University of Piraeus
Department of Digital Systems

MSc in Digital Systems Security

Master's Thesis

Επίθεση Σε Κυλιόμενους Κωδικούς Με Τη Χρήση SDR Εργαλείων

Παρασχάκης Ιωάννης

d.parashakis@ssl-unipi.gr

MTE 1731

Επιβλέπων Καθηγητής: **Κ. Νταντογιάν Χριστόφορος, dadoyan@unipi.gr**

Πειραιάς 2019-20

Contents

Περίληψη 5
Κεφάλαιο 1: Theory of Components 6
1.1 RKE Systems (Remote Keyless Entry) 6
1.2 Rolling Codes 6
1.2.1 Rolling Codes Features 7
1.2.2 Application in RF remote control 7
1.2.3 Vulnerabilities 7
1.3 Key Fob Hacks 8
1.3.1 Jamming the Key Fob Signal 8
1.3.2 Pulling the Response Codes from Memory 9
1.3.3 Brute-Forcing a Key Code 9
1.3.4 Forward-Prediction Attack 10
1.3.5 Dictionary Attacks 10
1.3.6 Dumping the Transponder Memory 10
1.4 Signal Modulation 11
1.4.1 Amplitude-Shift keying (ASK) 11
1.4.2 Frequency-Shift Keying 12
1.5 Volkswagen Schemes 12
1.5.1 VW-1 Scheme 13
1.5.2 VW-2 and VW-3 Scheme 13
1.5.3 VW-4 Scheme 15
Κεφάλαιο 2: Components 17
2.1 Hardware 17
2.1.1 Laptop 17
2.1.2 HackRF One 17
2.1.3 Antenna ANT500 18
2.1.4 Car Key Fob 19
2.2 Software 19
2.2.1 Operating System 19
2.2.2 Gqrx SDR 20
2.2.3 Osmocom_fft 20
2.2.4 GNU Radio Companion 20

2.2.5 Inspectrum 21
Κεφάλαιο 3: Software Installation Guide 23
Κεφάλαιο 4: Attack Scenario 26
4.1 Locating the Signal 26
4.2 Capturing the Signal 28
4.3 Sampling the Signal 30
4.4 Analyzing the Signal 37
4.5 Creating Blocks with GNU Radio 38
4.6 Editing the bruteforce_b.py 39
4.7 Editing the tutorial_bruteforce_b.xml 42
4.8 GNU Radio Flowgraph 43
Κεφάλαιο 5: Conclusion – Countermeasures 48
Κεφάλαιο 6: References 49

Table of Features

Figure 1: Man In The Middle 7
Figure 2: RollJam Attack 9
Figure 3: ASK Modulation 11
Figure 4: FSK Modulation 12
Figure 5: Packet structure of a rolling code for VW-1. Gray background indicates that the part is obfuscated o rholds the LFSR state. The start pulse is not shown 13
Figure 6: Packet structure of a rolling code for VW-2–4. Gray background indicates that the part is encrypted. Note that the fixed start pattern is shorter for VW-2. 14
Figure 7: One round i of the AUT64 block cipher as used in VW-2 and VW-3. a0,...,a7 is the 8bytestateofthecipher, g(a0,...,a7,keyi) the round function. 14
Figure 8: One round function g of the AUT64 block cipher as used in VW-2 and VW-3. a0,...,a7 is the 8-byte state of the cipher, key i the round key. 15
Figure 9: Laptop 17
Figure 10: HackRF One 17
Figure 11: ANT500 18
Figure 12: Audi Key Fob 19
Figure 13: GQRX Configuration Menu 27
Figure 14: GQRX Capturing a Pulse 27
Figure 15: GQRX Key Fob's Frequency 28
Figure 16: Osmocom_fft Configuration 29
Figure 17: osmocom_fft Capturing the Signal 30
Figure 18:Inspectrum 32
Figure 19: Inspectrum Sampling the Signal 33
Figure 20: Sampling the Signal 2 33
Figure 21: Inspectrum - Add Amplitude Plot 34
Figure 22: Inspectrum - Extracting Symbols 35
Figure 23: Extracted Symbols 35
Figure 24: Raw to Binary - Python Code 36
Figure 25: Binary Representation of Raw Input 36
Figure 26: GNU Radio Flowgraph 44
Figure 27: Inspectrum Information About The Signal 45
Figure 28: Comparing the Actual Signal with the our Signal 46

Περίληψη

Στην παρούσα εργασία περιγράφεται και αναλύεται ο τρόπος κατά τον οποίο μπορούμε να παραβιάσουμε την ασφάλεια ενός οχήματος, εκμεταλλευόμενοι την απουσία του αμοιβαίου ελέγχου αυθεντικοποίησης μεταξύ του κλειδιού και του δέκτη (εντός του οχήματος), απλά με την χρήση ενός ηλεκτρονικού υπολογιστή (laptop), σε συνδυασμό με το HackRF One - ενός ειδικά σχεδιασμένου εργαλείου τόσο σε επίπεδο hardware όσο και σε επίπεδο software με τέτοιο τρόπο ώστε να μπορεί να καταγράφει αλλά και να εκπέμπει σήματα σε συχνότητα UHF (Ultra-High Frequency).

