίκτυο κινητής τηλεφωνίας, το οποίο είναι διαμορφωμένο σε κυψελοειδή μορφή δεύτερης γενιάς (2G). Αυτό σημαίνει ουσιαστικά ότι η περιοχή της εμβέλειας του δικτύου είναι χωρισμένη σε περιοχές (κυψέλες), κάθε μία εκ των οποίων εξυπηρετείται από ένα Σταθμό Βάσης (Base Station). Το GSM παρέχει υπηρεσίες όπως SMS, κρυπτογραφημένες συνομιλίες,

εκτροπή κλήσεων, αναγνώριση κλήσεων, απόκρυψη κλήσεων, μεταφορά δεδομένων και roaming (περιήγηση).

2.1.2.1. Συντομογραφίες

Συντ.	Αγγλικός Όρος	Ελληνικός Όρος	Λειτουργία
MAP	Mobile Application Part		Πρωτόκολλο SS7 που προσθέτει μια βαθμίδα στην εφαρμογή μέσα από την οποία παρέχει πρόσβαση σε διάφορες οντότητες του δικτύου GSM.
MS	Mobile Station	Κινητός Σταθμός	Κινητός Σταθμός είναι οποιαδήποτε συσκευή περιλαμβάνει πομπό-δέκτη, κεραία, οθόνη και κάρτα SIM.
BSS	Base Station Subsystem	Υποσύστημα Σταθμού Βάσης	Οντότητα που περιέχεται μέσα στο δίκτυο GSM. Διαχειρίζεται την κίνηση ανάμεσα στους κινητούς σταθμούς και το Υποσύστημα Δικτύου Μεταγωγής (Network Switching System, NSS).
BTS	Base Transceiver Station	Βασικός Σταθμός Πομπού-Δέκτη	Οντότητα που περιέχεται μέσα στο δίκτυο GSM, η οποία λαμβάνει τα μηνύματα που στέλνει ένας Κινητός Σταθμός και τα στέλνει στο Βασικό Σταθμό Ελέγχου (Base Station Controller, BSC) ή δέχεται μηνύματα από το BSC και τα στέλνει στον Κινητό Σταθμό που προορίζονται.
BSC	Base Station Controller	Βασικός Σταθμός Ελέγχου	Διαχειρίζεται πολλά BTS και προωθεί τα μηνύματα στο Κέντρο Διανομής (Mobile Switching Center, MSC).
NSS	Network Switching Subsystem	Υποσύστημα Δικτύου Διανομής	Μέρος του δικτύου GSM και διαχειρίζεται τις συνδέσεις μεταξύ των Κινητών Σταθμών.

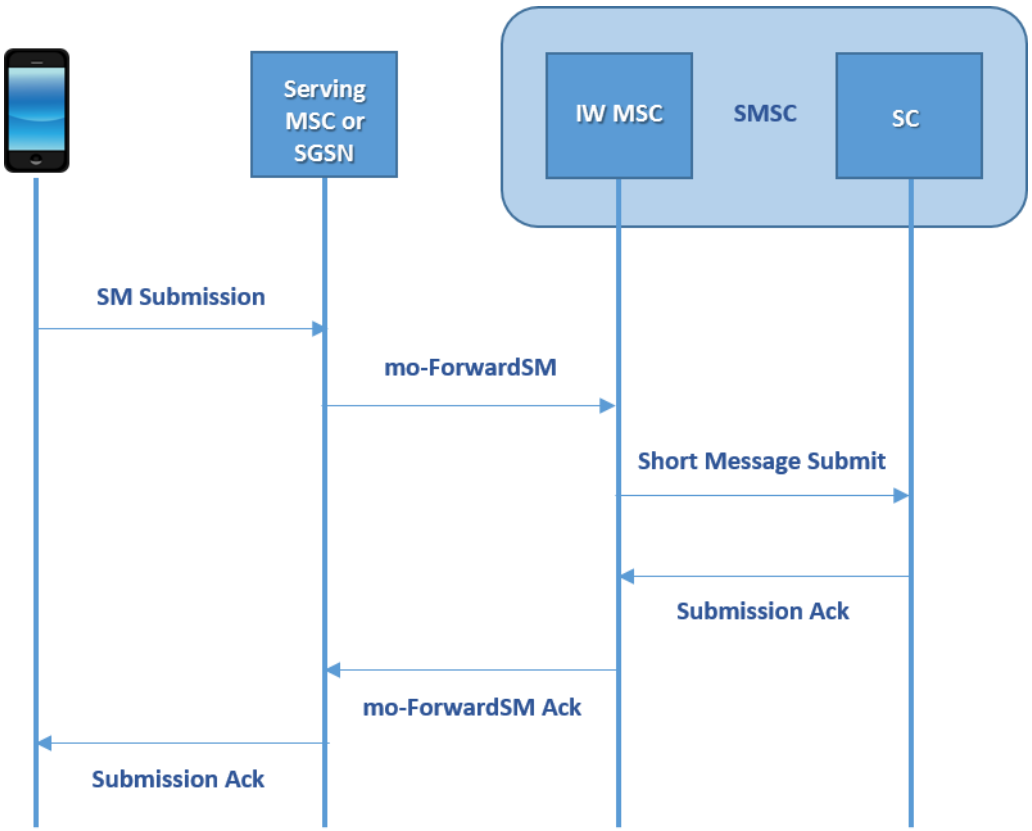
MSC	Mobile Switching Station	Κέντρο Διανομής	Οντότητα που περιέχεται μέσα στο δίκτυο GSM. Καθορίζει τη διαδρομή προς τον Κινητό Σταθμό του παραλήπτη ή προς το SMSC (Short Message Service Center) για SMS και τηλεφωνήματα.
SMSC	Short Message Service Center		Οντότητα που περιέχεται στο δίκτυο GSM. Αποθηκεύει και προωθεί μηνύματα SMS προς τον Κινητό Σταθμό του παραλήπτη, θέτοντας σε προτεραιότητα τους συνδρομητές που βρίσκονται αυτήν τη στιγμή στο ίδιο δίκτυο.
HLR	Home Location Register	Τοπικό Κέντρο Εγγραφής	Βάση δεδομένων που περιέχει το δίκτυο GSM. Περιέχει τις τοποθεσίες των συνδρομητών του δικτύου καθώς και τα στοιχεία του ιδιοκτήτη και τις υπηρεσίες στις οποίες είναι εγγεγραμμένος.
VLR	Visitor Location Register	Εικονικό Κέντρο Εγγραφής	Βάση δεδομένων στην οποία έχει πρόσβαση το BSS. Περιέχει τις πληροφορίες που έχει αποκομίσει από το HLR για τους συνδρομητές που βρίσκονται αυτήν τη στιγμή μέσα στην εμβέλεια του BSS.
APDU	Application Protocol Data Unit		Περιέχουν πληροφορίες που μεταφέρονται ως μονάδες μεταξύ των οντοτήτων του δικτύου. Στην περίπτωση του SMS περιέχει το κείμενο του μηνύματος, τη διεύθυνση του παραλήπτη Κινητού Σταθμού και τη διεύθυνση του SMSC.

2.1.2.2. Διαδικασίες Δικτύου

Όταν ένας Κινητός Σταθμός στέλνει ένα μήνυμα σε έναν άλλο, κατά τη διάρκεια του ταξιδιού του και μέχρι να φτάσει στον παραλήπτη αυτό το μήνυμα περνάει από αρκετά στάδια. Σε αυτό το χρονικό διάστημα οι ακόλουθες διαδικασίες λαμβάνουν χώρα:

2.1.2.2.1. Mobile Originated short message service transfer

Ο συνδρομητής στέλνει το μήνυμα από έναν Κινητό Σταθμό. Το μήνυμα αυτό αρχικά παραδίδεται από τον Κινητό Σταθμό στο Κέντρο Διανομής (MSC). Αυτή η διαδικασία γίνεται από το Υποσύστημα Σταθμού Βάσης (BSS) το οποίο εκπέμπει τα Application Protocol Data Units (APDU) του GSM μέσω του air interface από τον αρχικό Κινητό Σταθμό στο Υποσύστημα Δικτύου Διανομής (NSS). Το Συνεργατικό Κέντρο Διανομής (interworking MSC), που είναι κομμάτι του GSM, είναι συνδεδεμένο με το Short Message Service Center (SMSC) που είναι και το μέρος στο οποίο τα μηνύματα αποθηκεύονται ώστε να προωθηθούν αργότερα στον παραλήπτη Κινητό Σταθμό. Το Interworking SMC παραλαμβάνει μια ένδειξη επιτυχίας ή αποτυχίας από το SMSC και με τη σειρά του στέλνει ένδειξη πίσω στον αρχικό συνδρομητή.



2.1 MO short message service

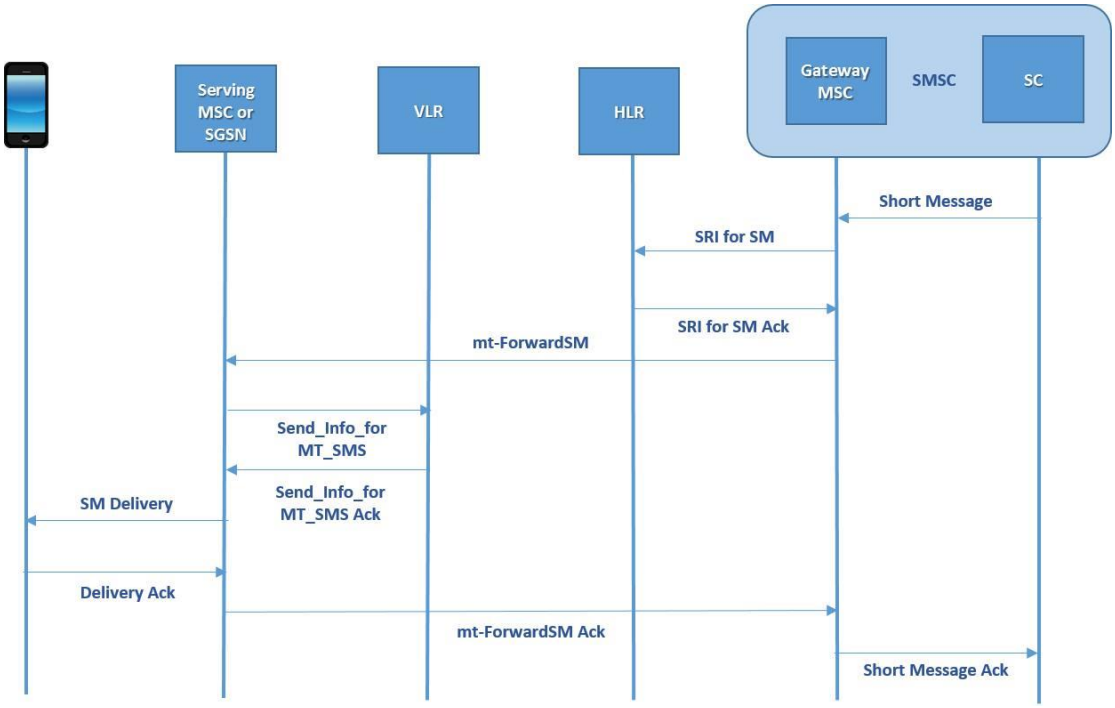
Μία αναπαράσταση βήμα προς βήμα έχει ως εξής:

1. Το επίπεδο μεταφοράς μηνυμάτων (Short Message Transfer Layer) του Κινητού Σταθμού του συνδρομητή που θέλει να στείλει το μήνυμα, στέλνει το SMS στο VMSC.
2. Το VMSC στέλνει την εντολή MAP_MO_FORWARD_SHORT_MESSAGE στο interworking MSC στο SMSC που έχει στην εμβέλειά του τον τελικό Κινητό Σταθμό.
3. Το MSC στέλνει τα APDU του μηνύματος στο κέντρο εξυπηρέτησης του SMSC.
4. Το κέντρο εξυπηρέτησης στέλνει μια αναφορά διανομής μαζί με ένα MAP_FORWARD_SHORT_MESSGE_ack μήνυμα στο interworking MSC.
5. Το interworking MSC στέλνει SMS-STATUS-REPORT στο MSC.
6. Το MSC στέλνει SMS-STATUS-REPORT στο επίπεδο μεταφοράς του Κινητού Σταθμού του παραλήπτη.

2.1.2.2.2. Mobile Terminated short message service transfer

Το Short Message Service Center (SMSC) πρέπει κάποια στιγμή να στείλει το μήνυμα στον προορισμό του. Τότε στέλνει τα APDU που περιέχουν όλες τις απαιτούμενες πληροφορίες, μεταξύ άλλων το κείμενο και τη διεύθυνση του παραλήπτη, στο Εισαγωγικό Κέντρο Διανομής (Gateway MSC). Όταν το GMSC λάβει το μήνυμα, πρέπει να επαληθεύσει τη διεύθυνση του παραλήπτη ώστε να στείλει σωστά το μήνυμα. Για να το κάνει αυτό, προμηθεύεται τις πληροφορίες δρομολόγησης από το Τοπικό Κέντρο Εγγραφής (HLR) καλώντας ένα πακέτο του Mobile Application Part (MAP) το MAP_SEND_ROUTING_INFO_FOR_SM που στέλνει μία εντολή στο Τοπικό Κέντρο Εγγραφής (HLR) στο οποίο είναι εγγεγραμμένος ο Κινητός Σταθμός του παραλήπτη, ώστε να πάρει τις πληροφορίες για την τρέχουσα τοποθεσία του. Το HLR εκτελεί μια έρευνα στη βάση δεδομένων του για να αποκτήσει την τοποθεσία του παραλήπτη και στη συνέχεια στέλνει τις πληροφορίες στο GSMC, αν ο παραλήπτης είναι διαθέσιμος να παραλάβει το μήνυμα ή, στην αντίθετη περίπτωση, μία αποτυχία. Με αυτές τις πληροφορίες το GMSC θα προσπαθήσει να στείλει το μήνυμα στη διεύθυνση του προορισμού του, την οποία έχει προμηθευτεί

από το HLR, καλώντας την υπηρεσία MAP_MT_FORWARD_SHORT_MESSAGE. Το Εικονικό Κέντρο Εγγραφής (visitor location register, VLR) ξεκινά μια διαδικασία ειδοποίησης. Αν η τοποθεσία του παραλήπτη είναι γνωστή στο VLR, τότε στέλνει το μήνυμα MAP_PAGE σε αυτήν τη διεύθυνση, διαφορετικά στέλνει το μήνυμα MAP_SEARCH_FOR_SUBSCRIBER σε όλες τις τοποθεσίες που επικοινωνούν με το Κέντρο Διανομής, με σκοπό να αποκτήσει τη τοποθεσία του παραλήπτη και μετά στέλνει το MAP_PAGE. Το Κέντρο Διανομής στο οποίο ο παραλήπτης είναι εγγεγραμμένος τον ειδοποιεί και στέλνει το μήνυμα MAP_PROCESS_ACCES_REQUEST πίσω στο VLR μόλις επιτύχει. Τότε το VLR στέλνει το μήνυμα MAP_SEND_INFO_FOR_MT_SMS_ack στο Κέντρο Διανομής και έτσι του επιτρέπεται να προχωρήσει με την προώθηση του μηνύματος στον προορισμό του. Το κέντρο διανομής στο οποίο είναι εγγεγραμμένος ο παραλήπτης λαμβάνει ένα SMS-STATUS-REPORT από τον Κινητό Σταθμό του παραλήπτη και μετά στέλνει ένα μήνυμα MAP_FORWARD_SHORT_MESSAGE_ack στο GSMC στο οποίο περιλαμβάνει την αναφορά αυτή, ώστε να επικυρώσει ότι το μήνυμα εστάλη. Στη συνέχεια το GSMC προσδιορίζει αν ο παραλήπτης έλαβε το μήνυμα ή δεν ήταν διαθέσιμος. Αυτή η πληροφορία τέλος προωθείται στο SMSC.



2.2 MT short message service transfer

Μία βήμα προς βήμα αναπαράσταση έχει ως εξής

1. Το SMSC στέλνει ένα αίτημα MAP_SEND_ROUTING_INFO_FOR_SM στο HLR.
2. Το HLR στέλνει το μήνυμα MAP_SEND_ROUTING_INFO_FOR_SM_ack στο GMSC κομμάτι του SMSC.
3. Το GMSC στέλνει MAP_FORWARD_SHORT_MESSAGE στο MSC που βρίσκεται ο παραλήπτης.
4. Το MSC στέλνει MAP_SEND_INFO_FOR_MT_SMS στο VLR.
5. Το VLR στέλνει MAP_PAGE στον παραλήπτη ή MAP_SEARCH_FOR_SUBSCRIBER στο MSC.
6. Το MSC στέλνει MAP_PROCESS_ACCESS_REQUEST στο VLR.
7. Το VLR στέλνει MAP_SEND_INFRO_FOR_MT_SMS_ack στο MSC.
8. Ο κινητός σταθμός του παραλήπτη στέλνει SMS_STATUS_REPORT στο MSC.
9. Το MSC στέλνει MAP_FORWARD_SHORT_MESSAGE_ack στο SMSC.

2.1.2.3. Προβλήματα Ασφαλείας του Δικτύου GSM

Όπως αναφέρθηκε προηγουμένως το SMS είναι πολύ διαδεδομένο, όμως το δίκτυο GSM που διανέμει τα μηνύματα έχει μερικά προβλήματα όσον αφορά την προστασία τους.

[3] Η αρχιτεκτονική του δικτύου GSM είχε σχεδιαστεί ώστε να παρέχει προστασία στα δεδομένα των χρηστών. Οι υπηρεσίες αυτού του είδους συμπεριλαμβάνουν πιστοποίηση των χρηστών, εμπιστευτικότητα των δεδομένων των χρηστών, ανωνυμία της ταυτότητάς τους και τη χρήση του Subscriber Identity Module (SIM) για λόγους προστασίας. Η κάρτα SIM είναι μία smart card εξοπλισμένη με τις εφαρμογές του GSM και είναι απαραίτητη για τη λειτουργία όλων των κινητών τηλεφώνων. Η SIM αποθηκεύει όλες τις πληροφορίες που χρειάζεται ώστε να αποκτήσει πρόσβαση στον λογαριασμό του συνδρομητή στο δίκτυο GSM, κάτι που την καθιστά ανεκτίμητη στον ιδιοκτήτη της και ταυτόχρονα επικίνδυνη για την ασφάλειά του σε περίπτωση που κάποιος εκτός του ιδίου αποκτήσει πρόσβαση. Έτσι η κάρτα SIM έρχεται με αρκετές ιδιότητες για την ασφάλεια των δεδομένων:

1. Το International Mobile Subscriber Identity (IMSI) που είναι ένας μοναδικός κωδικός μέχρι 15 ψηφία, ο οποίος ανήκει σε κάθε συνδρομητή.
2. Το Individual Subscriber Identification Key (Ki) που είναι ένα 128-bit κρυπτογραφικό κλειδί το οποίο χρησιμοποιείται για να δημιουργεί κλειδιά συνεδρίας που πιστοποιούν τους χρήστες στο δίκτυο GSM.
3. Το πλέον γνωστό Personal Identification Number (PIN) το οποίο είναι συνήθως ένας 4-ψήφιος κωδικός που χρειάζεται να εισαχθεί για να αποκτήσει ο χρήστης πρόσβαση στο κινητό τηλέφωνο και
4. Το PIN Unlock (PUK) το οποίο είναι ένας κωδικός που ζητείται για να ξεκλειδώσει την κάρτα SIM αν ο κωδικός PIN έχει εισαχθεί λάθος αρκετές φορές (συνήθως 3).

Παρόλα αυτά το δίκτυο GSM έχει τρωτά σημεία. Επειδή είναι ασύρματο χρησιμοποιεί το air interface για τη μετάδοση δεδομένων, επίσης δεν έχει αμοιβαία πιστοποίηση μεταξύ χρηστών για τα μηνύματα, δηλαδή αυτός που στέλνει το μήνυμα δεν πιστοποιεί ότι το μήνυμα έφτασε στο σωστό προορισμό, τη πιστοποίηση την κάνει μόνο το δίκτυο. Αυτό κάνει το δίκτυο πολύ τρωτό σε man in

the middle επιθέσεις όπου ένας επιτιθέμενος με λίγη παραποίηση στον κώδικα του δικτύου μπορεί να αναπαραστήσει έναν νόμιμο χρήστη και να διαβάσει ή να παραποιήσει την επικοινωνία μεταξύ δύο χρηστών. Όταν δημιουργήθηκε αρχικά το GSM αυτό δεν ήταν σημαντικό πρόβλημα γιατί ο εξοπλισμός που χρειαζόταν για να εκτελέσει κάποιος αυτού του είδους την επίθεση ήταν πολύ ακριβός, ωστόσο στις μέρες μας είναι προσιτός στον περισσότερο κόσμο. Εκτός από την έλλειψη αμοιβαίας πιστοποίησης χρηστών, το GSM δεν παρέχει ούτε αμοιβαία πιστοποίηση μεταξύ του BSS και του κινητού σταθμού που στέλνει το μήνυμα, με αποτέλεσμα κάποιος με τον κατάλληλο εξοπλισμό να μπορεί να υποδυθεί το BSS και να υποκλέπτει όλα τα μηνύματα που περνάνε από εκεί. Ταυτόχρονα δεν παρέχει κρυπτογράφηση από τέλος σε τέλος (end to end encryption), ενώ τα μηνύματα συνήθως κρυπτογραφούνται για τη μετάδοσή τους στο air interface, μετά αποκρυπτογραφούνται όταν φτάσουν στο BSS και κρυπτογραφούνται ξανά όταν φύγουν από αυτό, αφήνοντας έτσι ένα παράθυρο στο οποίο τα μηνύματα φαίνονται κανονικά αποκρυπτογραφημένα.

2.2. Τύποι Επιθέσεων

Υπάρχουν αρκετοί τύποι επιθέσεων που μπορούν να χρησιμοποιήσουν οι κρυπταναλυτές ώστε να αποκτήσουν δεδομένα. Ορισμένες από αυτές είναι οι ‘cold boot’ επιθέσεις, οι επιθέσεις Ωμής Βίας (Brute Force attack), οι Επιθέσεις Άρνησης Υπηρεσιών (Denial of Service attack, DoS) και οι ‘man in the middle’ επιθέσεις.

2.2.1. Cold Boot Επίθεση [4]

Όταν μία συσκευή απενεργοποιείται και αρχίζει να ψύχεται, ο περισσότερος κόσμος πιστεύει πως τα δεδομένα που βρίσκονται στη μνήμη χάνονται με το σβήσιμο της συσκευής. Ωστόσο στην πραγματικότητα η RAM (random allocated memory) της συσκευής χρειάζεται κάποιο χρόνο μέχρι να κρυώσει και τα δεδομένα που περιέχει διαγράφονται σταδιακά. Μία cold boot επίθεση εκμεταλλεύεται αυτό το γεγονός χρησιμοποιώντας τον μαγνητισμό που παραμένει στη μνήμη πριν κρυώσει ώστε να αποκτήσει πρόσβαση στα δεδομένα που παραμένουν σε αυτήν. Συνήθως αυτό λειτουργεί με τον ακόλουθο τρόπο: Ο κρυπταναλυτής αποκτά πρόσβαση στη συσκευή και την κάνει reboot χρησιμοποιώντας ένα δικό του λειτουργικό σύστημα

αποκτώντας έτσι πρόσβαση και στα δεδομένα που έχουν παραμείνει στη μνήμη. Μία ενδεικτική μέθοδος είναι απλά να πατήσει το κουμπί της επανεκκίνησης ή να κλείσει και να ξανανοίξει γρήγορα τη συσκευή. Αυτό έχει ως αποτέλεσμα να χαθούν ελάχιστα ή και καθόλου δεδομένα από τη μνήμη (εξαρτάται βέβαια και από τον τύπο της μνήμης), γιατί το σύστημα σταματάει τη διαγραφή των δεδομένων όταν η συσκευή επιστρέψει σε λειτουργία

2.2.2. Brute Force Attack (Επίθεση Ωμής Βίας)

Μία επίθεση τέτοιου τύπου είναι συνήθως η έσχατη λύση, και χρησιμοποιείται σε περίπτωση που ο κρυπταναλυτής δεν μπορεί να βρει κάποια πίσω πόρτα (backdoor) για να αποκτήσει πρόσβαση σε κάποια κρυπτογραφημένα δεδομένα. Αυτή η μέθοδος δοκιμάζει κλειδιά κρυπτογράφησης μέχρι να βρει το σωστό για να αποκρυπτογραφήσει αυτά τα δεδομένα, γι' αυτό και αυτή η μέθοδος μερικές φορές αναφέρεται και ως *'εξαντλητική έρευνα κλειδιού'* (exhaustive key search). Αυτή η μέθοδος βέβαια συνήθως υλοποιείται από λογισμικό το οποίο μπορεί να υπολογίσει όλους τους πιθανούς συνδυασμούς από τους οποίους μπορεί να αποτελείται ένας κωδικός. Το βασικό αντίμετρο εναντίον επιθέσεων brute force είναι η δημιουργία μεγάλων και ισχυρών κλειδιών, καθώς κάθε bit του κλειδιού αυξάνει εκθετικά τον χρόνο που θα χρειαστεί ένας κρυπταναλυτής για να σπάσει την κρυπτογράφηση. Επομένως ένα cipher text με ένα κλειδί από *n* bits χρειάζεται 2^n στη χειρότερη περίπτωση, το οποίο σημαίνει ότι ένα ισχυρό κλειδί μπορεί να κάνει μια τέτοια επίθεση να χρειαστεί δεκαετίες για να σπάσει τη κρυπτογράφηση, αλλά επίσης ότι ένα αδύναμο ή μικρό κλειδί μπορεί να σπάσει αρκετά γρήγορα.

2.2.3. Man in the Middle Attack

Αυτού του είδους η επίθεση αναχαιτίζει τις επικοινωνίες μεταξύ δύο σημείων εισάγοντας τον επιτιθέμενο ανάμεσά τους και παριστάνοντας τον αποστολέα και τον παραλήπτη και από τις δύο πλευρές. Με αυτόν τον τρόπο ο επιτιθέμενος μπορεί να διαβάζει και να παραποιεί όλα τα μηνύματα και από τα δύο σημεία.

2.2.4. Denial of Service Attack (DoS), (Επίθεση Άρνησης Υπηρεσιών) [21]

Αυτή η επίθεση αποδιοργανώνει τη λειτουργία ενός host με το να καθιστά τις υπηρεσίες του δικτύου μη διαθέσιμες προς τους προβλεπόμενους χρήστες. Αυτό γίνεται είτε προκαλώντας την κατάρρευση του συστήματος είτε πλημμυρίζοντάς το από αιτήσεις ώστε να μην μπορούν να συνδεθούν οι προβλεπόμενοι χρήστες. Οι χρήστες μπορεί να βιώσουν αδυναμία να συνδεθούν στην υπηρεσία, αποσύνδεση από την υπηρεσία, τρομερά χαμηλές ταχύτητες, αύξηση στην ανεπιθύμητη ηλεκτρονική αλληλογραφία (spam e-mail) ή αδυναμία να συνδεθούν σε κάποια συγκεκριμένη ή και όλες τις ιστοσελίδες.