Έχουν γίνει αρκετές αναφορές, στο παρελθόν, σε Replay attacks (επανεκπομπή σήματος). Στη περίπτωση αυτή όμως, προσπαθήσαμε να μαντέψουμε και να εξαντλήσουμε την ακολουθία των bits που στέλνεται ως πληροφορία στον δέκτη με την χρήση της γλώσσας προγραμματισμού Python σε συνδυασμό με ένα Out of Tree module που κατασκευάσαμε μέσω των δυνατοτήτων που μας παρέχοντας μέσω του GNU Radio – software που χρησιμοποιείται για την επεξεργασία, λήψη και αποστολή ενός σήματος από ψηφιακό σε αναλογικό.

Καθώς μας είναι γνωστό ότι τα κλειδιά των αυτοκινήτων σε Ευρωπαϊκά εδάφη εκπέμπουν στη συχνότητα των ~433 MHz, και μη έχοντας φυσική πρόσβαση στο τηλεχειριστήριο του οχήματος, το μόνο που χρειάζεται να γνωρίζουμε είναι η μάρκα του οχήματος που αντιστοιχεί στο εκάστοτε σήμα που έχουμε ως στόχο. Στην προκειμένη περίπτωση το σενάριο μας εφαρμόζεται σε ένα Audi A3 του 2011. Γνωρίζοντας λοιπόν την μάρκα του αυτοκινήτου και κατά προσέγγιση την χρονολογία κατασκευής του μπορούμε να συλλέξουμε βασικές πληροφορίες για την κατασκευή του τηλεχειριστηρίου αλλά και του αλγόριθμου κρυπτογράφησης.

Συλλέξαμε λοιπόν όλα τα απαραίτητα δεδομένα για το όχημά μας, και έπειτα από την ανάλυση του σήματος, χρησιμοποιήσαμε όλες αυτές τις πληροφορίες για την επανακατασκευή από την αρχή. Ολοκληρώνοντας και την τελευταία διαδικασία απλά βρεθήκαμε σε απόσταση ~10 μέτρων από το όχημά μας και με μία απλή εκτέλεση του κώδικα από το GNU Radio ήταν αρκετή για να το παραβιάσουμε.

Κεφάλαιο 1: Theory of Components

1.1 RKE Systems (Remote Keyless Entry)

Τα συστήματα RKE βασίζονται στη μετάδοση δεδομένων από το τηλεχειριστήριο, το οποίο είναι ενσωματωμένο στο κλειδί του αυτοκινήτου. Με το πάτημα ενός κουμπιού, ένας ενεργός πομπός ραδιοσυχνότητας (RF) στο τηλεχειριστήριο παράγει συνήθως σήματα σε μια ελεύθερα χρησιμοποιήσιμη ζώνη συχνοτήτων. Αυτές περιλαμβάνουν τη ζώνη 315 MHz στη Βόρεια Αμερική και τη ζώνη 433 MHz ή 868 MHz στην Ευρώπη, με μια τυπική σειρά από δεκάδες έως εκατοντάδες μέτρα. Σημειώστε ότι μερικά παλιά αυτοκίνητα χρησιμοποιούν τεχνολογία υπέρυθρων αντί RF. Τα συστήματα RKE επιτρέπουν στον χρήστη να κλειδώνει και να ξεκλειδώνει άνετα το όχημα από απόσταση και μπορεί να χρησιμοποιηθεί για την ενεργοποίηση και την απενεργοποίηση του συναγερμού κατά της κλοπής, όταν αυτός υπάρχει.

Τα πρώτα τηλεχειριστήρια για τα οχήματα δεν χρησιμοποίησαν καθόλου κρυπτογράφηση, καθώς το όχημα ξεκλείδωνε μετά την πρώτη επιτυχή λήψη ενός σήματος σταθερού κώδικα (fixed codes). Οι επιθέσεις επανεκπομπής (replay attack) σε αυτά τα συστήματα είναι αρκετά συνήθης.

Η επόμενη γενιά των συστημάτων RKE είναι τα αποκαλούμενα συστήματα κυλιόμενου κώδικα (rolling codes), τα οποία χρησιμοποιούν κρυπτογράφηση και έναν μετρητή που αυξάνεται σε κάθε πάτημα των κουμπιών.

Όλοι οι κυλιόμενοι κώδικες με μια παλαιότερη τιμή απορρίπτονται από το σύστημα του δέκτη. Αυτός ο μηχανισμός αποτελεί αποτελεσματική προστασία από επιθέσεις επανεκπομπής (replay attack), δεδομένου ότι ένας κωδικός κύλισης ακυρώνεται μόλις ληφθεί από το όχημα.

1.2 Rolling Codes

Rolling codes ή διαφορετικά hopping codes είναι εκείνοι οι κώδικες που χρησιμοποιούνται στα keyless entry συστήματα με σκοπό την αποτροπή επιθέσεων επανεκπομπής (replay attacks), όπου ένας κακόβουλος ενδιάμεσος (eavesdropper) καταγράφει την μετάδοση του σήματος και το αποθηκεύει με τελικό σκοπό την μελλοντική επανεκπομπή του. Τέτοια συστήματα χρησιμοποιούνται συνήθως σε πόρτες γκαράζ, καθώς επίσης και σε πολλά οχήματα.

1.2.1 Rolling Codes Features

Οι rolling codes (hopping codes) έχουν τα παρακάτω χαρακτηριστικά:

1. Χρησιμοποιούν ένα κοινό PRNG (Pseudorandom number generator) – προτιμότερα κρυπτογραφημένος τόσο από την πλευρά του πομπού όσο και από την πλευρά του δέκτη.
2. Ο πομπός κάθε φορά έτσι στέλνει τον επόμενο κωδικό σειριακά.
3. Ο δέκτης στη συνέχεια ελέγχει εάν αυτό που έλαβε αντιστοιχεί με αυτό που υπολογιστικά έπρεπε να λάβει.
4. Μια τυπική εφαρμογή της μεθοδολογίας αυτής συγκρίνει το λαμβανόμενο σήμα εντός των επόμενων 256 κωδικών σε περίπτωση που ο δέκτης χάσει κάποια μεταδιδόμενα σήματα.

1.2.2 Application in RF remote control

Ένας rolling code πομπός λοιπόν, είναι χρήσιμος στα συστήματα ασφαλείας για την παροχή ασφαλών κρυπτογραφημένων ραδιοσυχνοτήτων (RF) μετάδοσης που περιλαμβάνει ένα παρεμβαλλόμενο κώδικα με bits – σταθερό και μεταβλητό. Ένας δέκτης αποδιαμορφώνει την κρυπτογραφημένη μετάδοση ραδιοσυχνοτήτων και ανακτά και τον σταθερό κωδικό αλλά και τον μεταβαλλόμενο. Με τη σύγκριση των σταθερών και κυλιόμενων κωδικών με τους αποθηκευμένους κωδικούς και τον προσδιορισμό ότι το σήμα προέρχεται από έναν εξουσιοδοτημένο πομπό, παράγεται ένα σήμα για να ενεργοποιήσει έναν ηλεκτρικό κινητήρα για να ανοίξει ή να κλείσει ένα κινητό στοιχείο.