Μία πιο επιτυχής επίθεση άρνησης υπηρεσιών είναι η Κατανεμημένη Επίθεση Άρνησης Υπηρεσιών (Distributed Denial of Service Attack, DDoS) όπου συμβαίνουν πολλαπλές επιθέσεις σε πολλά εκτεθειμένα συστήματα, πολλές φορές χρησιμοποιώντας άλλους εκτεθειμένους χρήστες. Συχνά αυτή η επίθεση δεν γίνεται με σκοπό την κλοπή δεδομένων αλλά για να αποδιοργανώσει κάποιο host, για λόγους ακτιβισμού η εκβιασμού.

2.2.5. Κίνδυνοι Έκθεσης

Είναι αρκετά σύνηθες κάποιος τρίτος να δει τα μηνύματα ενός τηλεφώνου, είτε γιατί ο ιδιοκτήτης του το ξέχασε σε κάποιο ταξί, στη δουλειά του, σε κάποια καφετέρια ή γενικότερα επειδή απλά το άφησε σε κοινή θέα χωρίς να το προσέχει για κάποιο διάστημα όπου κάποιος τρίτος μπορεί να έχει πρόσβαση , είτε γιατί κάποιος το έκλεψε. Για αυτόν τον λόγο τα περισσότερα σύγχρονα τηλέφωνα έχουν προστασία με κωδικό για να ανοίξει κανείς τα εισερχόμενα ή τα απεσταλμένα μηνύματα ή είναι αποθηκευμένα στη SIM όπου επίσης προστατεύονται από το PIN. Παρόλα αυτά, ο περισσότερος κόσμος συνήθως δεν χρησιμοποιεί αυτού του είδους την προστασία προς χάρη της διευκόλυνσής του και έτσι είναι εύκολο για κάποιον επιτιθέμενο όχι μόνο να δει τα μηνύματα αλλά και να υποδυθεί τον ιδιοκτήτη και να στείλει ή να παραλάβει μηνύματα στο όνομα του ιδιοκτήτη.

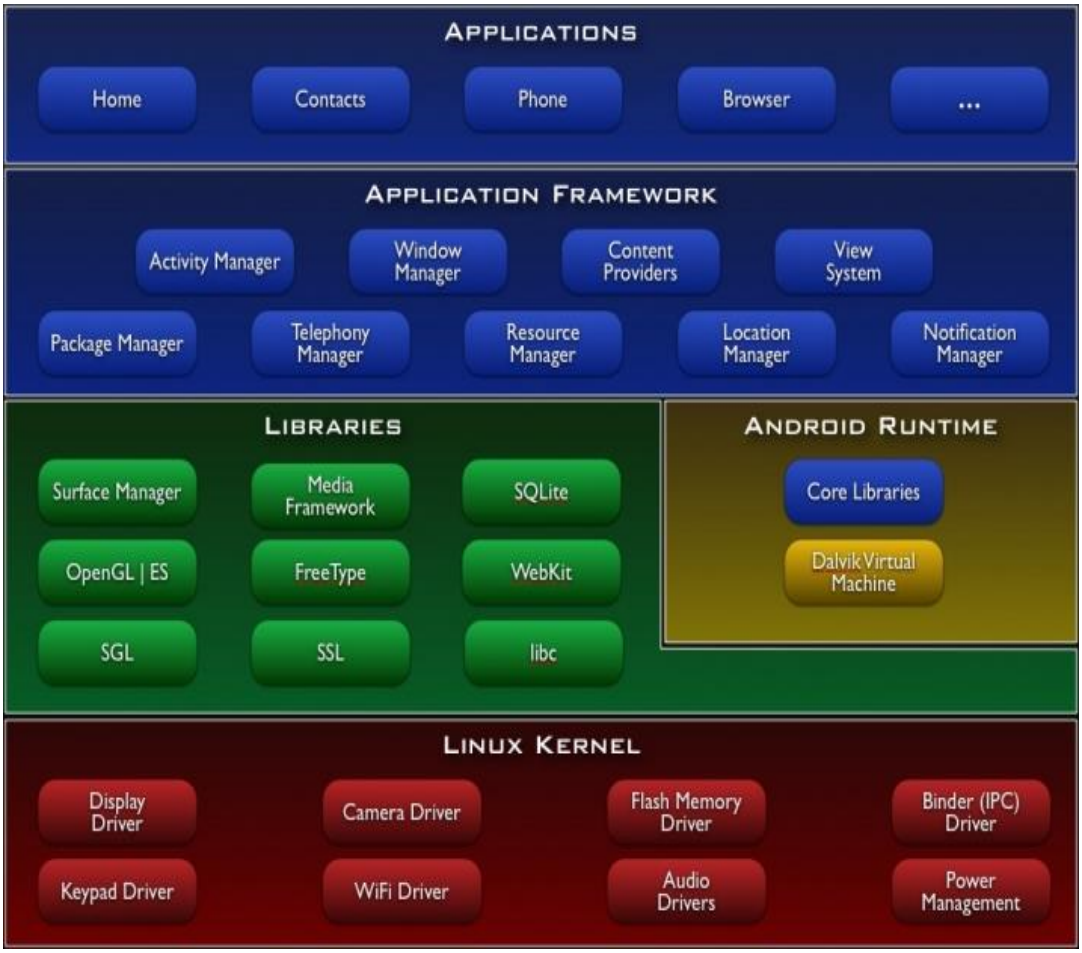
Η λογική πίσω από το πρόγραμμά μας είναι να περιορίσει τα τρωτά σημεία που αναδύονται κατά τη διάρκεια της αποστολής μηνυμάτων SMS μέσω του δικτύου

GSM, αλλά και σε περίπτωση που κάποιος επιτιθέμενος πάρει στα χέρια του τον Κινητό Σταθμό ενός χρήστη, να τον εμποδίσει από το να δει τα μηνύματά του και σε κάποιες περιπτώσεις να τον εμποδίσει από το να υποδυθεί τον ιδιοκτήτη του. Εκτός όμως από τα χαρακτηριστικά ασφαλείας, το πρόγραμμα σχεδιάστηκε ώστε να είναι εύκολο και προσιτό στη χρήση του αφού διατηρεί την εδραιωμένη εμφάνιση ενός messaging application, με τις συνομιλίες κάθε χρήστη και την διάταξη των μηνυμάτων με κάθε χρήστη ξεχωριστά σε μορφή chat.

2.3. Android

2.3.1. [6]Android Fundamentals (Βασικά Χαρακτηριστικά του Android)

Οι εφαρμογές Android είναι γραμμένες στη γλώσσα προγραμματισμού Java. Τα εργαλεία του Android SDK (Software Developer's Kit) μεταγλωττίζουν τον κώδικα μαζί με οποιαδήποτε δεδομένα ή άλλα αρχεία που πιθανόν να περιέχονται σε μια εφαρμογή, σε ένα πακέτο Android (Android package) το οποίο είναι ένα αρχείο με την κατάληξη .apk. Τα .apk αρχεία συγκροτούν τη κάθε μία εφαρμογή που μπορεί να εγκατασταθεί σε μία συσκευή με λειτουργικό Android.



2.3 Αρχιτεκτονική του Android

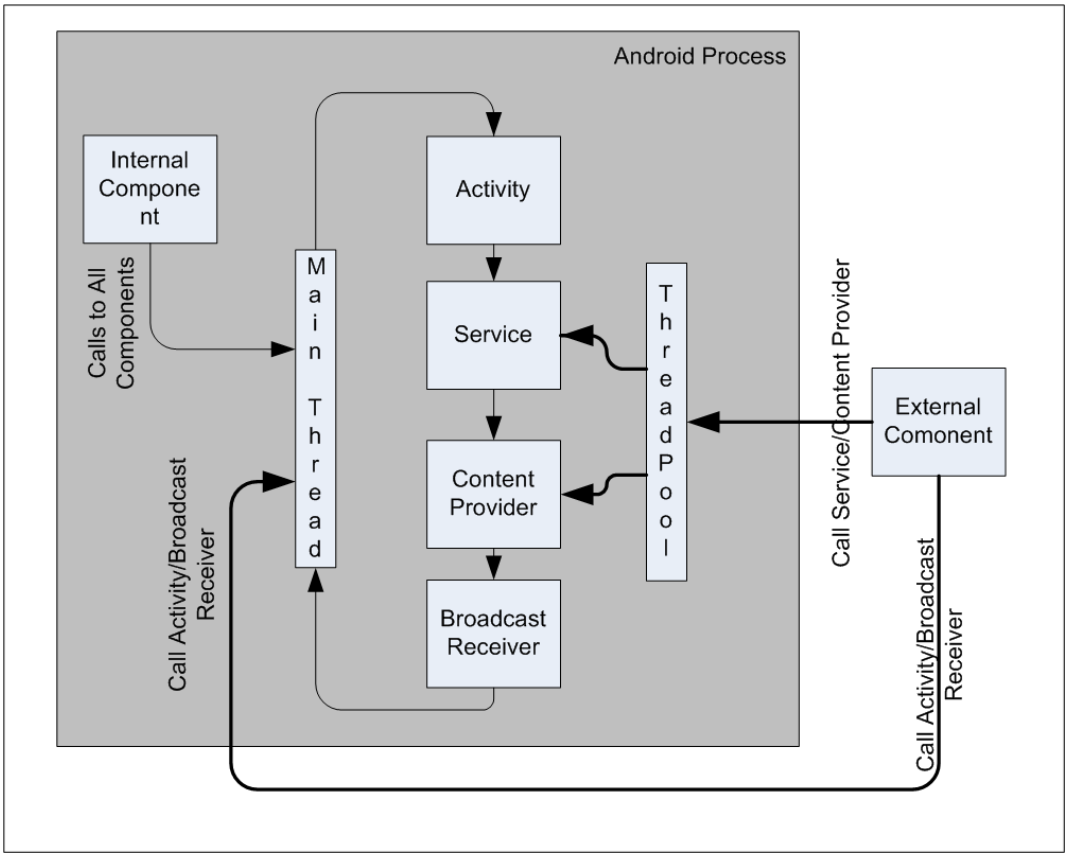
Τα βασικά χαρακτηριστικά του Android είναι:

- Το λειτουργικό σύστημα Android, είναι ένα σύστημα Linux πολλαπλών χρηστών, όπου κάθε εφαρμογή είναι ένας ξεχωριστός χρήστης.
- Το σύστημα αναθέτει σε κάθε εφαρμογή ένα ξεχωριστό user ID, το οποίο χρησιμοποιείται μόνο από το σύστημα και δεν το γνωρίζει η ίδια η εφαρμογή. Το σύστημα θέτει άδεια πρόσβασης για όλα τα αρχεία της εφαρμογής, έτσι ώστε μόνο το user ID της εφαρμογής να μπορεί να έχει πρόσβαση σε αυτά.
- Κάθε διαδικασία έχει τη δική της εικονική μηχανή (Virtual Machine, VM), δηλαδή ο κώδικας μίας εφαρμογής τρέχει απομονωμένος από άλλες εφαρμογές.

- Κάθε εφαρμογή τρέχει σε μια δική της διαδικασία σε Linux. Το Android ξεκινάει τη διαδικασία όταν κάποια εφαρμογή πρέπει να εκτελεστεί και την τερματίζει όταν δεν χρειάζεται πλέον ή όταν πρέπει να ανακτήσει μνήμη για κάποια άλλη εφαρμογή.
- Μία εφαρμογή μπορεί να ζητήσει άδεια για να αποκτήσει πρόσβαση σε δεδομένα της συσκευής όπως τις επαφές, τα μηνύματα SMS, τη κάρτα μνήμης και την κάμερα μεταξύ άλλων. Ο χρήστης πρέπει να δώσει ο ίδιος αυτήν την άδεια.

2.3.1.1. App Components (Μέρη Εφαρμογής)

Τα components μίας εφαρμογής είναι τα δομικά στοιχεία μίας εφαρμογής Android. Κάθε component είναι ένα διαφορετικό σημείο μέσα από το οποίο το σύστημα μπορεί να εισέλθει στην εφαρμογή. Δεν είναι όλα τα components σημεία εισαγωγής για τον χρήστη και μερικά μπορεί να εξαρτώνται από κάποιο άλλο, αλλά κάθε ένα υπάρχει σαν μία ξεχωριστή οντότητα και έχει ένα συγκεκριμένο ρόλο που ορίζει τη συμπεριφορά της εφαρμογής.



2.4 Android Process

Τα App Components είναι:

- **Activities (Δραστηριότητες)**

  Ένα activity εκπροσωπεί μία οθόνη με κάποιο user interface (διεπιφάνεια χρήστη). Για παράδειγμα, μία εφαρμογή για e-mail μπορεί να έχει ένα activity για να δείχνει τη λίστα με τα νέα e-mails, ένα άλλο activity για τη σύνταξη ενός e-mail και ένα τρίτο για να δείχνει το περιεχόμενο κάθε e-mail. Τα activities δουλεύουν μαζί ώστε να δημιουργήσουν μία συνεκτική εμπειρία για τον χρήστη, είναι όμως ανεξάρτητα μεταξύ τους. Έτσι, μια εφαρμογή μπορεί να ξεκινήσει ένα activity μίας άλλης εφαρμογής, για παράδειγμα μία εφαρμογή κάμερας να ξεκινήσει το activity σύνταξης e-mail ώστε να στείλει μία φωτογραφία.

- **Services (Υπηρεσίες)**

Ένα Service είναι ένα component που τρέχει στο πίσω μέρος (background) ώστε να παρέχει λειτουργίες μακράς διάρκειας ή να εκτελεί κάποια διαδικασία για την οποία δεν χρειάζεται ο χρήστης. Για παράδειγμα, ένα service μπορεί να παίζει μουσική στο πίσω μέρος, όσο ο χρήστης χρησιμοποιεί μία άλλη εφαρμογή ή να μεταφέρει δεδομένα μέσω κάποιου δικτύου χωρίς να επηρεάζει τη χρήση της συσκευής για άλλες διεργασίες. Άλλα components όπως τα activities μπορούν να ξεκινήσουν ένα service.

- **Content Providers (Προμηθευτές Περιεχομένου)**

  Ένα content provider διαχειρίζεται δεδομένα τα οποία μπορεί να είναι κοινόχρηστα, όπως σε μία βάση δεδομένων, στο διαδίκτυο ή σε οποιοδήποτε είδους αποθηκευτικό χώρο όπου η εφαρμογή μπορεί να έχει πρόσβαση. Για παράδειγμα, το Android παρέχει ένα content provider το οποίο διαχειρίζεται τις πληροφορίες των επαφών του χρήστη. Έτσι οποιαδήποτε εφαρμογή με τα κατάλληλα δικαιώματα (permissions) μπορεί να αποκτήσει πρόσβαση και να χρησιμοποιήσει ερωτήσεις (queries) για να πάρει ή να γράψει πληροφορίες για κάποια από τις επαφές. Εκτός από αυτό ένα content provider μπορεί να χρησιμοποιηθεί και για να διαβάζει ή να γράφει δεδομένα που είναι προσωπικά σε μία εφαρμογή.

- **Broadcast Receivers (Δέκτες Αναμετάδοσης)**

  Ένα broadcast receiver είναι ένα component που ανταποκρίνεται σε ανακοινώσεις που αναμεταδίδονται σε όλο το σύστημα. Πολλές αναμεταδόσεις ξεκινούν από το σύστημα, όπως για παράδειγμα ότι τελειώνει η μπαταρία ή ότι η εφαρμογή της κάμερας πήρε επιτυχώς μια φωτογραφία. Οι εφαρμογές μπορούν επίσης να ξεκινήσουν μια αναμετάδοση, για παράδειγμα, ώστε να ειδοποιήσουν άλλες εφαρμογές πως κάποια δεδομένα που κατέβασε το σύστημα είναι διαθέσιμα για άλλους χρήστες. Τα broadcast receivers δεν χρησιμοποιούν κάποιο user interface, αλλά μπορεί να δημιουργήσουν κάποια κοινοποίηση για να ειδοποιήσουν το χρήστη για κάποια αναμετάδοση που συνέβη.

Μία μοναδική ιδιότητα του Android είναι ότι οποιαδήποτε εφαρμογή μπορεί να εκκινήσει ένα component από κάποια άλλη εφαρμογή. Για παράδειγμα, για μία εφαρμογή κάμερας δεν χρειάζεται να δημιουργήσει κάποιος ένα activity το οποίο θα χρησιμοποιεί δικό του κώδικα για να παίρνει φωτογραφίες, ούτε να πάρει αυτούσιο κώδικα από κάποια άλλη εφαρμογή γι' αυτόν τον σκοπό. Μπορεί αντί γι' αυτό να ξεκινήσει ένα activity στην εφαρμογή της κάμερας το οποίο θα εκτελεί αυτήν την εργασία και μετά θα στείλει τη φωτογραφία στην αρχική εφαρμογή.

Όταν το σύστημα ξεκινάει ένα component, ξεκινάει τη διαδικασία της εφαρμογής και θέτει σε εφαρμογή τις κλάσεις που χρειάζονται για το component. Για παράδειγμα, αν η εφαρμογή ξεκινάει το activity στην εφαρμογή της κάμερας το οποίο παίρνει τις φωτογραφίες, αυτό το activity τρέχει στη διαδικασία που ανήκει στην εφαρμογή της κάμερας, όχι στην αρχική.

Επειδή το σύστημα τρέχει κάθε εφαρμογή σε διαφορετικές διαδικασίες, οι οποίες μπορεί να απαγορεύουν την πρόσβαση από άλλες εφαρμογές, μία εφαρμογή δεν μπορεί να ενεργοποιήσει απευθείας ένα component μίας άλλης εφαρμογής. Αυτό το κάνει το σύστημα του Android. Επομένως, για να ξεκινήσει ένα component σε μία άλλη εφαρμογή, πρέπει να σταλεί ένα μήνυμα προς το σύστημα το οποίο θα ξεκινήσει αυτό το component.

Ενεργοποίηση των Components

Το intent είναι ένα ασύγχρονο μήνυμα το οποίο μπορεί να ξεκινήσει τα τρία από τα τέσσερα components, τα activities, τα services και τα broadcast receivers. Είναι ουσιαστικά τα μηνύματα προς το σύστημα τα οποία δηλώνουν την πρόθεση του χρήστη να ξεκινήσει ένα component, είτε αυτό ανήκει στην ίδια είτε σε άλλη εφαρμογή. Τα intents δημιουργούνται με ένα Intent αντικείμενο (object).

Για τα activities και τα services, ένα intent ορίζει μία ενέργεια που πρέπει να εκτελεστεί, όπως το να στείλει μια εφαρμογή κάτι, ενώ μπορεί να ορίζει ταυτόχρονα κάποιο URI (Uniform Resource Identifier) δεδομένων στο οποίο θα ενεργήσει. Για παράδειγμα, ένα intent μπορεί να μεταφέρει ένα αίτημα ώστε ένα activity να ανοίξει μια ιστοσελίδα. Επίσης, μπορεί να ζητάει κάποιο αποτέλεσμα ως επιστροφή,

όπως το να επιστρέψει κάποιο στοιχείο από τις επαφές του χρήστη. Σε αυτήν την περίπτωση επιστρέφει τα στοιχεία επίσης με ένα intent το οποίο περιλαμβάνει και το URI της επιλεγμένης επαφής.

Αντίστοιχα για τα broadcast receivers, ένα intent ορίζει την ανακοίνωση που θα εκπεμφθεί. Μία τέτοια περίπτωση είναι η ανακοίνωση ότι η μπαταρία τελειώνει.

Τα content providers δεν ενεργοποιούνται από intents, αντίστοιχα ενεργοποιούνται από το ContentResolver. Το content resolver διαχειρίζεται όλες τις απευθείας συναλλαγές με το content provider, ώστε να μην εμπλέκεται το component που εκτελεί την συναλλαγή και να τρέχει απευθείας τις μεθόδους στο αντικείμενο content resolver.

2.3.1.2. Το Αρχείο Manifest

Για να μπορεί το σύστημα του Android να ξεκινήσει ένα component, το σύστημα πρέπει να γνωρίζει ότι αυτό το component υπάρχει. Αυτό γίνεται διαβάζοντας το αρχείο manifest (AndroidManifest.xml). Η εφαρμογή πρέπει να δηλώνει όλα τα components της σε αυτό το αρχείο, το οποίο πρέπει να βρίσκεται στο root του αρχείου της εφαρμογής.

Το αρχείο manifest εκτελεί διάφορες εργασίες:

- Αναγνωρίζει ό,τι δικαιώματα χρειάζεται η εφαρμογή, όπως πρόσβαση στο διαδίκτυο ή πρόσβαση στις επαφές του χρήστη.
- Δηλώνει το ελάχιστο επίπεδο API (Application Programming Interface) που χρειάζεται η εφαρμογή, το οποίο βασίζεται στο API που χρειάζεται για να τρέξει ο κώδικας της εφαρμογής.
- Δηλώνει το hardware και το λογισμικό (software) που χρησιμοποιεί ή χρειάζεται η εφαρμογή, όπως η κάμερα ή blue tooth.
- Δηλώνει βιβλιοθήκες API με τις οποίες η εφαρμογή πρέπει να είναι συνδεδεμένη πέρα από το Android API, όπως το Google Maps library.
- Δηλώνει τα components που χρειάζεται η εφαρμογή.
- Δηλώνει τις δυνατότητες που μπορεί να χρησιμοποιούν τα components.

- Δηλώνει τις απαιτήσεις που μπορεί να έχει μία εφαρμογή, γιατί το Android υπάρχει σε πολλές συσκευές οι οποίες δεν διαθέτουν όλες τις ίδιες δυνατότητες.

2.3.1.3. App Resources (Πόροι)

Μία εφαρμογή Android περιέχει εκτός από τον κώδικα, τα resources, δηλαδή φωτογραφίες, αρχεία ήχου και οτιδήποτε έχει σχέση με την εμφάνιση της εφαρμογής. Τα διάφορα animations, menus, styles, χρώματα και διάταξη (layout) ορίζονται με αρχεία XML. Με τη χρήση των resources γίνεται ευκολότερη η ενημέρωση των διαφόρων χαρακτηριστικών της εφαρμογής, χωρίς να χρειαστεί αλλαγή στον κώδικα, καθώς επίσης και η βελτιστοποίησή της, ώστε να είναι πιο συμβατή με περισσότερες συσκευές, όπως για συσκευές με διαφορετικά μεγέθη οθονών ή διαφορετικές γλώσσες.

Ένα από τα σημαντικότερα πλεονεκτήματα της ύπαρξης ξεχωριστών από τον κώδικα resources, είναι η δυνατότητα να παρέχει διαφορετικά resources για διαφορετικούς τύπους συσκευών. Αυτό για παράδειγμα, είναι ιδιαίτερα σημαντικό για μετάφραση της εφαρμογής σε πολλές γλώσσες. Γράφοντας στοιχεία του UI (user interface) σε strings σε XML και σώνοντας τη μετάφρασή τους σε ξεχωριστά αρχεία μέσα στο αρχείο resources με το κατάλληλο qualifier, όπως res/values-gr/ για ελληνικά, το σύστημα μπορεί αυτόματα, σύμφωνα με τις ρυθμίσεις γλώσσας του χρήστη, να ενεργοποιήσει το αντίστοιχο resource αρχείο με την κατάλληλη μετάφραση.

2.3.2. API (Application Programming Interface) [7][8]

Τα API είναι προκαθορισμένοι σκελετοί (frameworks) λογισμικού, τα οποία παρέχουν επαναχρησιμοποιήσιμες βιβλιοθήκες και προγραμματιστικά μοντέλα, με σκοπό να δημιουργούνται χρήσιμα προγράμματα, εύκολα, γρήγορα και με κώδικα υψηλής ποιότητας.

Το Android παρέχει πολλά APIs μέσα από τα οποία οι εφαρμογές μπορούν να επικοινωνούν με το σύστημα. Το framework API του Android αποτελείται από:

- Μία βασική συλλογή από πακέτα (packages) και κλάσεις (classes)

- Μία συλλογή από στοιχεία και ιδιότητες της XML προς δήλωση του αρχείου manifest
- Μια συλλογή από στοιχεία και ιδιότητες της XML προς δήλωση και πρόσβαση στα resources
- Μία συλλογή από Intents
- Μία συλλογή από δικαιώματα τα οποία μπορεί να ζητήσει η εφαρμογή και δικαιώματα που επιβάλλονται από το σύστημα.

Φυσικά με τον καιρό ενημερώνονται ή προστίθενται και άλλα APIs. Οι ενημερώσεις του framework API σχεδιάζονται με τέτοιον τρόπο, ώστε τα καινούρια APIs να είναι συμβατά με τις παλιότερες εκδόσεις του λογισμικού, ενώ τα παλιότερα μέρη του API που ενημερώθηκαν και άλλαξαν εξακολουθούν να υπάρχουν και να λειτουργούν ώστε οι ήδη υπάρχουσες εφαρμογές να μπορούν να τα χρησιμοποιήσουν, αλλά η χρήση τους πλέον αποδοκιμάζεται. Είναι πολύ λίγες εξαιρέσεις που τα παλιότερα APIs αντικαθίστανται, αλλά αυτό συμβαίνει μόνο για την εξασφάλιση της ευρωστίας και ασφάλειας των εφαρμογών ή του συστήματος .

Οι εκδόσεις του λογισμικού ενημερώνονται αντίστοιχα με κάθε αλλαγή στα APIs και τους ανατίθεται ένας μοναδικός αριθμός, το API level. Αυτήν τη στιγμή υπάρχουν 23 API levels στο Android, κάθε ένα από τα οποία αντιστοιχεί σε μία έκδοση Android που με τη σειρά τους παίρνουν το όνομά τους από κάποιο επιδόρπιο.

Τα API Levels του Android και οι αντίστοιχες εκδόσεις του λογισμικού:

Platform Version	API Level	VERSION_CODE
Android 6.0	23	MARSHMALLOW
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP
Android 4.4W	20	KITKAT_WATCH
Android 4.4	19	KITKAT
Android 4.3	18	JELLY_BEAN_MR2
Android 4.2, 4.2.2	17	JELLY_BEAN_MR1
Android 4.1, 4.1.1	16	JELLY_BEAN
Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH

Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2 Android 2.3.3 Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

2.5 Εκδόσεις Android

Ο ρόλος των API levels είναι να εξασφαλίζει την καλύτερη δυνατή εμπειρία για τους χρήστες και τους προγραμματιστές, καθώς:

- Επιτρέπει στην πλατφόρμα του Android να γνωρίζει τη μεγαλύτερη έκδοση του framework API το οποίο υποστηρίζει.
- Επιτρέπει στις εφαρμογές να επεξηγούν ποια έκδοση του framework API χρειάζονται για να λειτουργήσουν.
- Επιτρέπει στο σύστημα να μην αποδέχεται την εγκατάσταση εφαρμογών τα οποία δεν είναι συμβατά με την έκδοση του API της συσκευής.

Οι εφαρμογές μπορούν, με ένα στοιχείο του manifest που προμηθεύει το framework API, να δηλώνουν το ελάχιστο και μέγιστο API στα οποία είναι δυνατό να τρέξουν οι εφαρμογές, καθώς και το προτεινόμενο επίπεδο API για το οποίο σχεδιάστηκαν. Αυτή η ιδιότητα είναι σημαντική γιατί επιτρέπει στο σύστημα να γνωρίζει αν η εφαρμογή που προσπαθεί να εγκαταστήσει είναι συμβατή με τη συσκευή ή όχι, έτσι ώστε αν χρειαστεί να μην επιτρέψει την εγκατάσταση.

2.3.2.1. Application Forward Compatibility (Προς τα Εμπρός Συμβατότητα Εφαρμογών)

Οι εφαρμογές Android είναι ως επί το πλείστον, προς τα εμπρός συμβατές με τις καινούριες εκδόσεις της πλατφόρμας. Επειδή σχεδόν όλες οι αλλαγές στο framework API είναι πρόσθετες, οι εφαρμογές που τρέχουν σε παλαιότερο επίπεδο API μπορούν να τρέξουν και σε νεότερο επίπεδο. Υπάρχουν κάποιες εξαιρέσεις στις οποίες αφαιρείται κάποιο API, όπως αναφέρθηκε και νωρίτερα. Αν η εφαρμογή βασίζεται σε κάποιο API που αφαιρέθηκε, τότε δεν θα λειτουργεί στο καινούριο επίπεδο API. Εκτός από την εξαίρεση αυτή, είναι πιθανό μετά από ενημέρωση του API κάποια μέρη της εφαρμογής να συμπεριφέρονται με κάπως διαφορετικό τρόπο ή να αλλάζει η εμφάνισή τους. Για αυτές τις περιπτώσεις είναι σημαντικό οι προγραμματιστές να δοκιμάζουν και να εξετάζουν τον τρόπο λειτουργίας της εφαρμογής σε διάφορα επίπεδα API, για να εξασφαλίζουν την ομαλή λειτουργία της εφαρμογής. Για τη βοήθεια των προγραμματιστών σε αυτόν τον τομέα, υπάρχουν εικονικές συσκευές Android (Android Virtual Device, AVD), στις οποίες μπορεί να δοκιμάσει όλα τα επίπεδα API.

2.3.2.2. Application Backward Compatibility (Προς τα Πίσω Συμβατότητα Εφαρμογών)

Οι εφαρμογές Android δεν είναι αναγκαστικά προς τα πίσω συμβατές με παλαιότερες εκδόσεις της πλατφόρμας, από αυτήν για την οποία δημιουργήθηκε. Κάθε καινούρια έκδοση της πλατφόρμας Android μπορεί να περιέχει καινούρια framework APIs, τα οποία μπορεί να προσφέρουν δυνατότητες που οι παλαιότερες εκδόσεις δεν προσέφεραν ή να είναι μέρος κάποιου API που αντικαταστάθηκε. Επομένως, εφαρμογές που χρησιμοποιούν APIs κάποιου πρόσφατου επιπέδου είναι πιθανό να μην τρέχουν σε συσκευές με παλαιότερη έκδοση, αφού δεν θα διαθέτουν το κατάλληλο API.

2.4. Ανάγκες που καλύπτει η εφαρμογή

Οι ανάγκες που καλύπτει η εφαρμογή που σχεδιάσαμε αφορούν την ασφάλεια της επικοινωνίας και την ασφάλεια των προσωπικών δεδομένων σε πολλά επίπεδα:

Α. Ασφάλεια Επικοινωνίας

Σε προσωπικό επίπεδο καλύπτει φυσικά την ανάγκη της ασφάλειας της επικοινωνίας. Πάρα πολλοί χρήστες χρησιμοποιούν SMS για τις επικοινωνίες τους και βέβαια σε αυτές τις επικοινωνίες είναι πολύ λογικό να μεταφέρουν συχνά ευαίσθητες πληροφορίες τις οποίες δεν θα ήθελαν με κάποιο τρόπο να πέσουν στα χέρια τρίτων. Σε προσωπικό επίπεδο είναι σπάνιο να υποκλέψει κάποιος ξένος τα μηνύματα κατά την διάρκεια της μετάδοσης, μιας και συνήθως χρειάζεται ειδική γνώση και εξοπλισμός για να επιτευχθεί κάτι τέτοιο. Παρόλα αυτά η εφαρμογή μας παρέχει end to end κρυπτογράφηση και καλύπτει τα τρωτά σημεία του δικτύου GSM και γενικότερα τις Man in the middle επιθέσεις και έτσι προστατεύει τις πληροφορίες κατά τη μετάδοσή τους.

Β. Προστασία Προσωπικών Δεδομένων

Σε προσωπικό επίπεδο πάλι η εφαρμογή μας, καλύπτει την ανάγκη προστασίας των προσωπικών δεδομένων. Η εφαρμογή παρέχει end to end κρυπτογράφηση, αλλά η αποκρυπτογράφηση χρειάζεται κωδικό. Αυτό σημαίνει ότι τα μηνύματα παραμένουν κρυπτογραφημένα μέσα στην συσκευή και έτσι δεν μπορεί κάποιος τρίτος να αποκτήσει πρόσβαση στην συσκευή και να δει τα μηνύματα χωρίς τον κωδικό κρυπτογράφησης.

Ειδικά σε επαγγελματικό επίπεδο όπου συχνά χρησιμοποιούνται SMS για την επικοινωνία μεταξύ στελεχών, αλλά και για την μεταφορά σημαντικών δεδομένων, από ημερομηνίες συναντήσεων, μέχρι κωδικούς για πρόσβαση στον server της εταιρίας, συναλλαγές, στατιστικά στοιχεία και άλλες ευαίσθητες πληροφορίες, είναι κριτικής σημασίας να παραμείνουν μυστικά. Έτσι λοιπόν εξασφαλίζεται η ανάγκη της μυστικότητας της επικοινωνίας, αφού όπως προαναφέραμε η εφαρμογή προστατεύει από τις man in the middle επιθέσεις. Επομένως οι πληροφορίες αυτές είναι πολύ δύσκολο να υποκλαπούν κατά τη διανομή τους.

Γ. Διατήρηση Μυστικότητας Δεδομένων

Αντίστοιχα με τη περίπτωση των προσωπικών δεδομένων καλύπτεται και η ανάγκη μυστικότητας των δεδομένων. Σε επαγγελματικό επίπεδο είναι πολύ πιθανότερο από ό,τι σε προσωπικό επίπεδο, να υπάρξει κάποιος έμπειρος κρυπταναλυτής που έχει σκοπό να αποκτήσει πρόσβαση στα μηνύματα κάποιου ανταγωνιστή. Η εφαρμογή μας χρησιμοποιεί 128-bit AES (Advanced Encryption Standard) σαν σύστημα κρυπτογράφησης. Αυτό σε συνδυασμό με τα υπόλοιπα μέτρα που έχουμε λάβει, όπως την τυχαιότητα της παραγωγής του κλειδιού κρυπτογράφησης μέσω κωδικού, προστατεύει από brute force επιθέσεις, επομένως αν κάποιος αποκτήσει φυσική πρόσβαση στη συσκευή δεν θα μπορέσει να αποκτήσει πρόσβαση και στα μηνύματα.

Δ. Εξασφάλιση Ακεραιότητας Δεδομένων

Εκτός από αυτά η εφαρμογή μας καλύπτει και την ανάγκη ακεραιότητας των δεδομένων. Εξασφαλίζει δηλαδή ότι το μήνυμα που εστάλη από τη μια πλευρά παραλείφθηκε από την άλλη πλευρά χωρίς να έχει τροποποιηθεί ή πειραχτεί με κάποιον τρόπο κατά τη διάρκεια της μετάδοσης. Για να το πετύχουμε αυτό προσθέσαμε έναν αλγόριθμο κατακερματισμού (hash algorithm) που λειτουργεί κατά την κρυπτογράφηση. Αυτός ο αλγόριθμος δημιουργεί μια σειρά χαρακτήρων σαν δαχτυλικό αποτύπωμα που ελέγχεται από έναν ίδιο αλγόριθμο κατά την αποκρυπτογράφηση και καταλαβαίνει αν κάτι έχει πειραχτεί στο κείμενο.

Ε. Πιστοποίηση της Επικοινωνίας

Τέλος η εφαρμογή μας καλύπτει και την ανάγκη πιστοποίησης της επικοινωνίας, εξασφαλίζει δηλαδή ότι οι δύο πλευρές της επικοινωνίας είναι αυτοί που ισχυρίζονται ότι είναι και όχι κάποιος τρίτος που παριστάνει την μια από τις δύο πλευρές. Αυτό επιτυγχάνεται απλά γιατί ο αλγόριθμος κρυπτογράφησης είναι συμμετρικού κλειδιού και χρειάζεται ο ίδιος κωδικός για την κρυπτογράφηση και την αποκρυπτογράφηση του μηνύματος. Έτσι όσο ο κωδικός παραμένει μυστικός υπάρχει πιστοποίηση των δύο άκρων.

3. Σχεδιασμός

Πριν ξεκινήσουμε την υλοποίηση της εφαρμογής ήταν σημαντικό να καθορίσουμε τι ακριβώς θα κάνει η εφαρμογή, ποιες ανάγκες θα καλύπτει και με ποιον τρόπο. Επίσης, κατά τη διάρκεια της υλοποίησης εμφανίστηκαν διάφορα προβλήματα είτε λειτουργικά είτε σχεδιαστικά, που έπρεπε να αντιμετωπιστούν χωρίς να γίνουν συμβιβασμοί στον τρόπο λειτουργίας και τις δυνατότητες της εφαρμογής.

3.1. Επιλογή Αλγόριθμου Κρυπτογράφησης

Μία από τις σημαντικότερες προκλήσεις στην υλοποίηση της εφαρμογής ήταν η επιλογή του είδους της κρυπτογράφησης. Υπάρχουν πολλά είδη κρυπτογράφησης, όπως ιδιωτικού κλειδιού, δημοσίου κλειδιού, συμμετρική, ασύμμετρη, ελλειπτική, με κωδικό ή χωρίς και πολλά ακόμα (περισσότερες πληροφορίες στο κεφάλαιο 6). Για την επιλογή του κατάλληλου είδους έπρεπε να σκεφτούμε από ποιες επιθέσεις πρέπει να προστατευτούμε. Τα δύο συνηθέστερα προβλήματα για την περίπτωση του SMS είναι οι man-in-the-middle επιθέσεις και η αθέμιτη πρόσβαση στο τηλέφωνο από τρίτους, το οποίο μπορεί να αναχθεί σε brute force επίθεση. Επίσης, λόγω των ιδιαιτεροτήτων του SMS όσον αφορά το μέγεθος των μηνυμάτων (160 χαρακτήρες), έπρεπε να διαλέξουμε έναν αλγόριθμο κρυπτογράφησης ο οποίος να αφήνει περιθώριο για ικανοποιητικό αριθμό χαρακτήρων προς μεταφορά.

3.2. Συμβατότητα API

Μία ακόμα πρόκληση στην υλοποίηση ήταν να γίνει η εφαρμογή συμβατή με όσο το δυνατόν περισσότερες συσκευές. Αυτό αποτελούσε πρόβλημα γιατί το SMS δεν ήταν μέρος του επίσημου API του Android μέχρι και το API 18 (Android version 4.3, Jellybean), το οποίο σημαίνει ουσιαστικά ότι κάθε κατασκευαστής (εταιρία) που κυκλοφορούσε κάποιο κινητό τηλέφωνο, χρησιμοποιούσε το δικό της κώδικα για την εφαρμογή του SMS που μπορεί να ήταν διαφορετικός από κάποιου άλλου κατασκευαστή. Το πρόβλημα αυτό δεν αφορά στον τρόπο με τον οποίο γίνεται η αποστολή και η λήψη των μηνυμάτων, αλλά κυρίως στον τρόπο με τον οποίο η συσκευή αποθηκεύει τα μηνύματα στη βάση δεδομένων του συστήματος καθώς και

στον τρόπο που η εφαρμογή μπορεί να αποκτήσει πρόσβαση στη βάση αυτή και να απεικονίσει τα μηνύματα. Από το API 19 (Android version 4.4, Kit Kat) το SMS εντάχθηκε στο επίσημο API του Android και επομένως δεν υπάρχει κάποιο πρόβλημα συμβατότητας του κώδικα.

Πιο αναλυτικά, η σύγχυση που δημιουργείται πριν το API 19 έχει να κάνει με τη διαφορά στα URI (Uniform Resource Identifier) κάθε συσκευής. Συγκεκριμένα το URI, content://sms το οποίο είναι το resource στο σύστημα που περιέχει τα μηνύματα SMS. Το content://sms χωρίζεται συνολικά στα:

- Inbox, content://sms/inbox το οποίο περιέχει όλα τα μηνύματα που έχει λάβει ο χρήστης.
- Failed, content://sms/failed το οποίο περιέχει όλα τα μηνύματα που απέτυχαν στη μεταφορά τους, συνήθως λόγω κάποιου σοβαρού τεχνικού ζητήματος, όπως ότι δεν υπάρχει ο συνδρομητής αποστολής ή δεν υπάρχει πρόσβαση στο δίκτυο.
- Queued, content://sms/queued το οποίο περιέχει όλα τα μηνύματα που είναι στη διαδικασία αποστολής αλλά δεν έχουν αποσταλεί ακόμα.
- Sent, content://sms/sent το οποίο περιέχει όλα τα μηνύματα που έχουν αποσταλεί.
- Draft, content://sms/draft το οποίο περιέχει όλα τα προσχέδια μηνυμάτων (drafts).
- Outbox, content://sms/outbox το οποίο περιέχει όλα τα μηνύματα που έχουν αποσταλεί ή απέτυχαν ή είναι στη σειρά για αποστολή. Περιέχει τα sent, failed, queued, και undelivered.
- Undelivered, content://sms/undelivered το οποίο περιέχει όλα τα μηνύματα που απέτυχαν στη μεταφορά τους συνήθως λόγω κάποιου αμελητέου τεχνικού ζητήματος, όπως ότι το inbox του παραλήπτη είναι γεμάτο.
- All, content://sms/all το οποίο περιέχει όλα τα μηνύματα SMS της συσκευής.
- Conversations, content://sms/conversations το οποίο περιέχει όλα τα μηνύματα ανά επαφή του χρήστη, απεσταλμένα και παραληφθέντα, δηλαδή τις συνομιλίες.

Αυτά τα URI περιέχουν τα μηνύματα SMS σε μία συσκευή και κάθε SMS στη βάση περιέχει μεταξύ άλλων τα πεδία:

- Body, το οποίο είναι το κείμενο του μηνύματος.
- Address, το οποίο είναι ο αριθμός τηλεφώνου του παραλήπτη.
- Thread_id, το οποίο είναι ένας αριθμός που θέτει η συσκευή σε κάθε συνομιλία με τις επαφές του χρήστη. Για παράδειγμα, τα απεσταλμένα και παραληφθέντα μηνύματα, μεταξύ του χρήστη και του Γιώργου θα έχουν thread_id 3, ενώ τα απεσταλμένα και παραληφθέντα μηνύματα μεταξύ του χρήστη και της Μαρίας θα έχουν thread_id 5.
- Date, το οποίο είναι η ημερομηνία και ώρα αποστολής του μηνύματος σε milliseconds.
- Type, το οποίο είναι ένας αριθμός για τη διαφοροποίηση μεταξύ απεσταλμένων και παραληφθέντων, όπου 1 είναι τα μηνύματα που έχουν ληφθεί και 2 αυτά που έχουν αποσταλεί.

Το πρόβλημα, λοιπόν, είναι ότι επειδή τα παραπάνω δεν είναι μέρος του επίσημου API του Android, δεν χρησιμοποιούνται όλα αυτά τα URI από όλους τους κατασκευαστές, αλλά ούτε και με τον ίδιο τρόπο. Δηλαδή, για να εμφανίσουμε τις συνομιλίες, που είναι η πιο κλασσική υλοποίηση για εφαρμογές μηνυμάτων, θα καλούσε κανείς το content://sms/conversations, το οποίο όμως δεν υπάρχει σαν URI στα τηλέφωνα της SAMSUNG για παράδειγμα.

3.3. Τηλεφωνικοί Κωδικοί Χωρών

Οι αριθμοί των τηλεφώνων που καταχωρούνται στη βάση του συστήματος των εισερχόμενων μηνυμάτων περιλαμβάνουν πάντα τον κωδικό της χώρας καθώς εισάγεται αυτόματα από το σύστημα, στη περίπτωση της Ελλάδας είναι το +30. Οι χρήστες όμως όταν σώζουν κάποια επαφή ή όταν θέλουν να στείλουν απευθείας μήνυμα σε κάποιον αριθμό, δεν συνηθίζουν να πληκτρολογούν τον κωδικό οι ίδιοι. Αυτό δημιουργεί μια σύγχυση στην καταχώρηση των μηνυμάτων γιατί τα εισερχόμενα έχουν τον κωδικό της χώρας ενώ τα εξερχόμενα όχι, με αποτέλεσμα να καταχωρούνται σαν διαφορετικές διευθύνσεις με διαφορετικό thread_id και να μην

εμφανίζονται σαν μέρη της ίδιας συζήτησης, παρόλο που τα μηνύματα μεταδίδονται κανονικά.

3.4. Επιλογή Κατάλληλου Interface (Διαπροσωπία)

Μια πρόκληση ακόμα για την εφαρμογή ήταν η επιλογή του κατάλληλου interface. Αν θα ήταν αρκετά απλοϊκό, όπου σε αυτήν την περίπτωση τα μηνύματα θα επιλέγονταν από μια λίστα που θα περιείχε όλα τα μηνύματα, το καθένα προς εμφάνιση σε μια ξεχωριστή οθόνη ή θα χρησιμοποιούσε πιο μοντέρνα εμφάνιση με ξεχωριστά conversations για κάθε χρήστη και τα μηνύματα κάθε conversation σε μορφή chat. Επίσης αν μέσα στο chat τα μηνύματα θα εμφανίζονταν σε μια μονή λίστα, σε διπλή λίστα, με chat bubbles ή χωρίς, φωτογραφίες και άλλα. Εκτός από το στυλιστικό κομμάτι, η πιο σύνθετη απεικόνιση των conversations και chat, εισήγαγε και την πρόκληση της μεθόδου που θα χρησιμοποιούσαμε για την απεικόνιση αυτών των κομματιών. Μια τέτοια μέθοδος θα έπρεπε να είναι ελαφριά και αποδοτική ώστε το τελικό αποτέλεσμα που παράγει να μην χρησιμοποιεί πολλή μνήμη, να είναι γρήγορο, ομαλό και εύκολο στη χρήση.

3.5. Δυνατότητες Χρήστη

Άλλη μια πρόκληση στον σχεδιασμό της εφαρμογής ήταν τί επιπλέον δυνατότητες θα μπορούσε να έχει ο χρήστης όταν χρησιμοποιεί την εφαρμογή μας, εκτός από την δυνατότητα να κρυπτογραφεί ένα μήνυμα και να το στέλνει στον προορισμό του. Για παράδειγμα αν θα στέλνει μόνο κρυπτογραφημένα μηνύματα ή θα μπορεί να στέλνει και κανονικά μηνύματα, αν τα μηνύματα που αποκρυπτογραφεί ο χρήστης θα αποθηκεύονται στο σύστημα αυτόματα, με επιλογή ή καθόλου και αν θα μπορεί ο χρήστης να επηρεάσει τα ήδη υπάρχοντα μηνύματα της συσκευής.

4. Υλοποίηση

4.1. Αλγόριθμος Κρυπτογράφησης

Για την κρυπτογράφηση των μηνυμάτων στην εφαρμογή, καταλήξαμε στο Advanced Encryption Standard (AES) με 128-bit κλειδί, το οποίο δημιουργείται από έναν κωδικό που εισάγει ο χρήστης (password based encryption, PBE) (περισσότερες πληροφορίες στο κεφάλαιο 6). Αυτός ο αλγόριθμος καλύπτει τις ανάγκες που αναφέραμε νωρίτερα αφού αντιμετωπίζει τα περισσότερα είδη επιθέσεων, ενώ ξοδεύει 68 από του 160 χαρακτήρες από το μήνυμα, το οποίο είναι αρκετό περιθώριο για να μεταδώσει κανείς σημαντικές πληροφορίες με κρυπτογράφηση.

Όσον αφορά την κρυπτογράφηση η εφαρμογή αντιμετωπίζει:

- Τις man-in-the-middle επιθέσεις, γιατί κρυπτογραφεί τα μηνύματα πριν τα στείλει, επομένως τα μηνύματα μεταδίδονται κρυπτογραφημένα σε όλη τη διάρκεια της μεταφοράς τους (end to end). Έτσι δεν υπάρχει χρονική στιγμή που κάποιος με τον κατάλληλο εξοπλισμό να μπορεί να υποκλέψει τα μηνύματα και να τα διαβάσει. Επίσης, επειδή χρειάζεται τον κωδικό με το οποίο κρυπτογραφήθηκε το μήνυμα για να αποκρυπτογραφηθεί, εμποδίζει και το άλλο σκέλος της man-in-the-middle επίθεσης στο οποίο ο επιτιθέμενος μπορεί να υποδυθεί τον αποστολέα. Έτσι καλύπτονται και τα σημαντικότερα εκτεθειμένα σημεία του δικτύου GSM.
- Την αθέμιτη πρόσβαση του τηλεφώνου από τρίτους, γιατί τα μηνύματα παραμένουν κρυπτογραφημένα (εκτός αν επιλέξει ο χρήστης να τα κρατήσει αποκρυπτογραφημένα). Άρα ένας τρίτος που αποκτάει πρόσβαση στη συσκευή δεν μπορεί να δει τα μηνύματα χωρίς τον κατάλληλο κωδικό. Ο κωδικός κρυπτογράφησης, διαφέρει σε λειτουργία από τον κωδικό που μπορεί να θέσει κανείς στο τηλέφωνο ή στη συνηθισμένη εφαρμογή SMS, γιατί οι τελευταίοι εμποδίζουν την πρόσβαση στο τηλέφωνο, αλλά αν κάποιος κατεβάσει τα δεδομένα του τηλεφώνου σε κάποια άλλη συσκευή,

μπορεί να τα διαβάσει χωρίς πρόβλημα, ενώ στην εφαρμογή μας ακόμα και τότε θα εξακολουθεί να βλέπει το κείμενο κρυπτογραφημένο.

- Τις brute force επιθέσεις. Εφόσον τα μηνύματα είναι κρυπτογραφημένα με κωδικό, οποιοσδήποτε αποκτήσει πρόσβαση στα μηνύματα χωρίς να γνωρίζει τον κωδικό θα χρειαστεί κατά πάσα πιθανότητα να εξασκήσει μια brute force επίθεση για να σπάσει την κρυπτογράφηση. Η αντιμετώπιση αυτού του προβλήματος εξαρτάται από τη δύναμη του αλγορίθμου κρυπτογράφησης και την πολυπλοκότητα του κλειδιού. Όπως αναφέρθηκε νωρίτερα ο χρόνος δοκιμής κλειδιών για μια τέτοια επίθεση είναι 2^n στη χειρότερη περίπτωση, όπου n είναι τα bits του κλειδιού. Στην περίπτωση της εφαρμογής μας, το κρυπτογραφικό κλειδί που παράγει κάθε φορά είναι μεγέθους 128 bit, το οποίο για να σπάσει μπορεί να αναγκάσει κάποιο λογισμικό που εκτελεί brute force επιθέσεις να προσπαθεί για χρόνια. Βέβαια το κλειδί εξαρτάται από τον κωδικό που θέτει ο χρήστης, επομένως από αυτόν εξαρτάται και ο χρόνος που θα χρειαστεί για να εξαντλήσει τα κλειδιά μια επίθεση brute force. Επίσης, η συχνότητα με την οποία δοκιμάζει κλειδιά ένα λογισμικό εξαρτάται και από το hardware του υπολογιστή. Έτσι ένας αρκετά δυνατός συμβατικός υπολογιστής μπορεί να κάνει brute force έναν 8ψήφιο κωδικό μόνο από χαρακτήρες της εφαρμογής μας σε 11 ώρες και 40 λεπτά, ο χρόνος ανεβαίνει σε 1 χρόνο και 130 ημέρες αν είναι 8ψήφιος με χαρακτήρες μικρούς και κεφαλαίους και αριθμούς, σε 37 χρόνια και 265 ημέρες αν περιέχει και κενά και όλα τα σύμβολα και τέλος 333 χιλιάδες χρόνια αν περιέχει όλα τα προηγούμενα αλλά είναι 10ψήφιος ο κωδικός. Ένας ρεαλιστικός κωδικός για έναν άνθρωπο, είναι ένας 10ψήφιος με μόνο μικρούς χαρακτήρες και αριθμούς, ο οποίος θα χρειαστεί περίπου 23 χρόνια για να σπάσει.

Για κάθε μήνυμα που επιθυμεί ο χρήστης να κρυπτογραφήσει πρέπει να εισάγει έναν κωδικό κρυπτογράφησης. Αυτό μπορεί να είναι κάπως κουραστικό, αλλά σχεδιάστηκε με τη λογική ότι ο χρήστης θα θέλει να επιλέγει ποια από τα μηνύματά του είναι αρκετά σημαντικά ώστε να τα κρυπτογραφήσει. Επίσης, υπάρχει το πρόβλημα του να επικοινωνήσει ο αποστολέας στον παραλήπτη τον κωδικό της

κρυπτογράφησης. Εξαρτάται από τον χρήστη ο τρόπος που θα επιλέξει να επικοινωνήσει με ασφάλεια κάποιον κωδικό, με πλέον ενδεδειγμένο τον πρόσωπο με πρόσωπο και να χρησιμοποιούν στην πορεία τον συμφωνημένο κωδικό. Η χρήση του ίδιου κωδικού σε πολλά μηνύματα δεν είναι καλή πρακτική, αλλά ο αλγόριθμος κατακερματισμού (hash algorithm) που χρησιμοποιεί η εφαρμογή παράγει το κρυπτογραφικό κλειδί από τον κωδικό, γεμίζοντας με τυχαία bytes διάφορα σημεία του κλειδιού. Έτσι, ο ίδιος κωδικός παράγει κάθε φορά διαφορετικό κρυπτογραφικό κλειδί, το οποίο κάνει τη χρήση του ίδιου κωδικού πολλαπλές φορές αρκετά ασφαλή.

4.2. Συμβατότητα API

Για να αντιμετωπίσουμε το πρόβλημα της συμβατότητας των API για το SMS και να καλύψουμε όλα τα τηλέφωνα έπρεπε να επιλέξουμε, κάποια πολύ γενική μέθοδο που να μπορεί να χειριστεί τα μηνύματα από το σύστημα για κάθε πιθανή συσκευή που χρησιμοποιεί παλιότερο API (API 18 - Android version 4.3 και παλιότερα) και, επίσης, μια διαφορετική μέθοδο η οποία θα χειρίζεται τα μηνύματα σε συσκευές με νεότερη API (API 19 – Android version 4.4 και άνω). Έτσι χρειάστηκε να γράψουμε τη μέθοδο setConvData ένα κομμάτι της είναι το :

```
public void setConvData() {
      ContentResolver contentResolver = getContentResolver();
      //final String[] projection = new String[]{"*"};Uri uri =
      Uri.parse("content://sms/");
      Cursor smsConvCursor = contentResolver.query(uri, new String[]
{"DISTINCT thread_id","body","address", "type", "date"},"thread_id IS
NOT NULL) GROUP BY (thread_id", null,"DATE desc");
      int indexBody = smsConvCursor.getColumnIndex("body");
      int indexAddress = smsConvCursor.getColumnIndex("address");
      int indexThread = smsConvCursor.getColumnIndex("thread_id");
      int indexDate = smsConvCursor.getColumnIndex("date");
      int indexType = smsConvCursor.getColumnIndex("type");
```

Στη μέθοδο αυτή έχουμε πάρει το συνολικό URI content://sms/ που περιέχει όλα τα μηνύματα και κάνουμε μια επιλογή παρόμοια με αυτήν που θα έκανε αν υπήρχε το

content://sms/conversations, μέσω του SQL query `{"DISTINCT thread_id","body","address", "type", "date"},"thread_id IS NOT NULL) GROUP BY (thread_id", null,"DATE desc"` το οποίο επιλέγει τα τελευταία, χρονικά, μηνύματα που έχουν το ίδιο thread_id και τα κατηγοριοποιεί σύμφωνα με το thread_id και σύμφωνα με την ημερομηνία αποστολής τους.

Αυτή η μέθοδος, παρόλο που δεν δημιουργούσε κάποιο σφάλμα σε συσκευές με νεότερο API, δηλαδή API 19 και άνω, δεν είχε τα επιθυμητά αποτελέσματα. Για παράδειγμα τα εισερχόμενα και τα εξερχόμενα προς την ίδια επαφή εμφανίζονταν σε διαφορετικά conversations ή εμφανίζονταν όλα τα conversations δύο φορές. Ήταν λογικό να μη δημιουργεί σφάλμα η παλιότερη μέθοδος, λόγω της προς τα εμπρός συμβατότητας του Android, όπως επίσης και να χρειάζεται μια διαφορετική μέθοδο που να χρησιμοποιεί το καινούριο API, για να αντιμετωπίζει τα προβλήματα που εμφανίζονται από την αποδοκιμασμένη πλέον μέθοδο. Για αυτόν τον λόγο γράψαμε και τη μέθοδο setConvDataAPI19, της οποίας ένα μέρος είναι το:

```
public void setConvDataAPI19() {
	ContentResolver contentResolver = getContentResolver();
	Cursor smsConvCursor19 = contentResolver.query(Telephony.Sms.CONTENT_URI, // Official CONTENT_URI from docs
	new String[] {"DISTINCT thread_id","body","address", "type", "date"},"thread_id IS NOT NULL) GROUP BY (thread_id", null,Telephony.Sms.Inbox.DEFAULT_SORT_ORDER);
	int indexBody = smsConvCursor19.getColumnIndex("body");
	int indexAddress = smsConvCursor19.getColumnIndex("address");
	int indexThread = smsConvCursor19.getColumnIndex("thread_id");
	int indexDate = smsConvCursor19.getColumnIndex("date");
	int indexType = smsConvCursor19.getColumnIndex("type");
```

Αυτή η μέθοδος χρησιμοποιεί ένα παρόμοιο SQL query για την επιλογή των μηνυμάτων, με τη διαφορά ότι χρησιμοποιεί το URI Telephony.Sms που είναι μέρος του επίσημου API του Android πλέον. Το URI Telephony.Sms περιέχει την class Telephony.Sms.Conversations που περιέχει τα μηνύματα που χρειάζεται για την απεικόνιση των συνομιλιών με τον τρόπο που θέλαμε. Όμως τα πεδία που

υπάρχουν στη βάση για αυτήν την κλάση δεν είναι αυτά που χρειαζόμαστε, γι' αυτό χρησιμοποιήσαμε μια παρόμοια με προηγουμένως προσέγγιση, το SQL query στο γενικότερο Telephony.Sms URI.

```
"DISTINCT thread_id","body","address", "type", "date"},"thread_id IS
NOT NULL) GROUP BY (thread_id",
null,Telephony.Sms.Inbox.DEFAULT_SORT_ORDER);
```

Η εφαρμογή επιλέγει ποια από τις δύο μεθόδους θα χρησιμοποιήσει ανάλογα με την έκδοση του Android που τρέχει στη συσκευή, με τη μέθοδο:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT){
      setConvDataAPI19();
      }else{
      setConvData();
      }
```

Το κομμάτι που χειρίζεται τα μηνύματα για το Chat χρησιμοποιεί μια παρόμοια μέθοδο, την getSMSChatData :

```
public void getSMSChatData(String threadid, String phoneno) {

        ContentResolver contentResolver = getContentResolver();
        Cursor smsChatCursor =
contentResolver.query(Uri.parse("content://sms/"), null, null, null,
"DATE desc");
        int indexBody = smsChatCursor.getColumnIndex("body");
        int indexAddress = smsChatCursor.getColumnIndex("address");
        int indexThread = smsChatCursor.getColumnIndex("thread_id");
        int indexDate = smsChatCursor.getColumnIndex("date");
        int indexType = smsChatCursor.getColumnIndex("type");
        int indexId = smsChatCursor.getColumnIndex("_id");

        if (indexBody < 0 || !smsChatCursor.moveToLast()) {
            return;
        }
```

4.3. Επαφές

Η εφαρμογή μας χρησιμοποιεί την βασική εφαρμογή για τις επαφές της συσκευής για να επιλέγει ο χρήστης σε ποιόν θέλει να στείλει μήνυμα. Αυτό γίνεται

στέλνοντας ένα Intent στο σύστημα, ότι η εφαρμογή μας θέλει να αποκτήσει πρόσβαση στην εφαρμογή των επαφών με την μέθοδο:

```
public void onClick(View v) {
	Intent intent = new Intent(Intent.ACTION_PICK,
	ContactsContract.Contacts.CONTENT_URI);
	startActivityForResult(intent, REQUEST_CODE_PICK_CONTACTS
/**PICK_CONTACT**/);
	}
```

Αυτή η μέθοδος χρησιμοποιείται όταν πατήσει ο χρήστης το κουμπί των επαφών, η οποία ανοίγει την εφαρμογή των επαφών. Αφού αποκτήσει πρόσβαση στις επαφές, ο χρήστης επιλέγει μια επαφή και η εφαρμογή επιστρέφει το όνομα και τον αριθμό του τηλεφώνου της επαφής αυτής με τις μεθόδους:

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data){
	super.onActivityResult(requestCode, resultCode, data);
	if (requestCode == REQUEST_CODE_PICK_CONTACTS && resultCode ==
RESULT_OK){
		Log.d(TAG, "Response: " + data.toString());uriContact =
		data.getData();
		retrieveContactName();retrieveContactNumber();
		}
	}
```

Αυτή η μέθοδος καλεί τις μεθόδους retrieveContactNumber και retriveContactName όταν επιλέξει ο χρήστης μια από τις επαφές.

Η μέθοδος retrieveContactNumber:

```
private void retrieveContactNumber() {
	String cNumber = null;// getting contacts ID
	Cursor cursorID = getContentResolver().query(uriContact,
	new String[]{ContactsContract.Contacts._ID},null, null, null);
	if (cursorID.moveToFirst()) {
```

```
            contactID =
      cursorID.getString(cursorID.getColumnIndex(ContactsContract.Con
      tacts._ID));
            }
      cursorID.close();
      Log.d(TAG, "Contact ID: " + contactID);
 // Using the contact ID now we will get contact phone number
      Cursor cursorPhone =
      getContentResolver().query(ContactsContract.CommonDataKinds.Pho
      ne.CONTENT_URI,new
      String[]{ContactsContract.CommonDataKinds.Phone.NUMBER},
      ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = ? AND "
+
      ContactsContract.CommonDataKinds.Phone.TYPE + " = " +
      ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE,new
      String[]{contactID},null);
      if (cursorPhone.moveToFirst()) {
            cNumber =
            cursorPhone.getString(cursorPhone.getColumnIndex(Contacts
            Contract.CommonDataKinds.Phone.NUMBER));
      if(!cNumber.substring(0,GetCountryZipCode().length()+1).equals(
"+"+GetCountryZipCode())){
            numText.setText("+"+GetCountryZipCode()+cNumber.replaceAl
      l("[^0-9\\+]",""));}else{
            numText.setText(cNumber.replaceAll("[^0-9\\+]",""));
            }
      }
      cursorPhone.close();
      Log.d(TAG, "Contact Phone Number: " + cNumber);
}
```

Αυτή η μέθοδος χρησιμοποιεί την nested class ContactsContract.CommonDataKinds.Phone του object ContactsContract.Data, το οποίο περιέχει όλες τις πληροφορίες για του αριθμούς τηλεφώνων των επαφών και επιλέγει τα πεδία:

- NUMBER, Ο αριθμός τηλεφώνου της επιλεγμένης επαφής.
- CONTACT_ID, ο αναγνωριστικός αριθμός της επαφής στο σύστημα.
- TYPE_MOBILE, αν η επαφή έχει αριθμό τηλεφώνου καταχωρημένο ως κινητό.

Η μέθοδος λοιπόν όταν επιλέξει ο χρήστης μία επαφή χρησιμοποιεί το CONTACT_ID για να βρει και να επιστρέψει τον αριθμό του κινητού τηλεφώνου της πίσω στην εφαρμογή μας, με τον κωδικό της χώρας μπροστά αν δεν υπήρχε στην καταχώρηση.

Η μέθοδος retrieveContactName:

```
private void retrieveContactName() {String contactName = null;//
querying contact data store
      Cursor cursor = getContentResolver().query(uriContact, null,
null, null, null);
      if (cursor.moveToFirst()) {// DISPLAY_NAME = The display name
for the contact.
// HAS_PHONE_NUMBER =   An indicator of whether this contact has at
least one phone number.
            contactName =
      cursor.getString(cursor.getColumnIndex(ContactsContract.Contact
      s.DISPLAY_NAME));
      }
      cursor.close();Log.d(TAG, "Contact Name: " + contactName);} }
```

Με αυτή τη μέθοδο η εφαρμογή χρησιμοποιεί το object ContactsContract.Contacts, το οποίο περιέχει όλες τις πληροφορίες των επαφών, εκτός από τους αριθμούς τηλεφώνου, όπως όνομα και φωτογραφία και επιστρέφει το όνομα της επαφής που επιλέγει ο χρήστης.

4.4. Τηλεφωνικοί Κωδικοί Χωρών

Για να αντιμετωπίσουμε το πρόβλημα των τηλεφωνικών κωδικών των χωρών, χρειάστηκε να χρησιμοποιήσουμε μια κάπως χειροκίνητη προσέγγιση. Εισαγάγαμε στην εφαρμογή μια βιβλιοθήκη (library) με τους τηλεφωνικούς κωδικούς κάθε χώρας, στην οποία αποκτά πρόσβαση οποιαδήποτε κλάση χρειάζεται κάποιον κωδικό χώρας. Οι μέθοδοι που διαχειρίζονται αριθμούς τηλεφώνου, όπως η μέθοδος που επιλέγει επαφές, ελέγχουν αν ο αριθμός στον οποίο αποκτάει πρόσβαση περιέχει τον κωδικό της χώρας και αν δεν τον περιέχει, προσθέτει τον κωδικό μπροστά στον αριθμό τηλεφώνου πριν τον χρησιμοποιήσει.

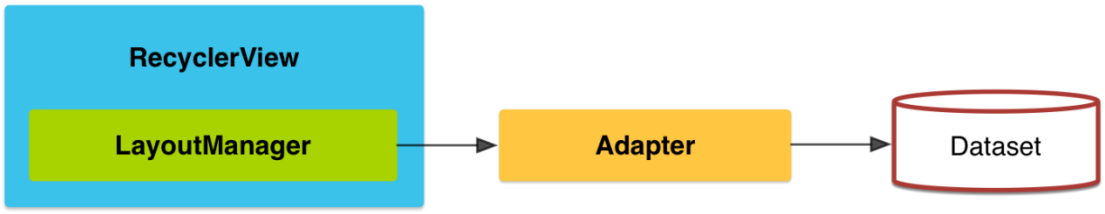
Η μέθοδος που επιστρέφει τον κωδικό της χώρας προς χρήση κάποιας άλλης μεθόδου είναι η GetCountryZipCode:

```
public String GetCountryZipCode(){
      String CountryID="";String CountryZipCode="";
      TelephonyManager       manager       =       (TelephonyManager)
this.getSystemService
      (Context.TELEPHONY_SERVICE);//getNetworkCountryIso
      CountryID= manager.getSimCountryIso().toUpperCase();
      String[]
rl=this.getResources().getStringArray(R.array.CountryCodes);
      for(int i=0;i<rl.length;i++){
      String[] g=rl[i].split(",");
      if(g[1].trim().equals(CountryID.trim())){
            CountryZipCode=g[0];
            break;}
      }
      return CountryZipCode;
}
```

4.5. Λειτουργία Interface (Διαπροσωπία)

Για την εμφάνιση των μηνυμάτων στα conversations και στο chat χρησιμοποιήσαμε το widget RecyclerView[9]. Το RecyclerView λειτουργεί σαν ένα δοχείο για την απεικόνιση μεγάλων σετ δεδομένων (data sets), με τέτοιον τρόπο ώστε να είναι πολύ αποδοτική και ομαλή η μετακίνηση τους προς τα πάνω ή κάτω (scrolling). Αυτό συμβαίνει επειδή διατηρεί έναν περιορισμένο αριθμό απεικονίσεων (views), αυτών που εμφανίζονται στην οθόνη και των κοντινών τους, αντί να κρατάει στη μνήμη δεδομένα τα οποία δεν εμφανίζονται στην οθόνη τη συγκεκριμένη στιγμή ή να δημιουργεί αχρείαστα views και να ψάχνει συνεχώς ids των views για να προβάλει. Το RecyclerView χρησιμοποιεί ένα LayoutManager για να τοποθετεί τα views μέσα στο RecyclerView, το οποίο αποφασίζει πότε να ξαναχρησιμοποιήσει τα views που δεν είναι ορατά από τον χρήστη τη συγκεκριμένη στιγμή. Για να ξαναχρησιμοποιήσει (recycle) ένα view, το LayoutManager ζητάει από το adapter να αντικαταστήσει τα δεδομένα του view με άλλα από το data set που χρειάζεται.

Το RecyclerView χρησιμοποιήθηκε γιατί είναι πιο σύγχρονο από τα υπόλοιπα list widgets, ενώ κρίθηκε αποδοτικότερο από το πιο συνηθισμένο ListView στην εμφάνιση των μηνυμάτων και επίσης γιατί έχει περισσότερες δυνατότητες και εφαρμογές όσον αφορά τη διάταξη της λίστας και την προσπέλασή της.



4.1. Recycler View

[10]Οι adapters χρησιμοποιούνται για να ενώνουν τα δεδομένα με τα View Groups τα οποία επεκτείνουν την κλάση AdapterView, όπως στη περίπτωσή μας το RecyclerView. Οι adapters είναι υπεύθυνοι για τη δημιουργία των child Views τα οποία εκπροσωπούν τα δεδομένα που υπάρχουν μέσα στο κύριο View με το οποίο είναι ενωμένοι. Το Android παρέχει αρκετούς adapters, οι οποίοι καλύπτουν πολλές από τις πιθανές ανάγκες των προγραμματιστών, ώστε να μην χρειάζεται να δημιουργούν δικούς τους.

Στον Adapter μας κάνουμε χρήση του ViewHolder. Το ViewHolder είναι ένα αντικείμενο (object) το οποίο αποθηκεύει τα views που χρειάζεται στο πεδίο Tag του Layout, έτσι ώστε όποτε χρειαστεί να έχει άμεση πρόσβαση σε αυτά χωρίς να χρειάζεται να τα αναζητά κάθε φορά.

Το RecyclerView μαζί με το adapter που χρησιμοποιεί η εφαρμογή μας για να απεικονίζει τα conversations και το chat, είναι το Conversation_Adapter και το Chat_Adapter αντίστοιχα. Θα εστιάσουμε στο Chat_Adapter γιατί έχει περισσότερες δυνατότητες, το Conversation_Adapter βέβαια λειτουργεί με παρόμοιο τρόπο.

Ένα μέρος του Chat_Adapter:

```
public class Chat_Adapter extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    private List<Chat_Message> chatmessagelist;
    private Context context;

    public Chat_Adapter(Context context, List<Chat_Message>list){
        this.context=context;
```

```
        this.chatmessagelist=list;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder (ViewGroup
parent, int viewType) {
        View view;

        switch (viewType) {
            case TYPE_LEFT:
                view =
LayoutInflater.from(context).inflate(R.layout.chat_row, parent,
false);
                return new LeftViewHolder(view);
            case TYPE_RIGHT:
                view =
LayoutInflater.from(context).inflate(R.layout.chat_row_right, parent,
false);
                return new RightViewHolder(view);
        }
        return null;
    }
```

- Η κλάση Chat_Adapter επεκτείνει τον RecyclerView Adapter.
- Δημιουργεί μια λίστα στην οποία καταχωρεί τα αντικείμενα της κλάσης Chat_Message, το οποίο είναι ένα object που δημιουργήσαμε για να περιέχει όλα τα στοιχεία ενός μηνύματος. Αυτά είναι:
    - TYPE_LEFT τοποθετεί το μήνυμα στην αριστερή πλευρά της οθόνης.
    - TYPE_RIGHT τοποθετεί το μήνυμα στην δεξιά πλευρά της οθόνης.
    - Address ο αριθμός τηλεφώνου του παραλήπτη.
    - Message το κείμενο του μηνύματος.
    - Timestamp η ώρα και ημερομηνία που εστάλη το μήνυμα.
    - Type ο τύπος του μηνύματος, 1 για εισερχόμενα 2 για εξερχόμενα.
- Δημιουργεί το context, που είναι μια abstract class που επιτρέπει την πρόσβαση σε resources και classes της εφαρμογής.
- Δημιουργεί την απεικόνιση (view) της λίστας. Αν το αντικείμενο είναι TYPE_LEFT τότε χρησιμοποιεί το layout chat_row, το οποίο είναι ένα XML

αρχείο Layout που δημιουργήσαμε για να απεικονίζουμε μηνύματα στο αριστερό μέρος της οθόνης. Αντίστοιχα για TYPE_RIGHT χρησιμοποιεί το chat_row_right που απεικονίζει το μήνυμα στο δεξί μέρος της οθόνης.

```
public class LeftViewHolder extends RecyclerView.ViewHolder {

        private TextView message;
        private TextView timestamp;

        public LeftViewHolder(View itemView){
            super(itemView);

            Typeface courier =
Typeface.createFromAsset(itemView.getContext().getAssets(),
"cour.ttf");
            message = (TextView)
itemView.findViewById(R.id.chatview);
            message.setTypeface(courier);
            timestamp = (TextView)
itemView.findViewById(R.id.timestamp);
            timestamp.setTypeface(courier);
            itemView.setOnClickListener(new
View.OnClickListener(){
                @Override
                public void onClick(View v) {
                    int mPosition = getAdapterPosition();
                    Chat_Activity sct = (Chat_Activity)context;
                    sct.onItemClick(mPosition);
                }
            });
        }

    }
```

Η κλάση LeftViewHolder διατηρεί τα μηνύματα σε μια λίστα στο viewholder της αριστερής απεικόνισης και δημιουργεί ένα OnClickListener που επιτρέπει στο χρήστη να πατάει πάνω σε ένα μήνυμα. Χωρίς το OnClickListener, αν ένας χρήστης πατήσει πάνω στο μήνυμα στην οθόνη του δεν θα υπάρχει κάποια απόκριση. Αντίστοιχα υπάρχει η κλάση RightViewHolder για τα μηνύματα που βρίσκονται στη

δεξιά πλευρά. Χρειάζονται δύο διαφορετικά view holders για αριστερά και δεξιά γιατί είναι δύο ξεχωριστά views και layouts για την κάθε πλευρά. Διαφορετικά μπορεί να δημιουργηθούν conflicts και να μην λειτουργεί σωστά η απεικόνιση ή ακόμα και καθόλου, ενώ επίσης η εφαρμογή με ξεχωριστά view holders είναι πιο αποδοτική.

Ο adapter για τα conversations είναι παρόμοιος, με τη διαφορά ότι χρησιμοποιεί ένα layout το conversation_row και επομένως ένα view holder. επίσης, χρησιμοποιεί το object, Conversation το οποίο δημιουργεί τα μηνύματα για τα conversations. Το Conversation είναι παρόμοιο με το Chat_Message με κάπως διαφορετικά πεδία, όπως Image για την φωτογραφία της επαφής και Thread για τον αναγνωριστικό αριθμό κάθε συζήτησης.

4.6. Αποστολή Μηνυμάτων SMS

Η εφαρμογή μας χρησιμοποιεί την βασική εφαρμογή της συσκευής για να εκτελέσει την αποστολή των μηνυμάτων SMS. Αυτό γίνεται από την μέθοδο sendChatMessage. Ένα μέρος της είναι:

```
 private void sendChatMessage(String phoneno) {        // Converts
the typed Message to String
        String myMsg = chatText.getText().toString();
        // Converts the typed Password to String
        String myPass = encPass.getText().toString();


        if (!TextUtils.isEmpty(myPass)){

            if (BuildConfig.DEBUG) {
                AESEnc.DEBUG_LOG_ENABLED = true;
            }

            // Performs Encryption of the typed Message using the
typed Password (calls the encryption method from AESEnc class)
            String encryptedMsg = null;
            try {
                encryptedMsg = AESEnc.encrypt(myMsg, myPass);
            } catch (Exception e) {
```

```
                e.printStackTrace();
            }
            chatText.setText("");

            // Sends the Encrypted Message via SMS
            try {
                SmsManager smsManager = SmsManager.getDefault();
                smsManager.sendTextMessage(phoneno, null,
encryptedMsg, null, null);
                Toast.makeText(getApplicationContext(), "SMS sent.",
                        Toast.LENGTH_LONG).show();
                chatlist.add(new Chat_Message(null, encryptedMsg,
"Just Now", Chat_Message.TYPE_RIGHT));
                adapter.notifyItemInserted(adapter.getItemCount());

recyclerView.smoothScrollToPosition(adapter.getItemCount());
                Conversation_Activity conv_inst =
Conversation_Activity.instance();
                conv_inst.updateList(phoneno, encryptedMsg);

                ContentValues values = new ContentValues();
                values.put("address",phoneno); // phone number to
send
                values.put("date", System.currentTimeMillis()+"");
                values.put("read", "1"); // if you want to mark is as
unread set to 0
                values.put("type", "2"); // 2 means sent message
                values.put("body", encryptedMsg);

                Uri uri = Uri.parse("content://sms/");
                Uri rowUri =
getApplicationContext().getContentResolver().insert(uri,values);
```

Αναλύουμε τον κώδικα:

1. Παίρνει το κείμενο που έχει πληκτρολογήσει ο χρήστης στο πεδίο του μηνύματος και στο πεδίο του κωδικού και τα μετατρέπει σε μορφή string.
2. Μέσα από μια δήλωση if, επιλέγει αν θα στείλει κρυπτογραφημένο ή όχι το μήνυμα. Αν το πεδίο του κωδικού δεν είναι κενό τότε θα στείλει

κρυπτογραφημένο μήνυμα, διαφορετικά θα στείλει ένα απλό μήνυμα. Εδώ φαίνεται η περίπτωση του κρυπτογραφημένου μηνύματος.

3. Αφού επιλέξει (εδώ στην περίπτωση που το πεδίου του κωδικού δεν είναι κενό), πραγματοποιεί την κρυπτογράφηση του μηνύματος σύμφωνα με τον κωδικό.
4. Καλεί το SmsManager, τη βασική μέθοδο με την οποία στέλνει τα μηνύματα το Android.
5. Στέλνει το μήνυμα μέσω του SmsManager ορίζοντας τον αριθμό του παραλήπτη και το μήνυμα το οποίο θα σταλεί.
6. Δημιουργεί ένα toast (μήνυμα ένδειξη) που λέει ότι το μήνυμα εστάλη.
7. Δημιουργεί ένα νέο αντικείμενο (object) Chat, που προβάλλει το μήνυμα στη λίστα του Chat, το οποίο περιέχει το κείμενο του μηνύματος, το μήνυμα “Just Now” που δείχνει ότι το μήνυμα μόλις εστάλη και τοποθετεί το μήνυμα στη δεξιά πλευρά της οθόνης.
8. Ειδοποιεί τον adapter ότι προστέθηκε ένα μήνυμα ώστε να κάνει ενημέρωση.
9. Κάνει scroll την οθόνη στο τέλος του adapter όπου τοποθετήθηκε το μήνυμα.
10. Ενημερώνει το Activity των συζητήσεων Conversation_Activity με το μήνυμα.
11. Τοποθετεί το μήνυμα στο σύστημα, ώστε να μπει στη βάση των μηνυμάτων της συσκευής.

4.7. Λήψη Μηνυμάτων SMS

Για να παραλαμβάνει τα μηνύματα η εφαρμογή μας χρησιμοποιεί μια κλάση broadcast receiver:

```
public class SMS_BroadcastReceiver extends BroadcastReceiver {
    public static final String SMS_BUNDLE = "pdus";
   public void onReceive(Context context, Intent intent) {
        Bundle intentExtras = intent.getExtras();
       if (intentExtras != null) {
            Object[] sms = (Object[]) intentExtras.get(SMS_BUNDLE);
```

```
            String smsMessageStr = "";
            String smsAddress= "";
            for (int i = 0; i < sms.length; ++i) {
                SmsMessage smsMessage =
SmsMessage.createFromPdu((byte[]) sms[i]);

                String smsBody =
smsMessage.getMessageBody().toString();
                String address = smsMessage.getOriginatingAddress();

                smsMessageStr += smsBody + "\n";
                smsAddress += address;
            }
            Toast.makeText(context, smsMessageStr,
Toast.LENGTH_SHORT).show();

            //this will update the UI with message

Chat_Activity.getInstance().updateList(smsAddress,smsMessageStr);

Conversation_Activity.getInstance().updateList(smsAddress,
smsMessageStr);
        }
    }
}
```

Το SMS_BroadcastReceiver:

- Δέχεται το intent με τα δεδομένα του μηνύματος σε ένα PDU (Protocol Data Unit) Bundle.
- Τοποθετεί τα δεδομένα στα κατάλληλα strings, δηλαδή το κείμενο στο String smsBody και τον αριθμό του αποστολέα στο String smsAddress.
- Δημιουργεί ένα toast (μήνυμα ειδοποίησης), με το κείμενο του μηνύματος.
- Καλεί τις μεθόδους updateList στο Chat_Activity και στο Conversation_Activity στα οποία περνάει τα Strings με τον αριθμό και το κείμενο του μηνύματος. Αυτές οι μέθοδοι ενημερώνουν την λίστα των

μηνυμάτων με το καινούριο μήνυμα που μόλις έφτασε, σε πραγματικό χρόνο.

4.8. Πρόσθετες Λειτουργίες

Όπως προαναφέρθηκε, εκτός από κρυπτογραφημένα μηνύματα η εφαρμογή μας μπορεί να στείλει και απλά μηνύματα κατ' επιλογήν του χρήστη. Αυτό γίνεται αν απλά ο χρήστης δεν πληκτρολογήσει κωδικό κρυπτογράφησης στο σχετικό πεδίο.

Δεν είναι όλα τα μηνύματα αρκετά σημαντικά για να παραμένουν κρυπτογραφημένα ή μπορεί κάποιος χρήστης να θέλει για ευκολία να μην αποκρυπτογραφεί κάθε φορά κάποιο μήνυμα για να το διαβάσει. Για αυτό έχουμε βάλει την επιλογή να μπορεί να αποθηκεύσει ο χρήστης κάποιο μήνυμα αποκρυπτογραφημένο. Αυτό επιτυγχάνεται με τη μέθοδο:

```
replaceButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {

                if (!decView.getText().toString().equals("")) {
                    long millidate = 0;

                    SimpleDateFormat sdf = new
SimpleDateFormat("dd/MM/yy hh:mm:ss");
                    try {
                        Date date = sdf.parse(timestamp);
                        millidate = date.getTime();
                    } catch (ParseException e) {
                        e.printStackTrace();
                    }

                    if (type == Chat_Message.TYPE_RIGHT) {
                        ContentValues values = new ContentValues();
                        values.put("address", address); // phone
```

```
number to send
                        if (millidate != 0) {
                            values.put("date", millidate);
                        }
                        values.put("read", "1"); // if you want to
mark is as unread set to 0
                        values.put("type", "2"); // 2 means sent
message
                        values.put("body",
decView.getText().toString());

                        Uri uri = Uri.parse("content://sms/");
                        Uri rowUri =
getApplicationContext().getContentResolver().insert(uri, values);

Toast.makeText(getApplicationContext(), "Message will be
stored above the chosen encrypted one",
Toast.LENGTH_LONG).show();
```

Αυτός ο κώδικας ουσιαστικά παίρνει το αποκρυπτογραφημένο μήνυμα από το πεδίο που εμφανίζεται καθώς επίσης και τα υπόλοιπα χαρακτηριστικά του μηνύματος, αριθμό τηλεφώνου, τον τύπο εισερχόμενο ή εξερχόμενο και την ώρα που έφτασε και τα εισάγει στο σύστημα σαν να παρέλαβε κανονικά κάποιο μήνυμα SMS τη συγκεκριμένη ώρα από αυτόν τον αριθμό. Το μήνυμα εμφανίζεται στη λίστα των μηνυμάτων πάνω από το αντίστοιχο κρυπτογραφημένο. Ορισμένα μηνύματα από την άλλη μπορεί να είναι αρκετά σημαντικά ώστε ο χρήστης να θέλει να τα κρατήσει κρυπτογραφημένα για ασφάλεια. Για αυτό υπάρχει η επιλογή να επιλέξει ο χρήστης ένα μήνυμα από την λίστα των μηνυμάτων και να το κρυπτογραφήσει. Αυτό γίνεται με τη μέθοδο:

```
encButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String myPass = encPass.getText().toString();
        String encryptedMsg= null;
        try {
            encryptedMsg = AESEnc.encrypt(msg, myPass);
        } catch (Exception e) {
            e.printStackTrace();
        }

            long millidate = 0;
```

```
            SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy
hh:mm:ss");
            try {
                Date date = sdf.parse(timestamp);
                millidate = date.getTime();
            } catch (ParseException e) {
                e.printStackTrace();
            }

            if (type == Chat_Message.TYPE_RIGHT) {
                ContentValues values = new ContentValues();
                values.put("address", address); // phone number to
send
                if (millidate != 0) {
                    values.put("date", millidate);
                }
                values.put("read", "1"); // if you want to mark is as
unread set to 0
                values.put("type", "2"); // 2 means sent message
                values.put("body", encryptedMsg);

                Uri uri = Uri.parse("content://sms/");
                Uri rowUri =
getApplicationContext().getContentResolver().insert(uri, values);
                Toast.makeText(getApplicationContext(), "Message will
be stored above the chosen message, you should delete the original",
Toast.LENGTH_LONG).show();

Chat_Activity.getInstance().notifyInsertedSms(position, address,
encryptedMsg, timestamp, id, Chat_Message.TYPE_RIGHT);
```

Αυτός ο κώδικας ενεργοποιείται όταν ο χρήστης κρατήσει πατημένο το μήνυμα που επιθυμεί. Μετά μπορεί να πληκτρολογήσει τον κωδικό κρυπτογράφησης. Τότε κρυπτογραφείται το κείμενο του μηνύματος και εισάγεται στο σύστημα όπως και στη προηγούμενη περίπτωση και στη συνέχεια ενημερώνει την λίστα των μηνυμάτων με το καινούριο μήνυμα σε πραγματικό χρόνο.

Πολλές φορές θέλει ένας χρήστης να σβήσει ένα μήνυμα. Αυτή η επιλογή επίσης υπάρχει με την μέθοδο :

```
delButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {


getApplicationContext().getContentResolver().delete(Uri.parse("conten
t://sms/" + id), null, null);
        Chat_Activity.getInstance().notifyDeletedSms(position);
    }
});
```

Αυτή η μέθοδος δέχεται, όταν πατήσει ο χρήστης το μήνυμα που επιθυμεί, έναν μοναδικό αναγνωριστικό αριθμό του μηνύματος (id). Τότε διαγράφει το μήνυμα που αντιστοιχεί σε αυτόν τον αριθμό από το σύστημα και ενημερώνει τη λίστα ότι το μήνυμα στη θέση που επέλεξε ο χρήστης διαγράφτηκε σε πραγματικό χρόνο.

Οι τελευταίες μέθοδοι που περιγράψαμε εισάγουν ή διαγράφουν μηνύματα στο σύστημα. Αυτό γίνεται χωρίς πρόβλημα στις συσκευές που χρησιμοποιούν API 18 (Android version 4.3 Jellybean) και πίσω. Σε πιο καινούριες συσκευές με API 19 (Android version 4.4 Kitkat) και άνω, το Android δεν επιτρέπει σε εφαρμογές που δεν έχουν επιλεχθεί ως οι βασικές εφαρμογές SMS για τη συσκευή να επηρεάζουν τα μηνύματα στο σύστημα. Για αυτό ορίσαμε στο αρχείο manifest όλα τα δικαιώματα που χρειάζεται η εφαρμογή για να μπορεί να οριστεί ως βασική. Την επιλογή πρέπει να την κάνει ο χρήστης από τις ρυθμίσεις της συσκευής.

Αυτά είναι:

```
<activity
    android:name=".Chat_Activity"
    android:label="SMSEncryptionApp">
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <action android:name="android.intent.action.SENDTO" />

        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />

        <data android:scheme="sms" />
        <data android:scheme="smsto" />
        <data android:scheme="mms" />
        <data android:scheme="mmsto" />
    </intent-filter>
<receiver
    android:name=".SMS_BroadcastReceiver"
    android:exported="true"
    android:permission="android.permission.BROADCAST_SMS">
    <intent-filter android:priority="999">
        <action android:name="android.provider.Telephony.SMS_DELIVER"
/>
    </intent-filter>
</receiver>

<service
    android:name=".smssendservice"
    android:exported="true"
    android:permission="android.permission.SEND_RESPOND_VIA_MESSAGE">
    <intent-filter>
        <action
android:name="android.intent.action.RESPOND_VIA_MESSAGE" />
```

```
        <category android:name="android.intent.category.DEFAULT" />

        <data android:scheme="sms" />
        <data android:scheme="smsto" />
        <data android:scheme="mms" />
        <data android:scheme="mmsto" />
    </intent-filter>
</service>

<receiver
    android:name=".smsemptyreceiver"
    android:enabled="true"
    android:exported="true"
    android:permission="android.permission.BROADCAST_WAP_PUSH">
    <intent-filter>
        <action
android:name="android.provider.Telephony.WAP_PUSH_DELIVER" />
        <data android:mimeType="application/vnd.wap.mms-message" />
    </intent-filter>
</receiver>
```

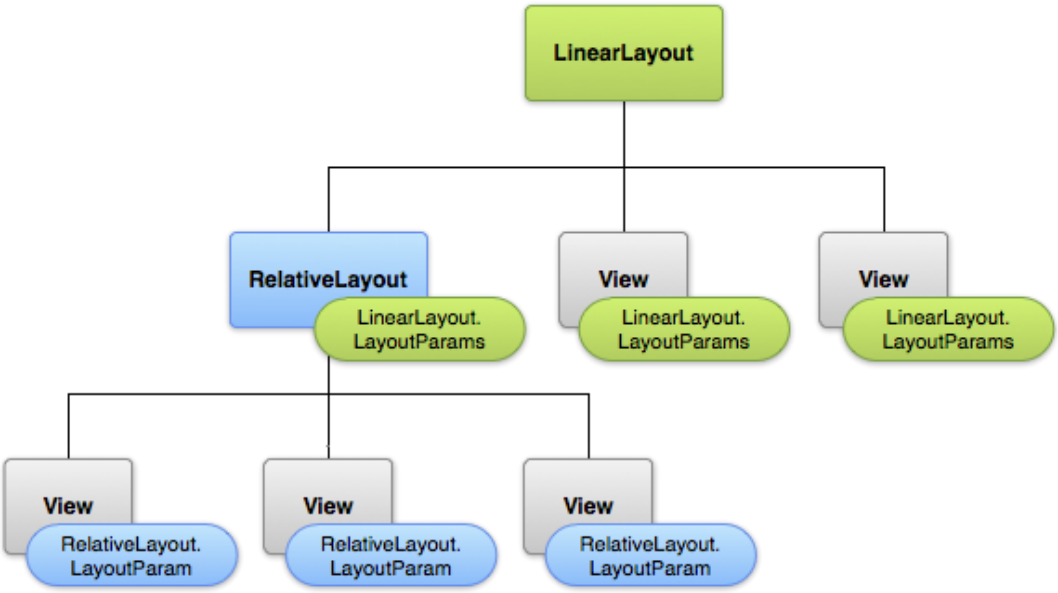
5. Interface (Διαπροσωπία)

5.1. Βασικά Χαρακτηριστικά

[10]Το Layout της εφαρμογής, όπως και κάθε εφαρμογής Android, είναι γραμμένο σε XML, με την οποία ορίζουμε τα διάφορα πεδία που εμφανίζονται στην οθόνη. Τα βασικότερα στοιχεία της XML στο Android που μας ενδιαφέρουν είναι:

- Linear Layout, μία διάταξη που οργανώνει τα παιδιά σε μια οριζόντια ή κάθετη σειρά.
- Relative Layout, μια διάταξη που επιτρέπει στον προγραμματιστή να ορίζει την τοποθεσία των παιδιών του σχετικά με το που βρίσκονται τα υπόλοιπα.
- List View, μια λίστα μιας στήλης, στην οποία μπορεί ο χρήστης να μετακινείται πάνω κάτω (scroll).
- Text View, ένα πεδίο για την απεικόνιση κειμένου.
- Edit Text, ένα πεδίο για την εισαγωγή κειμένου από τον χρήστη.
- Button, ένα κουμπί που μπορεί να πατήσει ο χρήστης.
- Image View, ένα πεδίο για την απεικόνιση εικόνων.
- Support.v7.widget.RecyclerView, ένα καινούριο widget που προσθέσαμε στην εφαρμογή, το οποίο επιτρέπει τη δημιουργία πολύπλοκων λιστών.

Μια αναπαράσταση είναι:



5.1 Linear Layout

Τα πεδία που αναφέραμε ορίζονται για περαιτέρω χρήση με κάποιο όνομα και παίρνουν τις ιδιότητες που τους προσθέτουμε μέσα, όπως:

- android:id, ορίζει το όνομα του πεδίου.
- android:layout_width, μήκος.
- android:layout_height, ύψος.
- android:textColor, χρώμα κειμένου.
- android:layout_margin, περιθώρια προς κάθε κατεύθυνση.
- android:background, φόντο.
- android:orientation, κατεύθυνση της διάταξης

5.1. Conversations

Ξεκινώντας την εφαρμογή ο χρήστης βλέπει στην οθόνη το Activity με τα conversations (συζητήσεις). Το Layout του ξεκινάει με ένα Linear Layout μέσα στο οποίο περιέχονται δύο ξεχωριστά Relative Layouts που το καθένα περιέχει διάφορα πεδία.

5.2 Conversations

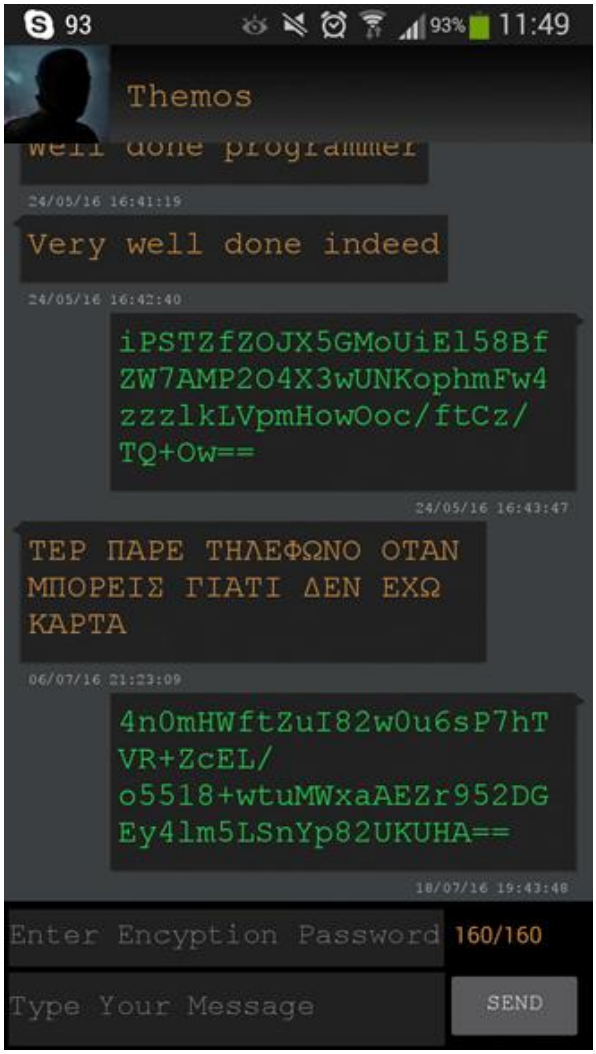
Με την σειρά η οθόνη είναι ως εξής:

- Header (επικεφαλίδα) είναι ένα πεδίο Text View που γράφει Conversations.
- Ένα πεδίο Edit Text στο οποίο πληκτρολογεί ο χρήστης τον αριθμό τηλεφώνου του παραλήπτη, στο πεδίο αυτό έχουμε ορίσει την ιδιότητα `android:inputType="phone"` με την οποία ο χρήστης μπορεί να εισάγει μόνο αριθμούς.
- Δεξιά από το πεδίο του τηλεφώνου είναι ένα κουμπί με τη φωτογραφία ενός άγνωστου κεφαλιού, το οποίο είναι το κουμπί των επαφών.
- Δεξιά από το κουμπί των επαφών είναι ένα κουμπί με τη φωτογραφία ενός μολυβιού, το οποίο είναι το κουμπί σύνθεσης νέου μηνύματος.

- Κάτω από αυτά τα τρία πεδία είναι η λίστα με τις συζητήσεις ανά επαφή σε φθίνουσα χρονολογική σειρά, αυτό είναι ένα Recycler View widget. Κάθε σειρά της λίστας αποτελεί ένα ξεχωριστό layout που είναι το αρχείο conversation_row.xml. Τα πεδία του είναι:
    - Ένα Image View που περιέχει τη φωτογραφία της επαφής, εφόσον υπάρχει, διαφορετικά μια φωτογραφία ενός άγνωστου προσώπου.
    - Στο πάνω μέρος της σειράς υπάρχει ένα Text View, στο οποίο αναγράφεται το όνομα της επαφής με την οποία γίνεται η κάθε συζήτηση, εφόσον αυτό υπάρχει στις επαφές του χρήστη, διαφορετικά φαίνεται ο αριθμός τηλεφώνου της συζήτησης.
    - Από κάτω υπάρχει ένα Text View στο οποίο εμφανίζεται ένα μέρος του τελευταίου μηνύματος της συζήτησης.
    - Στα δεξιά της κάθε σειράς υπάρχει ένα Text View στο οποίο εμφανίζεται η ημερομηνία και η ώρα που έχει ληφθεί ή σταλεί το τελευταίο μήνυμα της συζήτησης.

5.2. Chat

Η δεύτερη οθόνη της εφαρμογής είναι το chat. Εδώ η διάταξη είναι κάπως περισσότερο σύνθετη. Ολόκληρο το αρχείο περιλαμβάνεται σε ένα Relative Layout αντί για Linear.



5.3 Chat

Η οθόνη με τη σειρά είναι:

- Μια επικεφαλίδα που περιέχει:
    - Ένα Image View που περιέχει τη φωτογραφία της επαφής με την οποία συνομιλεί ο χρήστης, εφόσον έχει φωτογραφία στην επαφή.
    - Ένα Text View που περιέχει το όνομα της επαφής με την οποία συνομιλεί ο χρήστης, εφόσον υπάρχει το όνομα στις επαφές, διαφορετικά δείχνει τον αριθμό του τηλεφώνου της επαφής.
- Κάτω από την επικεφαλίδα είναι η λίστα με τα μηνύματα από και προς την επαφή, σε φθίνουσα χρονολογική σειρά. Αυτό είναι ένα Recycler View widget. Κάθε σειρά της λίστας αποτελεί ένα ξεχωριστό layout το οποίο

εξαρτάται από το αν το μήνυμα είναι εισερχόμενο ή εξερχόμενο. Τα μηνύματα στην αριστερή πλευρά με το πορτοκαλί κείμενο είναι τα εισερχόμενα και ορίζονται από το αρχείο chat_row.xml, ενώ τα μηνύματα στη δεξιά πλευρά με το πράσινο κείμενο είναι τα εξερχόμενα και ορίζονται από το αρχείο chat_row_right.xml. Εκτός από την τοποθεσία και το χρώμα τα δύο layouts δεν έχουν άλλες διαφορές. Τα πεδία τους είναι:

  - Ένα Text View που περιέχει το κείμενο του μηνύματος. Μέσα στο Text View έχουμε ορίσει το `android:background="@drawable/bubble_left_grey"` το οποίο είναι το bubble μέσα στο οποίο περιέχεται το μήνυμα.
  - Από κάτω ένα Text View που περιέχει την ημερομηνία και ώρα που λήφθηκε ή στάλθηκε το μήνυμα.

- Κάτω από τη λίστα των μηνυμάτων είναι, ένα Edit Text στο οποίο ο χρήστης μπορεί να πληκτρολογήσει τον κωδικό με τον οποίο θα κρυπτογραφήσει ένα καινούριο μήνυμα. Μέσα σε αυτό το πεδίο έχουμε ορίσει την ιδιότητα `android:inputType="textPassword"` με την οποία ό,τι πληκτρολογεί ο χρήστης μέσα στο πεδίο μετατρέπεται σε κουκίδα ώστε να μην φαίνεται.
- Δεξιά από το πεδίο του κωδικού είναι ένα Text View στο οποίο αναγράφεται ο αριθμός των χαρακτήρων που απομένουν στο μήνυμά μας.
- Κάτω από το πεδίο του κωδικού είναι ένα Edit Text στο οποίο ο χρήστης μπορεί να πληκτρολογήσει το μήνυμα που θέλει να στείλει.
- Δεξιά από το πεδίο του μηνύματος είναι το κουμπί με το οποίο εκτελεί η εφαρμογή την αποστολή του μηνύματος.

5.3. Decrypt Window

Όταν θέλει να αποκρυπτογραφήσει κάποιο μήνυμα ο χρήστης, ανοίγει η τρίτη οθόνη της εφαρμογής. Αυτό είναι ένα παράθυρο που ανοίγει πάνω από την οθόνη του Chat.

5.4 Το παράθυρο αποκρυπτογράφησης

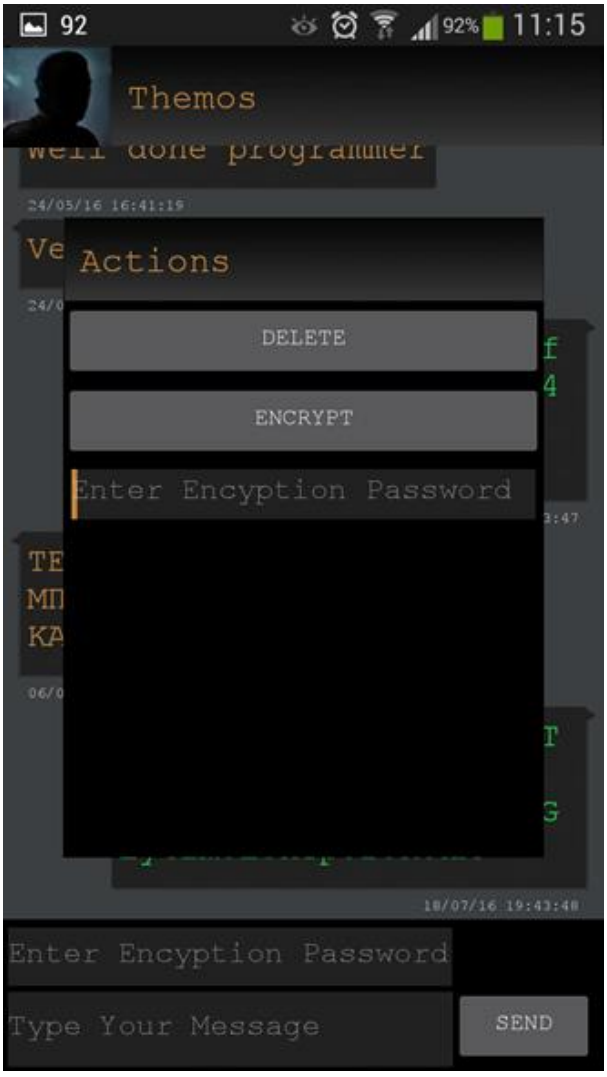
Η οθόνη με τη σειρά είναι:

- Η επικεφαλίδα, ένα Text View που αναγράφει Decrypt.
- Ένα πεδίο Edit Text στο οποίο μπορεί ο χρήστης να πληκτρολογήσει τον κωδικό αποκρυπτογράφησης του μηνύματος. Μέσα σε αυτό το πεδίο έχουμε ορίσει την ιδιότητα `android:inputType="textPassword"` με την οποία ό,τι πληκτρολογεί ο χρήστης μέσα στο πεδίο μετατρέπεται σε κουκίδα ώστε να μην φαίνεται.
- Ένα κουμπί που γράφει decrypt το οποίο ξεκινάει την διαδικασία της αποκρυπτογράφησης.

- Ένα κενό πεδίο Text View στο οποίο απεικονίζεται το μήνυμα αποκρυπτογραφημένο όταν πατήσει ο χρήστης το κουμπί της αποκρυπτογράφησης.
- Ένα κουμπί που γράφει Store Message, το οποίο αποθηκεύει το αποκρυπτογραφημένο μήνυμα στο Chat.

5.4. Actions Window

Ο χρήστης έχει κάποιες πρόσθετες επιλογές, όπως να διαγράψει ή να κρυπτογραφήσει κάποιο ήδη υπάρχον μήνυμα. Για να το κάνει αυτό ανοίγει το παράθυρο Actions.



5.5 Το παράθυρο των πρόσθετων λειτουργιών

Η οθόνη με τη σειρά είναι:

- Η επικεφαλίδα, ένα Text View που αναγράφει Actions.
- Ένα κουμπί που γράφει DELETE το οποίο διαγράφει το επιλεγμένο μήνυμα.
- Ένα κουμπί που γράφει ENCRYPT το οποίο κρυπτογραφεί το επιλεγμένο μήνυμα.
- Ένα πεδίο Edit Text, στο οποίο ο χρήστης πληκτρολογεί τον κωδικό κρυπτογράφησης. Μέσα σε αυτό το πεδίο έχουμε ορίσει την ιδιότητα `android:inputType="textPassword"` με την οποία ό,τι πληκτρολογεί ο χρήστης μέσα στο πεδίο μετατρέπεται σε κουκίδα ώστε να μη φαίνεται.

6. Κρυπτογραφία

6.1. Ιστορική Αναδρομή [11]

Η ανάγκη του ανθρώπου να κρατήσει μυστικά τα μηνύματα από οποιονδήποτε άλλον εκτός από τον παραλήπτη είναι σχεδόν τόσο παλιά όσο και ο πολιτισμός.

Το πρώτο κρυπτογραφημένο μήνυμα που έχει βρεθεί προέρχεται από την περιοχή της Μεσοποταμίας του 1500 π.Χ. Ήταν σκαλισμένο σε μια ταμπλέτα και αφορούσε τη μυστική συνταγή του γυαλίσματος των κεραμικών, μια τέχνη που η εμπορική της αξία την έκαναν τόσο σημαντική ώστε να χρειαστεί να διαφυλαχτεί η διάδοσή της. Αυτή η πρώτη μέθοδος κρυπτογραφίας ήταν μια πολύ βασική μέθοδος *αντικατάστασης*: Αντικαθιστούσαν δηλαδή ένα ιερογλυφικό με κάποιο άλλο.

Στη Βίβλο παρατηρείται μια τροποποιημένη μέθοδος από τους Εβραίους: Αντικαθιστούσαν το τελευταίο γράμμα της αλφαβήτου με το πρώτο, το προτελευταίο με το δεύτερο κ.ο.κ.

Στην Ελλάδα τον 5^{ου} αιώνα π.Χ., όπως μας πληροφορεί ο Ηρόδοτος, ο εξόριστος βασιλιάς της Σπάρτης Δημάρατος έστειλε ένα κρυπτογραφημένο μήνυμα στους Σπαρτιάτες για την επικείμενη εκστρατεία του Ξέρξη, γράφοντας το μήνυμα απ' ευθείας πάνω στο ξύλο μιας πινακίδας και περνώντας μετά το κερί από πάνω ώστε η πινακίδα να φαίνεται κενή. Αυτή η μέθοδος ανήκει στην κατηγορία της *στεγανογραφίας*. Μια παρόμοια μέθοδος στεγανογραφίας που ωστόσο δεν συγκαταλέγεται στη στενή κατηγορία της κρυπτογραφίας, περιγράφεται πάλι από τον Ηρόδοτο. Ξύριζαν το κεφάλι ενός δούλου, ο αποστολέας έγραφε το μήνυμα στο ξυρισμένο δέρμα του κρανίου και στη συνέχεια έστελναν τον δούλο στον παραλήπτη αφού τα μαλλιά του είχαν μεγαλώσει πάλι.

Μια άλλη μέθοδος που χρησιμοποιούσαν οι Έλληνες εκείνη την εποχή ανήκε στη μέθοδο της *μετάθεσης* όπου χρησιμοποιείται και η πρώτη κρυπτογραφική συσκευή. Επρόκειτο για μια σκυτάλη (ένα ξύλινο ραβδί γύρω από το οποίο τυλίγεται ένα δέρμα ή μια περγαμηνή), πάνω στην οποία ο αποστολέας έγραφε το μήνυμα, στη

συνέχεια τύλιγε το δέρμα και το οποίο όταν ξετυλιγόταν εμφάνιζε γράμματα χωρίς κάποιο νόημα. Το δέρμα έπρεπε να ξανατυλιχτεί από τον παραλήπτη σε μια ίδια σκυτάλη για να μπορέσει να διαβαστεί.

Ακόμα και στην Ινδία, τον 4^{ο} αιώνα π.Χ. αναφέρεται σε βιβλίο του Κάμα Σούτρα η ανάγκη να εξασκούνται άνδρες και γυναίκες στην κρυπτογραφία.

Στη Ρωμαϊκή εποχή ο Ιούλιος Καίσαρας διακρίθηκε στην κρυπτογραφία, μάλιστα υπήρχε μια ολόκληρη σχετική πραγματεία του Βαλέριου Πρόβου η οποία δυστυχώς δεν έχει διασωθεί. Ευρήματα δείχνουν πως χρησιμοποιούσε τη μέθοδο της *αντικατάστασης*, αντικαθιστώντας ένα γράμμα του Λατινικού αλφαβήτου με ένα γράμμα του Ελληνικού. Μια άλλη από τις κωδικοποιήσεις του ήταν πάλι με τη μέθοδο της *αντικατάστασης*. Αντικαθιστούσε τα γράμματα του λατινικού αλφαβήτου με τα γράμματα από 3 θέσεις δεξιά. Πρόκειται για μια εξίσωση C=p+3mod26, όπου p ο αριθμός που αντιστοιχεί στο γράμμα στο κανονικό κείμενο και C ο αριθμός που αντιστοιχεί στο γράμμα στο κρυπτογραφημένο κείμενο. Προφανώς το αντίστροφο του είναι το p=C-3mod26. Το 3 στη προκειμένη περίπτωση αποτελεί το κρυπτογραφικό κλειδί του συστήματος αυτού.

Η ανάγκη για κρυπτογραφημένα μηνύματα έγινε ακόμα πιο επιβεβλημένη την εποχή της Αναγέννησης στην Ιταλία. Η ύπαρξη πολλών κρατιδίων με διαρκείς πολιτικές εντάσεις οδήγησαν την κρυπτογραφία σε νέα επίπεδα. Το 1518 εκδίδεται το πρώτο έντυπο βιβλίο κρυπτογραφίας, η «Πολυγραφία» του Τριθεμίου, η οποία περιέχει ένα σύστημα όπου τοποθετείται στην πρώτη γραμμή ενός πίνακα το λατινικό αλφάβητο και στη πρώτη στήλη το απλό κείμενο που είναι προς κρυπτογράφηση. Οι υπόλοιπες γραμμές περιέχουν κυκλικές μεταθέσεις της πρώτης γραμμής κατά 0,1,2,...,25 θέσεις. Το κρυπτογραφημένο κείμενο παράγεται από την τομή της πρώτης γραμμής και στήλης στον πίνακα.

Μια από τις πρώτες συστηματικές εργασίες καταγραφής και μελέτης των τρόπων κρυπτογράφησης και αποκρυπτογράφησης είναι του Άραβα μαθηματικού Al-Kindi με τίτλο «Περί Αποκρυπτογράφησης Κρυπτογραφημένης Αλληλογραφίας» ο οποίος έζησε τον 9^{ο} αιώνα μ.Χ. Η μελέτη του χρησιμοποιείτο μέχρι τις αρχές του 20^{ου} αιώνα και πάνω της βασίστηκε η σύγχρονη κρυπτογράφηση. Το 1918 ο Arthur Scherbius

ανέπτυξε το γνωστό Αίνιγμα το οποίο θυμίζει στη λειτουργία του μια παραλλαγή των δίσκων του Αλμπέρτι. Το Αίνιγμα αποτελείται από ένα πληκτρολόγιο στο οποίο πληκτρολογείται το απλό κείμενο προς κρυπτογράφηση, μια αναδιατακτική μονάδα δίσκων, που αποτελεί το κύριο μέρος του μηχανισμού, η οποία κρυπτογραφεί κάθε γράμμα του απλού κειμένου σε κάποιο αντίστοιχο του κρυπτογραφημένου κειμένου και έναν ηλεκτρικό πίνακα στον οποίο εμφανίζεται το αντίστοιχο γράμμα του κρυπτογραφημένου κειμένου. Το Αίνιγμα χρησιμοποιούσε ο Γερμανικός στρατός κατά τον $2^{ο}$ παγκόσμιο πόλεμο για τις επικοινωνίες του, το οποίο έσπασε, στην αρχική του μορφή το 1938, η πολωνική ομάδα κρυπταναλυτών με την βοήθεια του Marian Rejewski. Αργότερα προστέθηκαν και άλλοι δίσκοι στο Αίνιγμα φτάνοντας τους 8 από τους οποίους χρησιμοποιούνταν οι 3 και η διαδικασία της κρυπτογράφησης έγινε πιο πολύπλοκη στον βαθμό που να έχει πάνω από 10^{15} πιθανά κλειδιά. Κατά τη διάρκεια του πολέμου μια ομάδα κρυπταναλυτών στο Bitchley Park του Backinghamshire με τη βοήθεια του Alan Turing, κατάφερε τελικά να σπάσει το πιο σύνθετο Αίνιγμα.

Το 1976 δημοσιεύτηκε το “New Directions in Cryptography” από τους Diffie και Hellman. Σε αυτήν την διατριβή ακούστηκε για πρώτη φορά η ιδέα της κρυπτογραφίας δημοσίου κλειδιού καθώς και ένας τρόπος για την ανταλλαγή κλειδιών, ο οποίος είναι βασισμένος στη δυσκολία του προβλήματος του διακριτού λογαρίθμου. Τότε δεν είχαν βρει ακόμα κάποια πρακτική εφαρμογή για αυτό το είδος κρυπτογράφησης, αλλά ξεκίνησε ένα καινούριο ρεύμα που οδήγησε στην ανακάλυψη του RSA το 1978 από τους Rivest, Shamir και Aldeman, που αποτελεί την πρώτη πρακτική εφαρμογή της κρυπτογράφησης δημοσίου κλειδιού, η οποία βασίζεται στο πρόβλημα παραγοντοποίησης μεγάλων αριθμών και του συστήματος ElGamal, από τον ElGamal το 1985, το οποίο επίσης βασίζεται στο πρόβλημα του διακριτού λογαρίθμου.

6.2. Κρυπτογραφία [11][20]

Η κρυπτογραφία είναι η επιστήμη και η μελέτη μυστικών γραφών μέσα από μαθηματικές τεχνικές. Ένα απλό κείμενο (plaintext ή cleartext) μετατρέπεται σε ένα κρυπτογραφημένο κείμενο (ciphertext ή cryptogram) μέσα από κάποια μυστική

μέθοδο, το κρυπτογράφημα (cipher) (θα χρησιμοποιούμε τον όρο cipher για να αποφύγουμε κάποια πιθανή σύγχυση με τους υπόλοιπους όρους).

Η διαδικασία της κρυπτογράφησης γενικά μπορεί να αναπαρασταθεί από την εξίσωση:

$$C=T_K(M)$$

Όπου:

- C είναι το κρυπτογραφημένο κείμενο
- T ένας μετασχηματισμός το Cipher
- K είναι το κλειδί της κρυπτογράφησης
- M το απλό κείμενο

Υπάρχουν δύο βασικοί τύποι ciphers.

1. *Μετάθεσης*, οι οποίοι αναδιατάσσουν τα bits ή τους χαρακτήρες από τα δεδομένα, όπως για παράδειγμα το rail-fence cipher (ή κρυπτογράφηση συρματοπλέγματος), με το οποίο οι χαρακτήρες του απλού κειμένου γράφονται με τρόπο που θυμίζουν το πάνω μέρος ενός φράχτη και μετά ξαναγράφεται με τους χαρακτήρες από κάθε σειρά. Μια απεικόνιση είναι:

Το απλό κείμενο ΤΙΠΟΤΑ ΚΑΙΝΟΥΡΙΟ γράφεται:

Τ Τ Ι Ρ

Ι Ο Α Α Ν Υ Ι

Π Κ Ο Ο

Και αναδιατάσσεται σε ΤΤΙΡΙΟΑΑΝΥΙΠΚΟΟ.

2. *Αντικατάστασης*, οι οποίοι αντικαθιστούν τα bits ή τους χαρακτήρες των δεδομένων με κάποια άλλα. Ένας τέτοιος τύπος είναι το cipher του Καίσαρα που προαναφέραμε, το οποίο αντικαθιστά τους χαρακτήρες του λατινικού αλφάβητου με αυτούς από 3 θέσεις δεξιά. Άρα το απλό κείμενο “may the force be with you” γίνεται “pdb wkh irufh eh zlwk brx”.

Ο στόχος της κρυπτογραφίας είναι να εξασφαλίσει την ασφάλεια των πληροφοριών. Αυτό επιτυγχάνεται με τις διαδικασίες:

- *Εμπιστευτικότητας (Confidentiality)*, όπου η πρόσβαση στις πληροφορίες επιτρέπεται μόνο σε συγκεκριμένους χρήστες. Πρακτικά αυτό γίνεται με διάφορους τρόπους, όπως η χρήση κάποιου αλγορίθμου που κάνει το κείμενο να μην βγάζει νόημα.
- *Ακεραιότητας Δεδομένων (Data Integrity)*, όπου πρέπει να εξασφαλίζεται ότι οι πληροφορίες όταν φτάνουν στο προορισμό τους δεν έχουν παραποιηθεί, συνήθως από κάποιον ενδιάμεσο παράγοντα (man in the middle).
- *Πιστοποίησης (Authentication),* όπου όταν φτάνει μια πληροφορία από έναν χρήστη σε κάποιον άλλο, ο δεύτερος μπορεί να εξασφαλίσει ότι αυτή η πληροφορία προήλθε από τον πρώτο και όχι κάποιον τρίτο, καθώς επίσης και ότι ήρθε την σωστή ημερομηνία ή ώρα.
- *Υπευθυνότητας (Accountability),* όπου πρέπει να εξασφαλίζει ότι αυτός που στέλνει την πληροφορία, δεν μπορεί αργότερα να αρνηθεί ότι την έστειλε.

Υπάρχουν δύο βασικές μέθοδοι κρυπτογράφησης :

- *Stream ciphers*, τα οποία κρυπτογραφούν το μήνυμα στοιχείο-στοιχείο, συνήθως bytes.
- *Block ciphers*, τα οποία παίρνουν έναν αριθμό στοιχείων που κρυπτογραφούνται μαζί ως μια οντότητα. Επίσης προσθέτουν κάποια bits στο κείμενο ώστε να είναι πολλαπλάσιο του μεγέθους του block που χρησιμοποιείται.

6.3. Κρυπτανάλυση [11] [12]

Κρυπτανάλυση είναι η επιστήμη και η μελέτη μεθόδων για το σπάσιμο των ciphers. Το cipher δεν είναι ασφαλές αν μπορεί ένας κρυπταναλυτής να παράγει ή να καταλάβει το απλό κείμενο ή το κλειδί της κρυπτογράφησης από κάποιο κρυπτογραφημένο κείμενο σε συνδυασμό μερικές φορές και με το απλό κείμενο.

Οι κρυπταναλυτικές επιθέσεις χωρίζονται σε:

- Επίθεση μόνο στο κρυπτογραφημένο κείμενο, όπου προφανώς ο κρυπταναλυτής διαθέτει μόνο το κρυπτογραφημένο κείμενο και προσπαθεί να το αποκρυπτογραφήσει ή να βρει το κλειδί της κρυπτογράφησης αναλύοντας τη δομή του κειμένου και στατιστικά χαρακτηριστικά που μπορεί να εμφανίζονται.
- Επίθεση στο γνωστό απλό κείμενο, όπου ο κρυπταναλυτής διαθέτει κάποιες πληροφορίες για το απλό κείμενο και προσπαθεί από τον συνδυασμό αυτών και του κρυπτογραφημένου κειμένου να αποκρυπτογραφήσει και το υπόλοιπο κείμενο.
- Επίθεση στο επιλεγμένο γνωστό κείμενο, η οποία είναι η πιο ευνοϊκή κατάσταση, όπου ο κρυπταναλυτής έχει την επιλογή να διαλέξει το απλό κείμενο και να το συγκρίνει με το κρυπτογραφημένο.

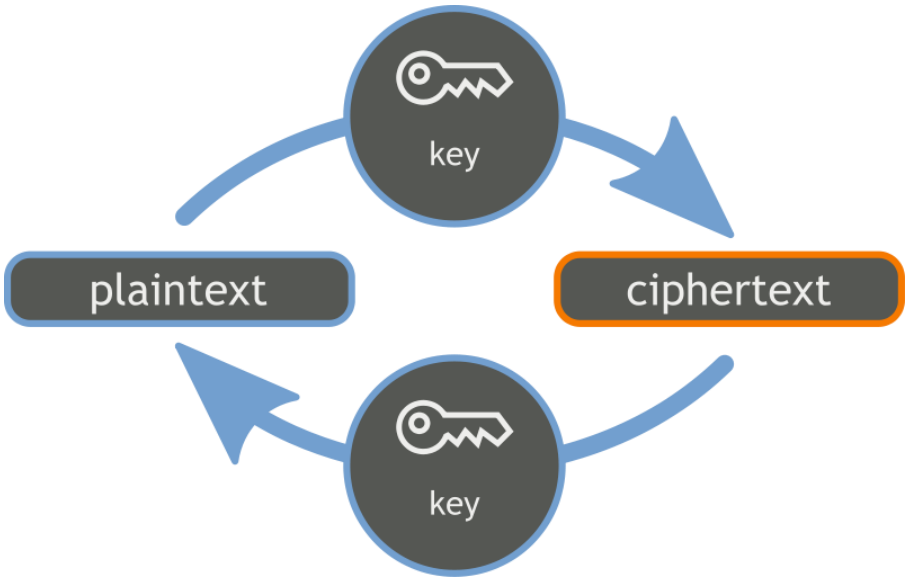
Οι εφαρμογές της κρυπτογραφίας μπορούν να χωριστούν σε δύο κατηγορίες:

- *Αποθήκευση*, όπου τα δεδομένα είναι αποθηκευμένα σε κάποιο σημείο, όπως π.χ. σε έναν σκληρό δίσκο. Συνήθως σε μια τέτοια περίπτωση ένας κρυπταναλυτής είναι πιθανό να γνωρίζει τη μέθοδο κρυπτογράφησης, αλλά όχι το κλειδί, ενώ διαθέτει αρκετό χρόνο ώστε να προσπαθήσει να βρει το κρυπτογραφικό κλειδί.
- *Μετάδοση*, όπου τα δεδομένα μεταδίδονται από κάποιο μέσο, όπως για παράδειγμα τηλέφωνο ή SMS. Σε αυτήν την περίπτωση το μήνυμα είναι διαθέσιμο σε κάποιον κρυπταναλυτή για μικρό χρονικό διάστημα. Επίσης συχνά τέτοιου είδους πληροφορίες χάνουν την αξία τους με το χρόνο, επομένως μια κάπως πιο αδύναμη κρυπτογράφηση μπορεί να είναι αρκετή για την εξασφάλιση του μηνύματος.

6.4. Κρυπτογράφηση Συμμετρικού/Ιδιωτικού Κλειδιού

Η κρυπτογράφηση συμμετρικού κλειδιού χρησιμοποιεί ένα μοναδικό κλειδί για την κρυπτογράφηση και την αποκρυπτογράφηση του μηνύματος, δηλαδή ο παραλήπτης ενός μηνύματος πρέπει να έχει το ίδιο κλειδί με τον αποστολέα. Αυτό αποτελεί και το κύριο πρόβλημα της κρυπτογράφησης με συμμετρικό κλειδί, γιατί

πρέπει να μεταδοθεί το κλειδί με κάποιον ασφαλή τρόπο, κάτι που δεν είναι πάντα εύκολο, ειδικά στις μέρες μας που είναι πολύ πιθανό ο παραλήπτης ενός μηνύματος να μην γνωρίζει καν τον αποστολέα. Οι αλγόριθμοι συμμετρικού κλειδιού παρόλα αυτά είναι πολύ γρήγοροι στην κρυπτογράφηση και αποκρυπτογράφηση και δεν χρησιμοποιούν πολύ υπολογιστική δύναμη και για αυτόν τον λόγο χρησιμοποιούνται ευρέως.



6.1 Κρυπτογράφηση με συμμετρικό/ιδιωτικό κλειδί

6.4.1. DES (Data Encryption Standard)[20]

Ο αλγόριθμος DES είναι ένα παράδειγμα κρυπτογράφησης συμμετρικού (ιδιωτικού) κλειδιού (private key encryption) που χρησιμοποιεί block ciphers. Ο DES αναπτύχθηκε από την IBM κατά τη διάρκεια 1968-1975. Ήταν μέχρι πρόσφατα από τους πιο χρησιμοποιημένους αλγόριθμους κρυπτογράφησης αλλά πλέον κρίνεται ανασφαλής διότι χρησιμοποιεί μικρό κλειδί κρυπτογράφησης για τα σημερινά δεδομένα. Το κλειδί κρυπτογράφησης του DES είναι 64 bits από τα οποία όμως χρησιμοποιούνται μόνο τα 56 για την κρυπτογράφηση και έτσι ο αριθμός των κλειδιών (keyspace) είναι περίπου $7x2x10^6$. Τα υπόλοιπα 8 bits χρησιμοποιούνται για διάφορα parity checks (έλεγχοι ισοτιμίας).

6.4.2. AES (Advanced Encryption Standard) - Rijndael [13]

Ο αλγόριθμος AES είναι μια παραλλαγή του αλγόριθμου Rijndael ο οποίος δημιουργήθηκε από τους Joan Daemen και Vincent Rijmen. Η διαφορά τους είναι ότι το AES χρησιμοποιεί 128 bit μέγεθος block cipher, ενώ ο Rijndael μπορεί να χρησιμοποιήσει μεγέθη πολλαπλάσια του 32, (ελάχιστο 128, 192 και μέγιστο 256 bits). Ο AES είναι ουσιαστικά ο διάδοχος του DES και είναι επίσης αλγόριθμος κρυπτογράφησης συμμετρικού κλειδιού.

Το 1997 το National Institute of Standards and Technology (NIST) ανακοίνωσε το ξεκίνημα της ανάπτυξης ενός νέου αλγορίθμου κρυπτογράφησης που θα αντικαθιστούσε το DES και το triple-DES. Το NIST ανακοίνωσε επίσης πως το AES θα ήταν ανοιχτό ως προς το cipher που θα χρησιμοποιεί, με την έννοια ότι οποιοσδήποτε θα μπορεί να υποβάλει ένα cipher το οποίο θα έχει την δυνατότητα να επιλεγεί για το AES και ότι το ίδιο το NIST δεν θα έκανε την αξιολόγηση της ασφάλειας και της αποδοτικότητας των ciphers, αλλά θα το έδιναν στην κοινότητα της κρυπτολογίας και κρυπταναλυτών για να προσπαθήσουν να το “σπάσουν”.

Αυτό που έψαχνε το NIST ήταν ένα block cipher το οποίο να είναι εξίσου ασφαλές με το triple-DES, αλλά πολύ πιο αποδοτικό. Υπήρχαν διάφορα αιτήματα από το NIST για τα υποψήφια ciphers ώστε να μπορούν να ληφθούν υπ’ όψη, αυτά ήταν:

1. Έπρεπε να χρησιμοποιούν block ciphers τα οποία να μπορούν να υποστηρίζουν μέγεθος block, των 128 bits και μέγεθος κλειδιού των 128, 192 και 256 bits.
2. Έπρεπε να περιέχουν πλήρεις γραπτές προδιαγραφές του block cipher στη μορφή αλγορίθμου.
3. Έπρεπε να περιέχουν αναφορά υλοποίησης σε ANSI C και μαθηματικά βελτιστοποιημένη υλοποίηση σε ANSI C και Java.
4. Έπρεπε να εκτελέσουν διάφορα tests, μαζί με αναμενόμενα αποτελέσματα αυτών των test.

5. Έπρεπε να έχουν αναφορές για την αναμενόμενη υπολογιστική αποδοτικότητα του hardware και του software, την αναμενόμενη δύναμή τους ενάντια σε κρυπταναλυτικές επιθέσεις και τα πλεονεκτήματα και μειονεκτήματα του cipher σε διάφορες εφαρμογές.
6. Έπρεπε να περιέχει ανάλυση της δύναμης του cipher εναντίον κρυπταναλυτικών επιθέσεων

Από τα ciphers που πέρασαν από το NIST, επιλέχτηκε τελικά το cipher του Rijndael, λόγω της απλής και συμπαγούς δομής του, καθώς και επειδή ήταν κατάλληλο για υπολογιστικές πλατφόρμες των 8 και 32 bit, που είναι συνήθως διαθέσιμες.

6.4.2.1. Λειτουργία του AES

Το AES χρησιμοποιεί κυκλικούς μετασχηματισμούς και διάφορα βήματα για την μετατροπή του απλού κειμένου σε κρυπτογραφημένο, τα οποία λειτουργούν πάνω σε έναν 4x4 πίνακα που ονομάζεται state και περιέχει τα bytes του block του απλού κειμένου. Το κλειδί περιγράφεται επίσης από έναν αντίστοιχο πίνακα τεσσάρων σειρών και $N_k/32$ στηλών όπου N_k είναι το μέγεθος σε bits του κλειδιού. Το μέγεθος του κλειδιού ορίζει τον αριθμό των επαναλήψεων των κυκλικών μετασχηματισμών.

- 10 κύκλοι για 128-bit κλειδί.
- 12 κύκλοι για 192-bit κλειδί.
- 14 κύκλοι για 256-bit κλειδί.

Για τις ανάγκες της εργασίας θα μιλήσουμε για κλειδί μεγέθους 128 bit που είναι και το μέγεθος του κλειδιού στην εφαρμογή μας.

k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

p_0	p_4	p_8	p_{12}
p_1	p_5	p_9	p_{13}
p_2	p_6	p_{10}	p_{14}
p_3	p_7	p_{11}	p_{15}

6.2 Ο πίνακας state

Ο κυκλικός μετασχηματισμός, που αναφέρεται ως κύκλος, είναι μία σειρά από τέσσερεις μετασχηματισμούς που ονομάζονται βήματα. Αυτά είναι:

1. Το SubBytes βήμα, είναι ο μόνος μη-γραμμικός μετασχηματισμός του cipher. Σε αυτό το βήμα κάθε στοιχείο του πίνακα state αντικαθιστάται από ένα αντίστοιχό στοιχείο σε έναν 8-bit πίνακα, μέσω του S-box. Το S-box ορίζεται από τη συνάρτηση του GF(2^8) (Galois Field/ Finite Field):

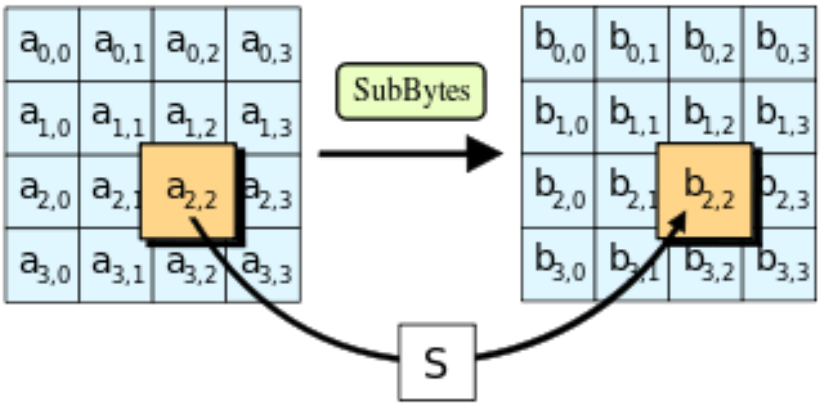
   $g: a \rightarrow b = a^{-1}$

   Αυτό έχει μη-γραμμικές ιδιότητες αλλά είναι ευάλωτο σε αλγεβρικούς χειρισμούς. Για αυτό συνδυάζεται με έναν affine μετασχηματισμό ο οποίος δεν επηρεάζει την μη-γραμμικότητα αλλά δημιουργεί μια πολύ πιο σύνθετη αλγεβρική παράσταση. Το S-box διαλέγεται με τέτοιον τρόπο ώστε να μην υπάρχουν σταθερά και αντίθετα σταθερά στοιχεία δηλαδή:

   $S[a] \oplus a \neq 00 \quad \forall a$

   $S[a] \oplus a \neq FF \quad \forall a$

   Για την αποκρυπτογράφηση χρησιμοποιείται το Inverse SubBytes το οποίο αντιστρέφει την διαδικασία.

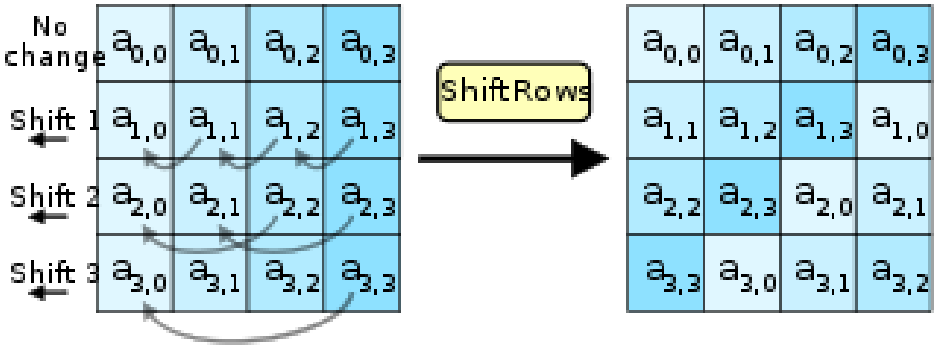


6.2 Το βήμα SubBytes

2. Το ShiftRows βήμα, είναι μια μεταφορά bytes η οποία μεταφέρει κυκλικά τις σειρές του state σε διαφορετικά offsets. Η μεταφορά γίνεται έτσι ώστε το byte στη θέση j και γραμμή i μεταφέρεται στη $(j- C_i)$ mod N_b, όπου τα offsets C_0, C_1, C_2 και C_3 εξαρτώνται από το N_b που στη περίπτωση του AES είναι 4. Τα τέσσερα offsets πρέπει να είναι διαφορετικά.

N_b	C_0	C_1	C_2	C_3
4	0	1	2	3

Τα offsets για 128-bit block (N_b=4)

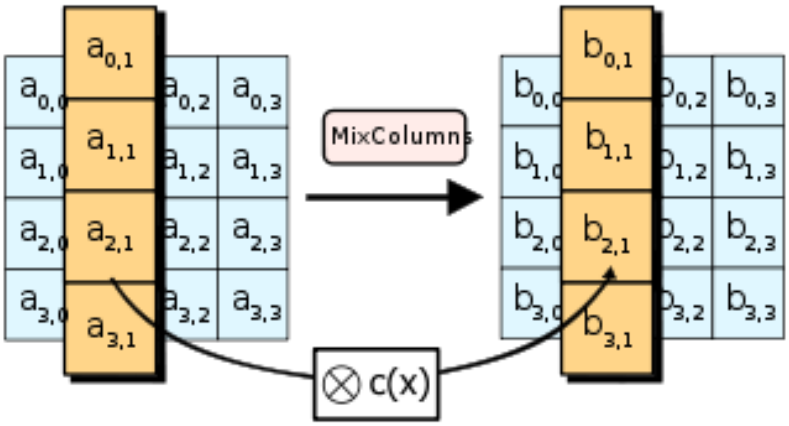


6.4 Το βήμα ShiftRows

3. Το βήμα MixColumns. Εδώ οι 4 στήλες του state συνδυάζονται με έναν αντιστρέψιμο γραμμικό μετασχηματισμό σε GF(2). Οι στήλες του state θεωρούνται πολυώνυμα σε GF(2^8) και πολλαπλασιάζονται modulo x^4+1 με ένα σταθερό πολυώνυμο c(x) = 03 x^3 + 01 x^2 + 01 x + 02. Αυτό μπορεί να γραφεί ως πίνακας:

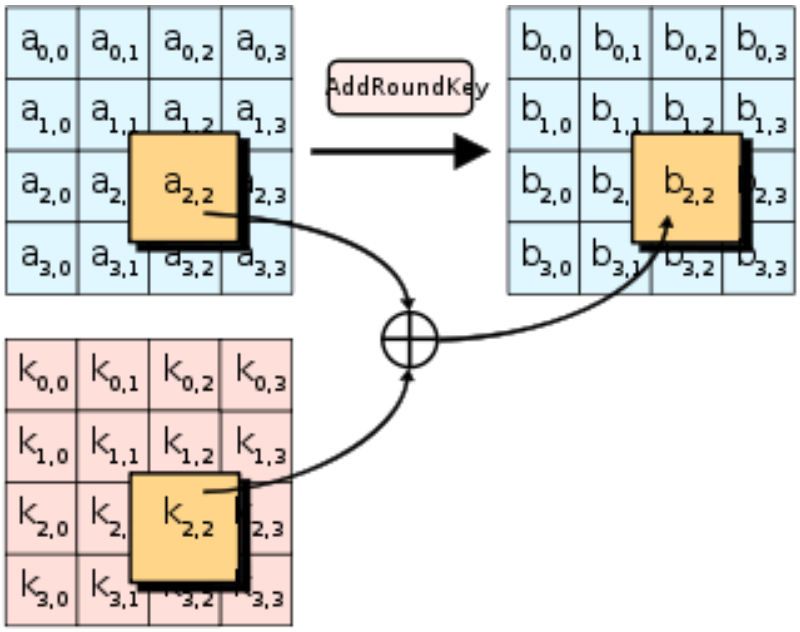
2	3	1	1
1	2	3	1
1	1	2	3
3	1	1	2

Με τον οποίο γίνεται ο μετασχηματισμός του MixColumns. Αυτή η διαδικασία μαζί με το ShiftRows προσθέτει ασάφεια.



6.5 Το βήμα MixColumns

4. Το βήμα AddRoundKey. Εδώ ο state τροποποιείται με τον συνδυασμό ενός κυκλικού κλειδιού με την bitwise XOR λειτουργία. Ο πίνακας με τα κλειδιά ExpandedKey βγαίνει από το cipher key μέσα από το key schedule του Rijndael. Το μέγεθος του κυκλικού κλειδιού είναι ίδιο με το μέγεθος του block.



6.6 Το βήμα AddRoundKey

6.5. Κρυπτογράφηση Ασύμμετρου/Δημοσίου Κλειδιού [11][15]

Η κρυπτογράφηση με δημόσιο κλειδί είναι ένα σύστημα το οποίο χρησιμοποιεί δύο κλειδιά, ένα δημόσιο με το οποίο γίνεται η κρυπτογράφηση και ένα ιδιωτικό με το οποίο γίνεται η αποκρυπτογράφηση και είναι γνωστό μόνο στον παραλήπτη του κρυπτογραφημένου μηνύματος. Οι αλγόριθμοι κρυπτογράφησης δημοσίου κλειδιού βασίζονται σε μαθηματικά προβλήματα που δεν έχουν απλή λύση, όπως το πρόβλημα του διακριτού λογαρίθμου ή το πρόβλημα παραγοντοποίησης ακέραιων αριθμών. Επειδή κατά τη μεταφορά του μηνύματος χρησιμοποιείται μόνο το δημόσιο κλειδί, δεν υπάρχει το πρόβλημα ασφαλούς μεταφοράς του κλειδιού που είναι το βασικό πρόβλημα των συμμετρικών συστημάτων κρυπτογράφησης. Παρόλα αυτά η κρυπτογράφηση δημοσίου κλειδιού είναι υπολογιστικά πολύπλοκη και αρκετά πιο αργή από την κρυπτογράφηση με ιδιωτικό κλειδί. Για αυτόν το λόγο συνήθως χρησιμοποιείται για μικρά μηνύματα και κυρίως για την μεταφορά του ιδιωτικού κλειδιού ενός άλλου συμμετρικού συστήματος.

6.5.1. Το Κρυπτοσύστημα RSA

Το κρυπτοσύστημα RSA είναι από τα πιο συνηθισμένα συστήματα κρυπτογράφησης δημοσίου κλειδιού και παίρνει το όνομά του από τους μελετητές που το δημοσίευσαν το 1978, Rivest, Shamir και Adleman. Το RSA βασίζεται στη συνάρτηση Euler φ(m) του modulo m. Για την κρυπτογράφηση το απλό κείμενο μετατρέπεται σε έναν θετικό αριθμό M στο δεκαδικό ή δυαδικό σύστημα. Θέλουμε ένα r modulo για την κρυπτογράφηση τέτοιο ώστε $1<M<r$ και ΜΚΔ(M,r)=1. Το τελικό κρυπτογραφημένο κείμενο βγαίνει όταν υψώσουμε το M σε έναν εκθέτη s και πάρουμε το modulo r, δηλαδή:

$$E \equiv M^s \bmod r \text{ με } 1<E<r$$

Το r επιλέγεται ως ένας πολύ μεγάλος όχι πρώτος αριθμός (διαφορετικά η

αποκρυπτογράφηση γίνεται αρκετά ευκολότερη), ο οποίος βγαίνει από r=pq όπου p και q είναι πρώτοι αριθμοί με περίπου 100 ψηφία. Ο εκθέτης κωδικοποίησης s πρέπει να είναι τέτοιος ώστε ΜΚΔ(s,φ(r))=1. Το ζεύγος (s,r) είναι το δημόσιο κλειδί. Οι παράγοντες του r, τα p και q είναι γνωστά μόνο στον παραλήπτη.

Για την αποκρυπτογράφηση πρέπει να υπολογίσουμε τον εκθέτη αποκρυπτογράφησης από την σχέση $ts \equiv 1 \bmod \phi(r)$ (αν το r ήταν πρώτος το θα ήταν πιο απλό mod(r-1)). Το ζεύγος (t,r) είναι το ιδιωτικό κλειδί. Ισχύει $r=pq \Rightarrow \phi(r) = r\,(1 - 1/p)\,(1 - 1/q) = (p - 1)\,(q - 1)$ και επομένως $t \equiv s^{\phi(\phi(r))-1} \bmod \phi(r)$

Ο t είναι ο μοναδικός εκθέτης αποκρυπτογράφησης. Η αποκρυπτογράφηση τελικά γίνεται αν υψώσουμε το κρυπτογραφημένο μήνυμα E στον εκθέτη t. $E^t \equiv M^{st} \bmod r = M^{\phi(r)k+1} \bmod r = M^{\phi(r)k}\, M \cdot \bmod r \equiv 1 \cdot M \bmod r$ Τελικά $E^t \equiv M \bmod r$ το οποίο μας δίνει το απλό κείμενο M.

6.6. Η Κλάση Κρυπτογράφησης της Εφαρμογής [18]

Η εφαρμογή μας όπως έχουμε προαναφέρει χρησιμοποιεί το σύστημα AES για την κρυπτογράφηση των μηνυμάτων. Το AES σαν μέθοδος κρυπτογράφησης υπάρχει μέσα στις βιβλιοθήκες javax.crypto της Java για το Android και τις καλούμε με τα:

```
import javax.crypto.Cipher;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
```

Οι λειτουργίες που χρησιμοποιεί η κλάση είναι:

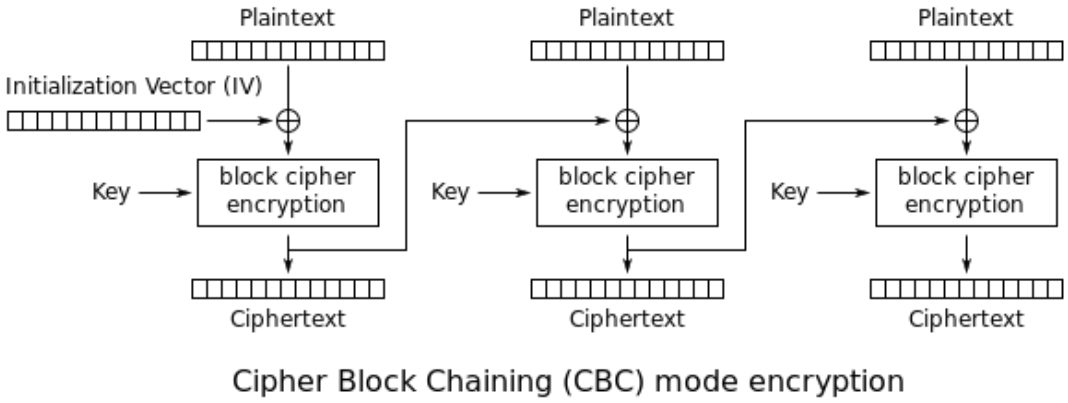
```
public static final String AES_MODE = "AES/CBC/PKCS5Padding";
public static final String CHARSET = "UTF-8";
private static final int PBE_ITERATION_COUNT = 4000;
private static final int AES_KEY_LENGTH_BITS = 128;
```

Το `AES/CBC/PKCS5Padding` σημαίνει ότι χρησιμοποιεί τον αλγόριθμο AES με τις ιδιότητες Cipher Block Chaining (CBC) και PKCS5Padding ο οποίος είναι ένας

μηχανισμός που προσθέτει bytes στο κείμενο για να το φέρει στο κατάλληλο μήκος ώστε να μπορεί να χωριστεί σε blocks. Μεγαλύτερο ενδιαφέρον παρουσιάζει η λειτουργία Cipher Block Chaining.

6.6.1. Cipher Block Chaining [16]

Το CBC δημιουργήθηκε από τους Erhsam, Meyer, Smith και Tuchman το 1976. Με αυτή τη λειτουργία το απλό κείμενο κρυπτογραφείται σε blocks όπως σε ένα απλό block cipher, με τη διαφορά ότι κάθε block απλού κειμένου για να κρυπτογραφηθεί εξαρτάται από το κρυπτογραφημένο κείμενο του προηγούμενου block με ένα bitwise XOR, δημιουργώντας έτσι μια αλυσίδα όπου η κρυπτογράφηση κάθε block εξαρτάται από όλα τα blocks που έχουν προηγηθεί.



6.7 Cipher Block Chaining Κρυπτογράφηση

Η κρυπτογράφηση του CBC αναπαρίσταται από :

$$C_i = E_k (P_i \oplus C_{i-1}) \text{ και } C_0 = IV$$

Η αποκρυπτογράφηση αντίστοιχα αναπαρίσταται από:

$$P_i = D_k(C_i) \oplus C_{i-1} \text{ και } C_0 = IV$$

Όπου:

- C_i είναι το κρυπτογραφημένο κείμενο του block i.
- E_k είναι το κλειδί της κρυπτογράφησης
- P_i είναι το απλό κείμενο του block i

- D_k είναι το κλειδί της αποκρυπτογράφησης
- IV είναι το Initialization Vector που χρησιμοποιείται στο πρώτο block ώστε να δημιουργεί τυχαία (όχι μοναδικά) μηνύματα σε κάθε κρυπτογράφηση.

6.6.2.1. Initialization Vector [17]

Το initialization vector (IV) είναι μια λειτουργία που χρησιμοποιεί ο αλγόριθμος κρυπτογράφησης, ώστε τα κρυπτογραφημένα μηνύματα να μην έχουν την ίδια μορφή και να είναι μοναδικά, ανεξάρτητα από το απλό κείμενο και το κλειδί που χρησιμοποιείται, δηλαδή το ίδιο κείμενο με το ίδιο κλειδί παράγουν διαφορετικό κρυπτογραφημένο κείμενο κάθε φορά μέσα από το IV. Το IV πρέπει να έχει το ίδιο μέγεθος με το μέγεθος block του αλγορίθμου, πρέπει να παράγεται από τυχαίους αριθμούς και δεν πρέπει να χρησιμοποιείται το ίδιο IV για την κρυπτογράφηση διαφορετικών μηνυμάτων.

Στην εφαρμογή μας το IV παράγεται από το:

```
public static byte[] generateIv() throws GeneralSecurityException {
    SecureRandom randomiv = new SecureRandom();
    byte[] ivBytes = new byte[16];
    randomiv.nextBytes(ivBytes);
    return ivBytes;
}
```

Η μέθοδος secureRandom() είναι αυτή που παράγει τυχαίους αριθμούς για το IV.

6.6.3. Salt

Το Salt είναι τυχαία bytes που εισάγονται στο κείμενο, όπου σε συνδυασμό με έναν αλγόριθμο κατακερματισμού κάνουν πιο δυνατή τη κρυπτογράφηση, κυρίως εναντίον επιθέσεων λεξικού. Το salt πρέπει, όπως και το IV, να είναι τυχαίο κάθε φορά.

Στην εφαρμογή μας το salt παράγεται από το:

```
public static byte[] generateSalt() throws GeneralSecurityException {
    SecureRandom randomsalt = new SecureRandom();
    byte[] saltBytes = new byte[20];
    randomsalt.nextBytes(saltBytes);
```

```
    return saltBytes;
}
```

6.6.4. Το Κρυπτογραφικό Κλειδί

Το κλειδί της κρυπτογράφησης στην εφαρμογή μας παράγεται από το:

```
public static byte[] deriveKey (String password, byte[] salt, int
iterations, int length) throws NoSuchAlgorithmException, Exception {
    PBEKeySpec keyspec = new PBEKeySpec(password.toCharArray(), salt,
iterations, length);
    SecretKeyFactory seckeyfac =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1");
    return seckeyfac.generateSecret(keyspec).getEncoded();
}
```

Αυτή η μέθοδος παίρνει σαν input τον κωδικό που χρησιμοποιεί ο χρήστης για την κρυπτογράφηση, το salt, τον αριθμό των επαναλήψεων και το μέγεθος του κλειδιού και παράγει το κλειδί μέσα από τη μέθοδο SecretKeyFactory από την αντίστοιχη βιβλιοθήκη της Java. Τα iterations που δέχεται σαν input η μέθοδος είναι ο αριθμός των κατακερματισμών του κωδικού που γίνονται κατά τη παραγωγή του κλειδιού, όπου όσο μεγαλύτερος είναι αυτός ο αριθμός τόσο πιο ασφαλές είναι το κλειδί. Το `PBKDF2WithHmacSHA1` είναι η λειτουργία που έχουμε εισάγει να χρησιμοποιεί το SecretKeyFactory με το οποίο παράγει το κλειδί της κρυπτογράφησης μέσα από τον κωδικό που εισάγει ο χρήστης και με τον αλγόριθμο κατακερματισμού SHA1 (Secure Hash Algorithm 1).

6.6.5. Η Διαδικασία της Κρυπτογράφησης

Η κρυπτογράφηση της εφαρμογής γίνεται από τη μέθοδο:

```
public static String encrypt (String message, String password) throws
Exception, UnsupportedEncodingException {

    // Calls the Initialization Vector Generation method
    byte[] iv = generateIv();
    // Calls the Salt Generation method
    byte[] salt = generateSalt();

    // Generates 128 bit encryption key (calls the deriveKey method)
    byte[] derkey = deriveKey(password, salt, PBE_ITERATION_COUNT,
AES_KEY_LENGTH_BITS);

    // Performs the Encryption
    SecretKeySpec enckeyspec = new SecretKeySpec(derkey, "AES");
    Cipher cipher = Cipher.getInstance(AES_MODE);
```

```
    cipher.init(Cipher.ENCRYPT_MODE, enckeyspec, new
IvParameterSpec(iv));
    byte[] ciphertext = cipher.doFinal(message.getBytes(CHARSET));

    // Output message that is an Array "Salt + Ciphertext + IV"
    byte[] outputmsg = new byte[20 + ciphertext.length + 16];
    System.arraycopy(salt, 0, outputmsg, 0, 20);
    System.arraycopy(ciphertext, 0, outputmsg, 20,
ciphertext.length);
    System.arraycopy(iv, 0 , outputmsg, 20+ciphertext.length, 16);

    // Encodes the output message to a Base64 String
    String encodedmsg = Base64.encodeToString(outputmsg,
Base64.NO_WRAP);

    return encodedmsg;
}
```

Αυτή η μέθοδος δέχεται σαν input το μήνυμα προς κρυπτογράφηση και τον κωδικό που εισάγει ο χρήστης. Ακολουθούν:

- Καλεί τη μέθοδο για τη δημιουργία IV.
- Καλεί τη μέθοδο για τη δημιουργία salt.
- Καλεί τη μέθοδο για τη παραγωγή του 128-bit κλειδιού.
- Εκτελεί την κρυπτογράφηση με τον συνδυασμό του κλειδιού, του αλγόριθμου AES και των διαδικασιών που αναφέραμε παραπάνω.
- Βάζει το κρυπτογραφημένο μήνυμα σε έναν πίνακα με τη σειρά salt, κρυπτογραφημένο κείμενο και IV.
- Κωδικοποιεί τον πίνακα σε μορφή base64, το οποίο μετατρέπει τα δυαδικά δεδομένα σε χαρακτήρες, ώστε να μπορεί να μεταφερθεί χωρίς πρόβλημα.

6.6.6. Η Διαδικασία της Αποκρυπτογράφησης

Η αποκρυπτογράφηση της εφαρμογής γίνεται από τη μέθοδο:

```
public static String decrypt(String encoutmsg, String password)
throws Exception {
    encoutmsg.toCharArray();

    // Get's the Array of the output message by decoding Base64
String
    byte[] outputmsg = Base64.decode(encoutmsg, Base64.NO_WRAP);

    // Recovers salt ciphertext and initialization vector (checks
minimum length 20byte salt +16 IV)
    if (outputmsg.length>36) {
        byte[] salt = Arrays.copyOfRange(outputmsg, 0, 20);
        byte[] ciphertext = Arrays.copyOfRange(outputmsg, 20,
```

```
outputmsg.length - 16);
        byte[] iv = Arrays.copyOfRange(outputmsg, outputmsg.length -
16, outputmsg.length);

        // Get's the encryption key from the output salt
        byte[] derkey = deriveKey(password, salt, 4000, 128);

        // Performs the Decryption
        SecretKeySpec enckeyspec = new SecretKeySpec(derkey, "AES");
        Cipher cipher = Cipher.getInstance(AES_MODE);
        cipher.init(Cipher.DECRYPT_MODE, enckeyspec, new
IvParameterSpec(iv));
        byte[] decmsg = cipher.doFinal(ciphertext);

        // Returns the Decrypted String
        return new String (decmsg, CHARSET);
    }
```

Αυτή η μέθοδος είναι παρόμοια με τη μέθοδο κρυπτογράφησης με αντίστροφη σειρά.

- Αποκωδικοποιεί το base64.
- Παίρνει το κρυπτογραφημένο μήνυμα από τον πίνακα που είχε τοποθετηθεί.
- Εκτελεί την αποκρυπτογράφηση εκτελώντας τα αντίστροφα βήματα της κρυπτογράφησης.
- Επιστρέφει το απλό μήνυμα.

7. MOC APP

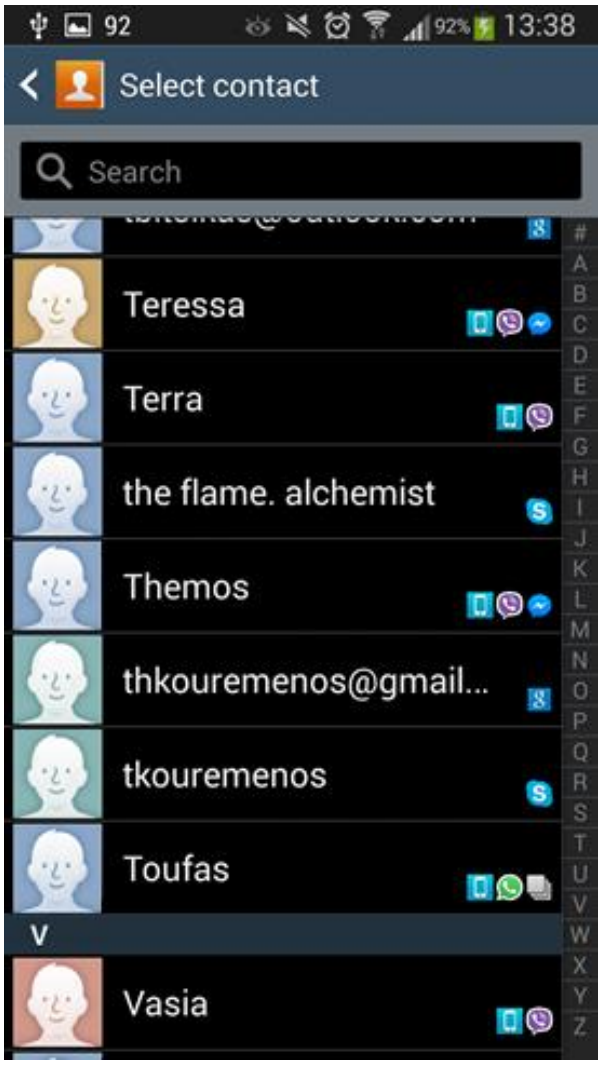
Θα κάνουμε μια επίδειξη της λειτουργίας της εφαρμογής σε ένα πραγματικό τηλέφωνο. Για την επίδειξη θα χρησιμοποιήσουμε τον χρήστη Themos από τις επαφές για να στείλουμε και να λάβουμε μηνύματα.

Αρχικά όταν ξεκινάμε την εφαρμογή, ανοίγει η οθόνη στα Conversations:

7.1 Conversations

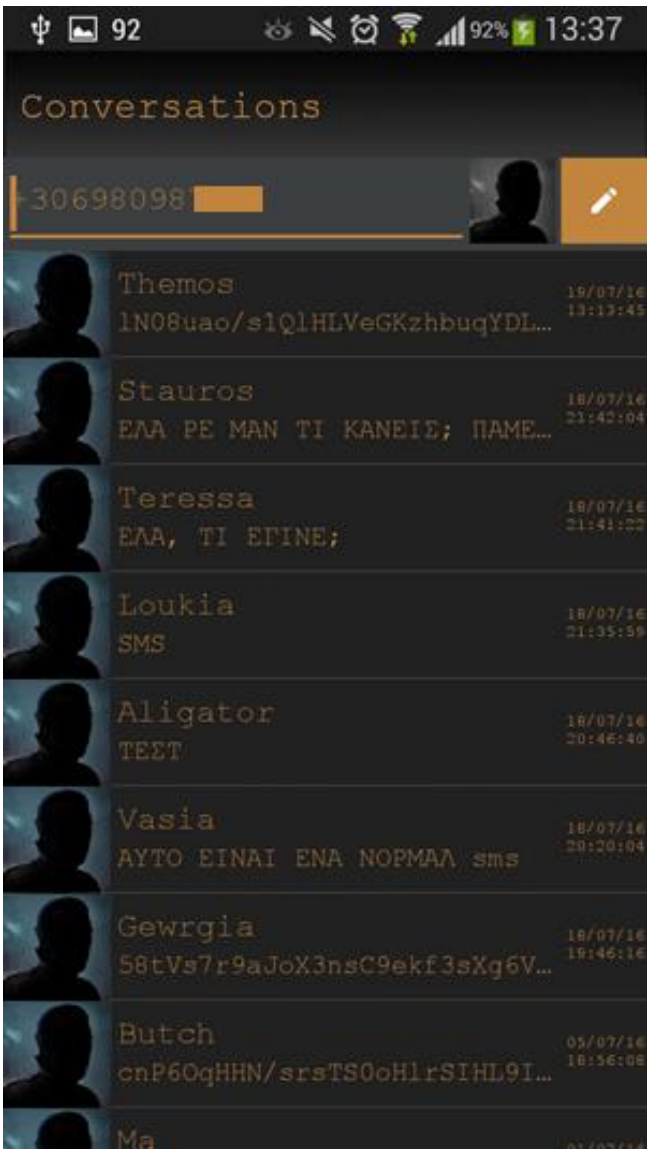
Σε αυτήν την οθόνη επιλέγουμε κάποια από τις συζητήσεις που θα μας ανοίξει την οθόνη του chat της συγκεκριμένης επαφής. Διαφορετικά πληκτρολογούμε τον αριθμό τηλεφώνου ή πατάμε στο κουμπί των επαφών, ώστε να επιλέξουμε την επαφή στην οποία θέλουμε να στείλουμε μήνυμα. Αν επιλέξουμε επαφή ή πληκτρολογήσουμε αριθμό πρέπει να πατήσουμε το κουμπί της σύνθεσης μηνύματος, το οποίο θα μας ανοίξει πάλι την οθόνη του chat της συγκεκριμένης επαφής. Θα πάμε μέσα από τις επαφές για να καλύψουμε όλες τις περιπτώσεις.

Πατώντας το κουμπί με το άγνωστο πρόσωπο στο πάνω δεξιά μέρος ανοίγουν οι επαφές.



7.2 Επαφές

Επιλέγουμε την επαφή Themos όπως προαναφέραμε. Όταν πατήσουμε πάνω στην επαφή επιστρέφουμε στην οθόνη με τα Conversations και ο αριθμός τηλεφώνου της επαφής έχει τοποθετηθεί στο πεδίο του αριθμού τηλεφώνου της εφαρμογής.



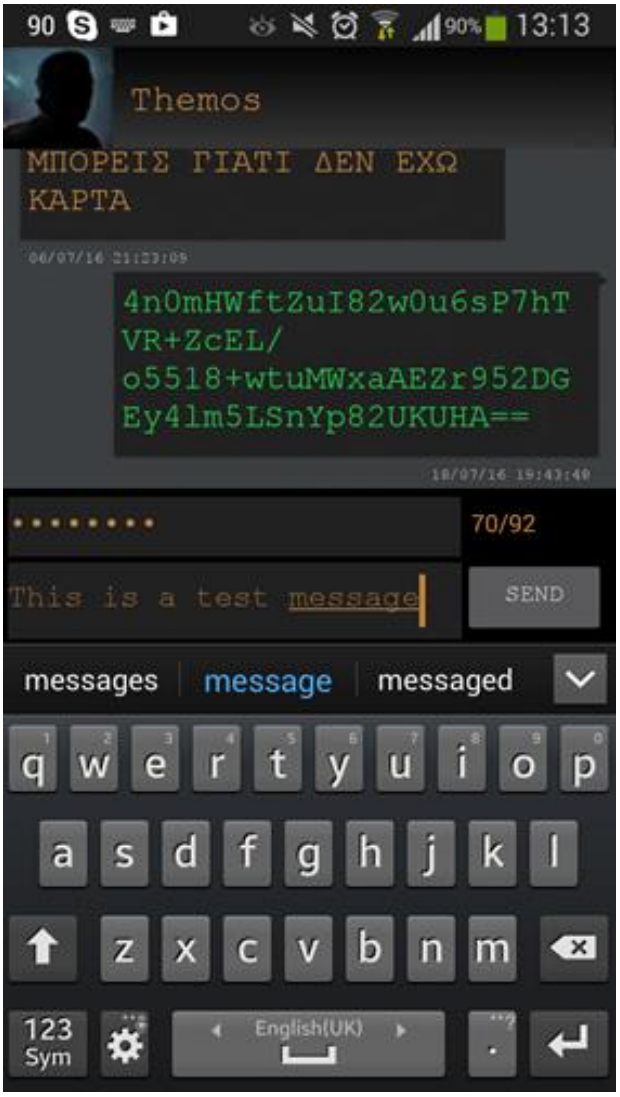
7.3 Το πεδίο του τηλεφωνικού αριθμού

Και τώρα πατάμε στη σύνθεση μηνύματος, το κουμπί με το μολύβι στο πάνω δεξιά μέρος της οθόνης.

Τώρα ανοίγει η οθόνη του chat με τον Themos.

7.4 Chat

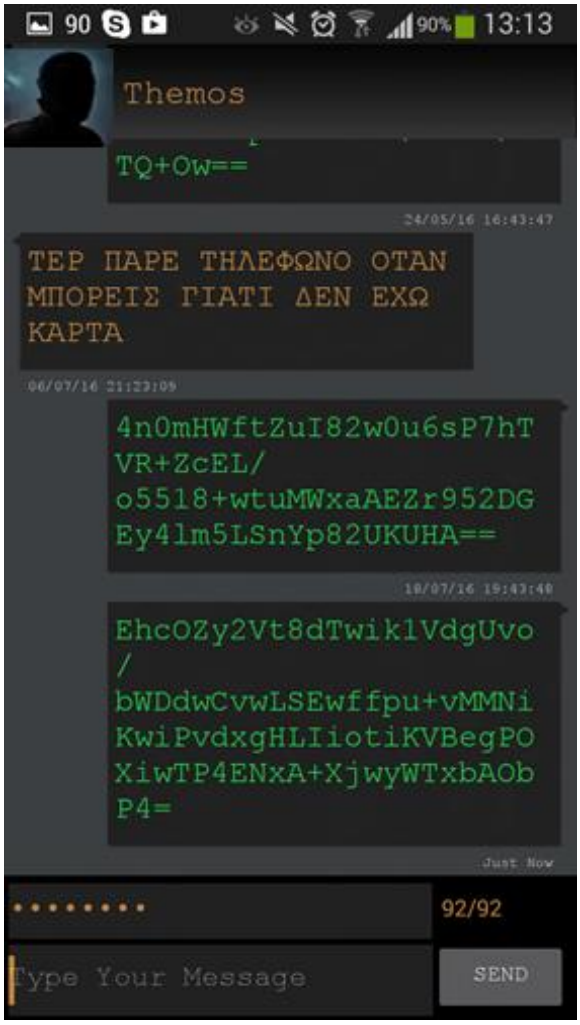
Τώρα θα συντάξουμε ένα κρυπτογραφημένο μήνυμα, με κωδικό password (κάτι που πρέπει να αποφεύγεται υπό κανονικές συνθήκες).



7.5 Τα πεδία του κωδικού και του μηνύματος

Ο κωδικός εμφανίζεται σε κουκίδες, όπως είχαμε προαναφέρει, ενώ ο μετρητής χαρακτήρων στα δεξιά, από 160 έγινε 92 διότι λαμβάνει υπόψη του τους χαρακτήρες που χρησιμοποιεί η κρυπτογράφηση. Πατάμε SEND και τότε στέλνει το μήνυμα που πληκτρολογήσαμε “This is a test message”.

Η οθόνη αφού πατήσουμε SEND είναι:

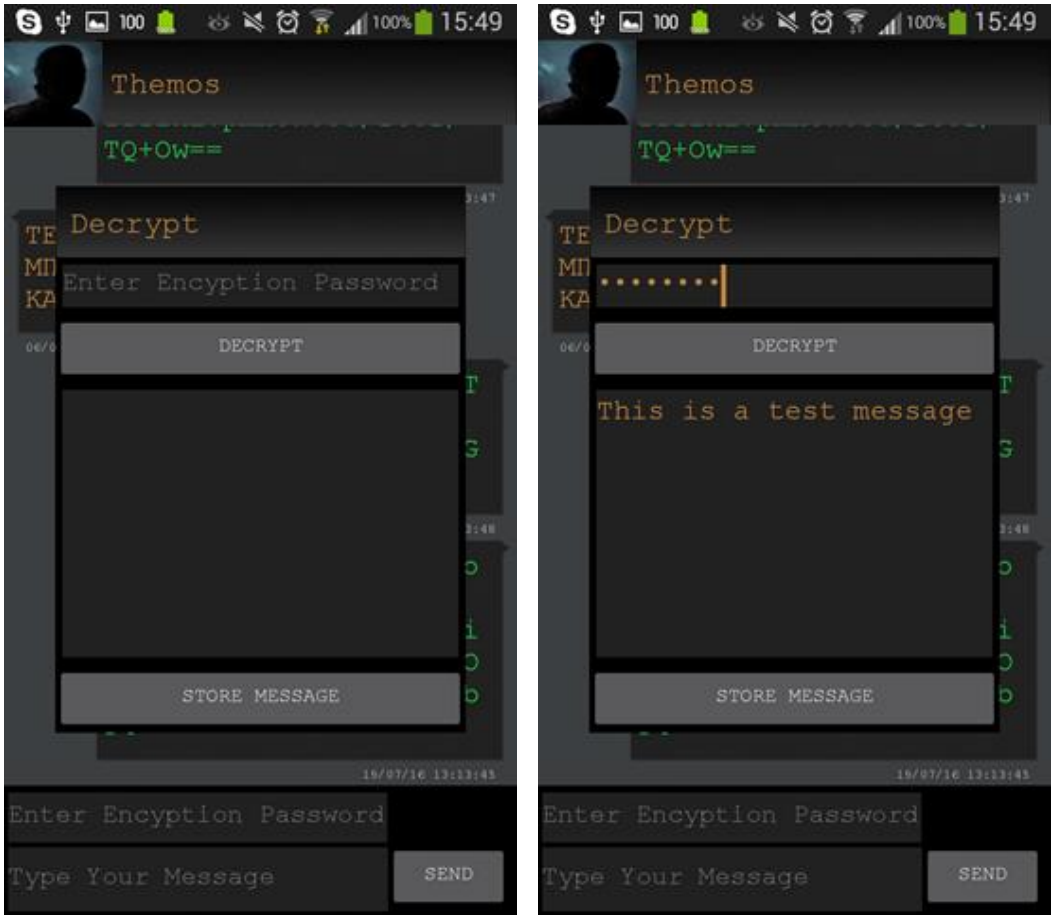


7.6 Το κρυπτογραφημένο μήνυμα

Στο κάτω μέρος της λίστας φαίνεται το καινούριο κρυπτογραφημένο μήνυμα που μόλις στείλαμε.

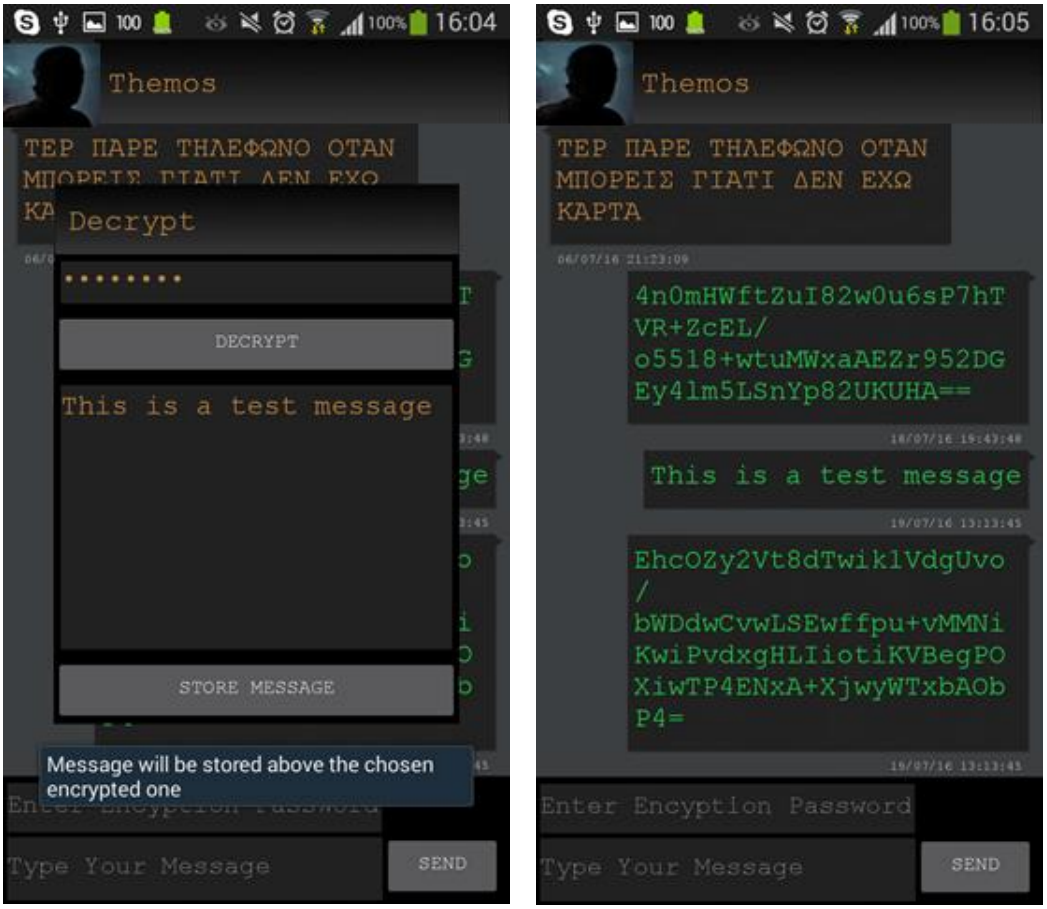
Τώρα θα αποκρυπτογραφήσουμε το μήνυμα που στείλαμε. Σημειώνουμε πως με τον τρόπο που θα δείξουμε, με τον ίδιο τρόπο αποκρυπτογραφούμε και μηνύματα που έχουμε λάβει.

Πατώντας μια φορά στο μήνυμα εμφανίζεται το παράθυρο DECRYPT. Εδώ πληκτρολογούμε τον κωδικό με τον οποίο κρυπτογραφήθηκε το μήνυμα (password στη προκειμένη περίπτωση), στο πεδίο που γράφει “Enter encryption password” και πατάμε το κουμπί DECRYPT. Τότε εμφανίζεται το μήνυμα στο κενό πεδίο από κάτω.



7.7 Το παράθυρο της αποκρυπτογράφησης

Το αποκρυπτογραφημένο μήνυμα δεν αποθηκεύεται στη συσκευή εκτός αν πατήσουμε το κουμπί STORE MESSAGE. Αν πατήσουμε το STORE MESSAGE βγαίνει ένα προειδοποιητικό μήνυμα και τοποθετείται το αποκρυπτογραφημένο μήνυμα ακριβώς πάνω από το μήνυμα που είχαμε επιλέξει και με την ίδια ημερομηνία ώστε να διατηρείται η φυσιολογική πορεία της συζήτησης.

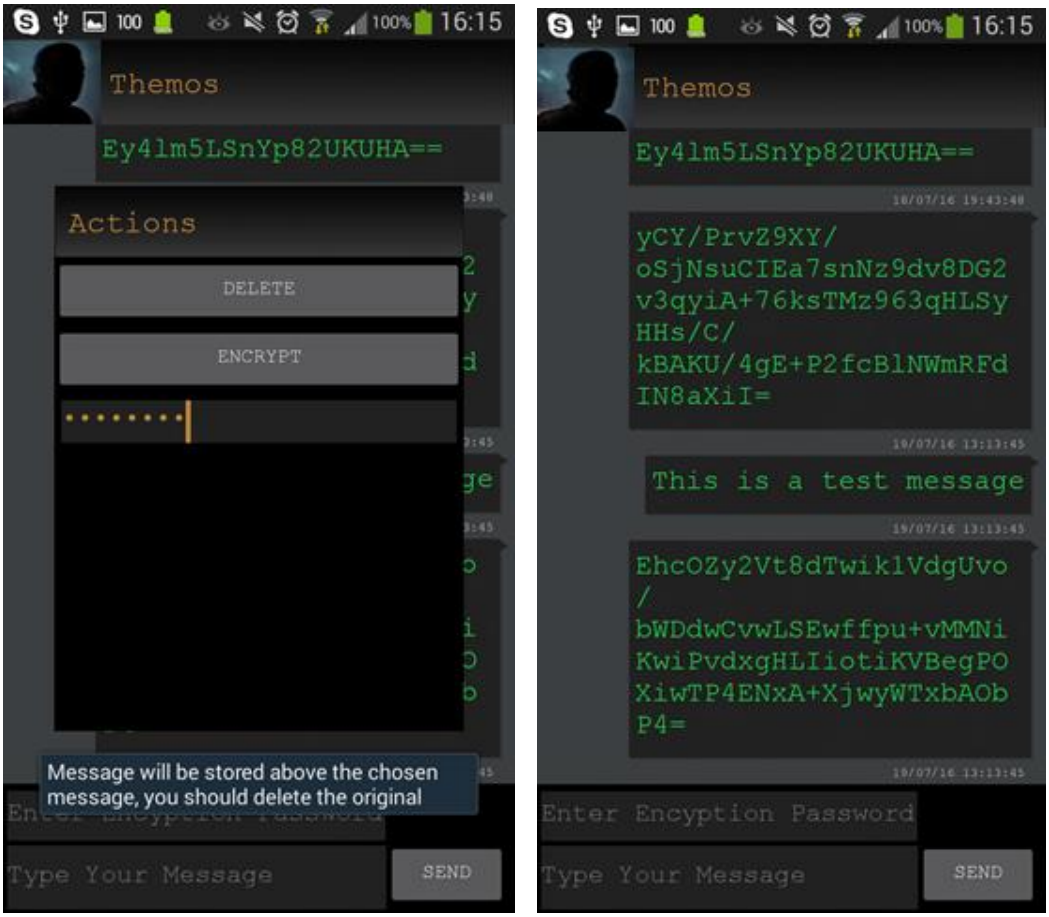


7.8 Η αποθήκευση του μηνύματος

Έστω τώρα ότι αυτό το μήνυμα δεν προήλθε από κάποιο άλλο κρυπτογραφημένο (δεν έχει καμία διαφορά από ένα άλλο απλό μήνυμα πλέον). Αν κρατήσουμε πατημένο λοιπόν αυτό το μήνυμα ανοίγει το παράθυρο Actions.

7.9 Οι πρόσθετες λειτουργίες

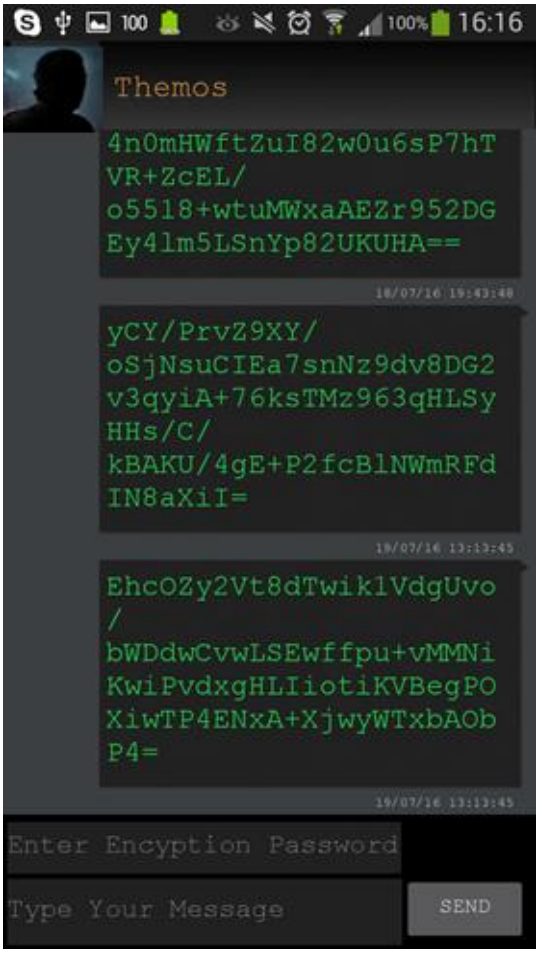
Εδώ μπορούμε είτε να διαγράψουμε το μήνυμα είτε να το κρυπτογραφήσουμε. Πληκτρολογούμε τον κωδικό κρυπτογράφησης για το μήνυμα που επιλέξαμε, εδώ χρησιμοποιούμε πάλι “password” και πατάμε το ENCRYPT. Τότε εμφανίζεται ένα μήνυμα ειδοποίησης και τοποθετεί το κρυπτογραφημένο μήνυμα ακριβώς πάνω από το μήνυμα που είχαμε επιλέξει και με την ίδια ημερομηνία ώστε να διατηρείται η φυσιολογική πορεία της συζήτησης.



7.10 Η διαδικασία της κρυπτογράφησης υπάρχοντος μηνύματος

Καταρχήν παρατηρούμε ότι το ίδιο μήνυμα όταν το κρυπτογραφήσαμε κατά την αποστολή και όταν το κρυπτογραφήσαμε από τη λίστα, με τον ίδιο κωδικό, παρήγαγαν διαφορετικό κρυπτοκείμενο, το οποίο είναι ιδιότητα της μεθόδου που χρησιμοποιήσαμε για την κρυπτογράφηση.

Τώρα μένει να διαγράψουμε το απλό μήνυμα ώστε να μην μπορεί να το διαβάσει κάποιος τρίτος, χρησιμοποιώντας την επιλογή DELETE στο παράθυρο Actions. Οπότε το chat παίρνει την τελική μορφή.



7.11 Το chat μετά από όλες τις διαδικασίες.

Τα δύο τελευταία μηνύματα είναι προφανώς το ίδιο μήνυμα, κάτω κάτω το αυθεντικό που στείλαμε και ακριβώς από πάνω το κρυπτογραφημένο που παρήγαμε κατά την κρυπτογράφηση από την λίστα.

8. Συμπεράσματα

8.1. Συνεργασία με άλλες εφαρμογές

Η εφαρμογή μας μπορεί να συνδυαστεί με διάφορες άλλες εφαρμογές, οι οποίες θα μπορούσαν είτε να χρησιμοποιήσουν ορισμένα τμήματα αυτής είτε να τη χρησιμοποιήσουν ολόκληρη.

Α. Σε εφαρμογές instant messaging

Πολλές instant messaging εφαρμογές για κινητά, όπως για παράδειγμα το Skype, έχουν τη δυνατότητα να στείλουν SMS, επομένως θα μπορούσε να χρησιμοποιηθεί αυτούσια η εφαρμογή μας για την ασφαλή μεταφορά SMS μέσα από αυτήν. Φυσικά πέρα από την μεταφορά SMS μπορεί να χρησιμοποιηθεί το μέρος της κρυπτογράφησης της εφαρμογής για αντίστοιχες instant messaging εφαρμογές, ώστε να παρέχουν end to end κρυπτογράφηση για την αποστολή των μηνυμάτων τους μέσω του διαδικτύου.

Β. Σε εφαρμογές με εικόνα και ήχο

Με κάποιες μετατροπές και προσθέσεις στον κώδικα ο αλγόριθμος της κρυπτογράφησης μπορεί να εφαρμοστεί σε εφαρμογές που χρησιμοποιούν εικόνα και ήχο για επικοινωνίες. Συγκεκριμένα θα μπορούσε να χρησιμοποιηθεί σε εφαρμογές video calls, ένα μέσο που χρησιμοποιείται όλο και περισσότερο ιδιαίτερα σε επαγγελματικό επίπεδο, και παρέχεται από πολλές εφαρμογές επικοινωνίας μέσω του διαδυκτίου, όπως για παράδειγμα το Viber και το Skype, είτε πρόκειται για κανονικές τηλεφωνικές επικοινωνίες, είτε πρόκειται για την μεταφορά αρχείων video, εικόνας και ήχου, όπως για παράδειγμα το MMS, κάτι που επίσης χρησιμοποιείται ευρύτατα, κυρίως σε προσωπικό επίπεδο.

Γ. Σε εφαρμογές διατήρησης προσωπικών δεδομένων

Φυσικά μπορεί να εφαρμοστεί ο αλγόριθμος κρυπτογράφησης σε εφαρμογές διατήρησης προσωπικών δεδομένων. Αυτές οι εφαρμογές διατηρούν ολόκληρα

αρχεία κρυπτογραφημένα ώστε να παραμένουν προστατευμένα από κρυπταναλυτικές επιθέσεις. Μια τέτοια εφαρμογή για παράδειγμα είναι το Crypt4all το οποίο επίσης χρησιμοποιεί το σύστημα AES για την κρυπτογράφηση των αρχείων. Ο αλγόριθμος όπως τον έχουμε εφαρμόσει στην εφαρμογή μας, είναι εύκολο να μεταφερθεί σε μια τέτοια εφαρμογή και παρέχει αρκετά δυνατή προστασία από τις περισσότερες συνηθισμένες κρυπταναλυτικές επιθέσεις.

8.2. Εφαρμογές και μελλοντικές αναβαθμίσεις

Σε αυτήν την εργασία καταλήξαμε ότι υπάρχουν προβλήματα ασφάλειας στο σύστημα που χρησιμοποιεί το SMS, είτε από την εκάστοτε εφαρμογή είτε από το δίκτυο που μεταδίδει τα μηνύματα. Οι περισσότερες εφαρμογές δεν χρησιμοποιούν κρυπτογράφηση για τα μηνύματα και το GSM χρησιμοποιεί αδύναμη κρυπτογράφηση, τέτοια ώστε τα μηνύματα να είναι ευάλωτα σε αρκετές επιθέσεις που μπορεί να εφαρμόσει ένας κρυπταναλυτής.

Η εφαρμογή μας καταπολεμά αρκετά από τα αδύναμα σημεία του δικτύου GSM καθώς και τα βασικά προβλήματα μυστικότητας των προσωπικών δεδομένων που έχουν πολλές εφαρμογές οι οποίες δεν χρησιμοποιούν κρυπτογράφηση. Το Advanced Encryption Standard είναι ένας δυνατός αλγόριθμος κρυπτογράφησης ο οποίος εφόσον έχει εφαρμοστεί σωστά, δεν έχει κάποια γνωστή μέθοδο με την οποία μπορεί να σπάσει.

Έτσι η εφαρμογή μας μπορεί να αξιοποιηθεί εμπορικά από εταιρίες που χρησιμοποιούν SMS για να μεταφέρουν σημαντικά δεδομένα, πληροφορίες ή γενικότερα για επικοινωνία και που επιθυμούν να βεβαιώνουν την ασφάλειά τους, καθώς επίσης και από απλούς χρήστες που προτιμούν κάποια μυστικότητα στις επικοινωνίες τους. Επίσης μπορεί να χρησιμοποιηθεί από άλλους προγραμματιστές για την προστασία προσωπικών δεδομένων των εφαρμογών τους και για την κρυπτογράφηση οποιουδήποτε κειμένου ή σαν μια βάση για εφαρμογές SMS.

Βέβαια σκοπεύουμε να βελτιώσουμε στο μέλλον την εφαρμογή. Μια από τις βασικότερες εφαρμογές που μπορούμε να προσθέσουμε είναι ένας δεύτερος αλγόριθμος κρυπτογράφησης δημοσίου κλειδιού, το RSA. Με το RSA θα μπορεί

ένας χρήστης να στέλνει τον κωδικό κρυπτογράφησης της αρχικής κρυπτογράφησης χωρίς κίνδυνο να τον διαβάσει κάποιος κατά τη μετάδοση καταπολεμώντας έτσι το πρόβλημα μετάδοσης του κλειδιού που έχουν τα συστήματα ιδιωτικού κλειδιού όπως το AES. Άλλη μία ιδιότητα που θέλουμε να προσθέσουμε είναι το να μην αποθηκεύει η εφαρμογή τα κλειδιά κρυπτογράφησης και τα προσωρινά αποκρυπτογραφημένα μηνύματα στη μνήμη, έτσι ώστε να καταπολεμάει τις cold boot επιθέσεις.

Επίσης θα ήταν καλό να δημιουργηθεί μια συμβατή εφαρμογή για άλλα λειτουργικά συστήματα όπως iOS ή windows, έτσι ώστε να μπορούν να επικοινωνούν με ασφάλεια χρηστές από διαφορετικές πλατφόρμες.

Βιβλιογραφία

[1] https://www.google.com/patents/US6208870

[2] http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[3] Mohsen Toorani, Ali A. Behesti, Solutions to the GSM Security Weaknesses

[4] J. Alex Halderman , Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, Edward W. Felten, Lest We Remember: Cold Boot Attacks on Encryption Keys, 2009

[5] https://www.sans.org/reading-room/whitepapers/telephone/gsm-standard-an-overview-security-317/

[6] https://developer.android.com/guide/components/fundamentals.html

[7] Wei Wang and Michael W. Godfrey, Detecting API Usage Obstacles: A Study of iOS and Android Developer Questions 2013

[8] https://developer.android.com/guide/topics/manifest/uses-sdk-element.html

[9] https://developer.android.com/training/material/lists-cards.html

[10] https://developer.android.com/guide/topics/ui/declaring-layout.html

[11] Α.Παπαϊωάννου, Χ.Κουκουβίνος, Κρυπτογραφία, 2007

[12] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, Handbook of Applied Cryptography

[13] Joan Daemen, Vincent Rijmen, The Design of Rijndael, 2002

[14] Owen Harrison, John Waldron, AES Encryption Implementation and Analysis on Commodity Graphics Processing Units, 2008

[15] http://www.iet.unipi.it/g.dini/Teaching/sncs/lectures/handouts/05.public-key-encryption.pdf

[16]https://www.google.com/patents/EP0725511B1?cl=en&dq=cipher+block+chaining+encryption&hl=el&sa=X&ved=0ahUKEwj4o-KojpTOAhWsIsAKHeZFDR4Q6AEIHDAA

[17] S. Frankel, R. Glenn, NIST, S. Kelly, Airespace, The AES-CBC Cipher Algorithm and Its Use with IPsec, September 2003

[18] http://docs.oracle.com/javase/7/docs/technotes/guides/security/StandardNames.html

[19] Lars Lockefeer, Encrypted SMS, an analysis of the theoretical necessities and implementation possibilities, 2010

[20] Dorothy Elizabeth, Robling Denning, Cryptography and Data Security, 1982

[21] Qijun Gu, Peng Liu, Denal of Service Attacks, 2007