1.2.3 Vulnerabilities

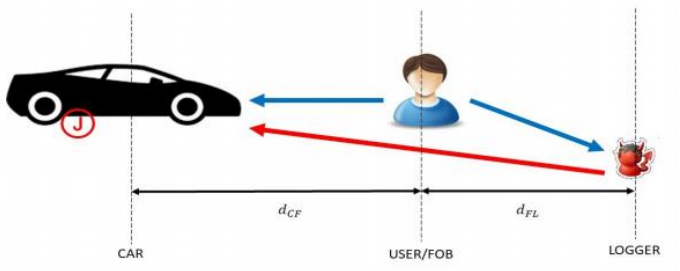

Figure 1: Man In The Middle

Ένας rolling code που μεταδίδεται με ραδιοσήματα μπορεί να τον καθιστά ευάλωτο σε πλαστογράφηση. Το 2015 αναφέρθηκε ότι ο Samy Kamkar είχε κατασκευάσει μια φτηνή ηλεκτρονική συσκευή σχετικά στο μέγεθος ενός πορτοφολιού που θα μπορούσε να αποκρύπτεται πάνω ή κοντά σε ένα κλειδωμένο όχημα για να καταγράψει έναν ενιαίο κωδικό εισόδου χωρίς την ανάγκη ύπαρξης κλειδιού που θα χρησιμοποιηθεί αργότερα για να ξεκλειδώσει το όχημα. Η συσκευή μεταδίδει ένα σήμα μπλοκαρίσματος (jamming signal) με σκοπό να εμποδίσει τη λήψη του πραγματικού σήματος από το δέκτη του κατόχου του οχήματος, ενώ ταυτόχρονα το καταγράφει. Έτσι όταν ο κάτοχος παρατηρεί ότι με την πρώτη προσπάθεια δεν καταφέρνει να ανοίξει το όχημά του, στέλνει στην συνέχεια τον δεύτερο κωδικό. Ο πρώτος καταγεγραμμένος κωδικός διαβιβάζεται στο όχημα μόνο όταν ο κάτοχος κάνει τη δεύτερη προσπάθεια, ενώ ο καταγραμμένος δεύτερος κωδικός διατηρείται/αποθηκεύεται για μελλοντική χρήση.

1.3 Key Fob Hacks

Υπάρχουν πολλοί τρόποι για να παραβιάσουμε τα συστήματα των τηλεχειριστηρίων που χρησιμοποιούνται στα οχήματα και παρακάτω παρουσιάζονται κάποια παραδείγματα μερικών μεθόδων όπου ένας εισβολέας μπορεί να εκμεταλλευτεί προς όφελός του.

1.3.1 Jamming the Key Fob Signal

Ένας τρόπος για να επιτεθεί ένας κακόβουλος σε ένα σήμα που εκπέμπεται από το τηλεχειριστήριο ενός κλειδιού είναι να το μπλοκάρει περνώντας δεδομένα παρεμβολής εντός της ζώνης διέλευσης του δέκτη RFID, δηλαδή στην περιοχή που ακούει και λαμβάνει ο δέκτης για ένα έγκυρο σήμα. Το πλάτος του passband παραθύρου περιλαμβάνει κάποιο πρόσθετο χώρο όπου υπάρχει η δυνατότητα να προστεθεί θόρυβος για την παρεμπόδιση του δέκτη ώστε να αλλάξει τον κυλιόμενο κωδικό, rolling code, ενώ παράλληλα επιτρέπει στον εισβολέα να δει τη σωστή ακολουθία των bits. Ενώ λοιπόν ο εισβολέας διατηρεί στην μνήμη τον έγκυρο κωδικό από το σήμα για το ξεκλείδωμα του αυτοκινήτου, περιμένει να αποσταλεί και το επόμενο στη σειρά σήμα και το καταγράφει και αυτό. Ο επιτιθέμενος μπορεί στη συνέχεια να επαναλάβει το πρώτο έγκυρο πακέτο που κατέγραψε στο όχημα, προκαλώντας το κλείδωμα ή το ξεκλείδωμα του αυτοκινήτου, ανάλογα με το σήμα που αποστέλλεται από το πλήκτρο του κλειδιού. Όταν ο ιδιοκτήτης του αυτοκινήτου εγκαταλείπει τελικά το όχημα, ο εισβολέας με το αποθηκευμένο έγκυρο κλειδί, δηλαδή αυτό που έλαβε τελευταίο, μπορεί να το επανεκπέμψει για να ανοίξει τις πόρτες του οχήματος ή να ξεκινήσει το όχημα.

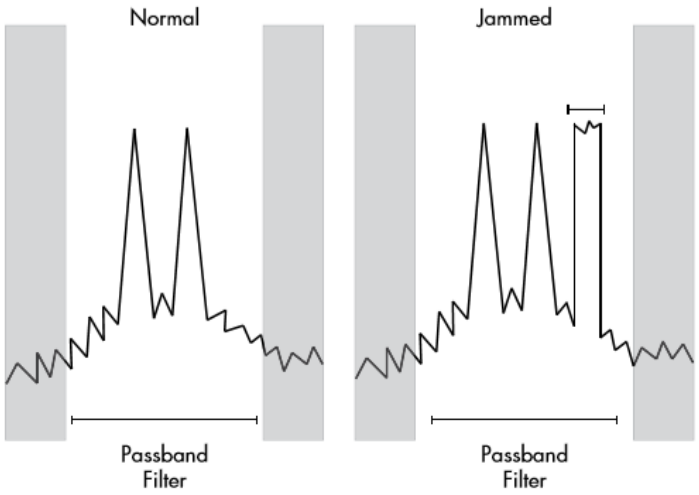

Figure 2: RollJam Attack

1.3.2 Pulling the Response Codes from Memory

Μερικές φορές είναι δυνατό να βρεθεί ο κώδικας απόκρισης ακόμα και στη μνήμη του immobilizer, ακόμη και λίγα λεπτά μετά τη διακοπή της αποστολής σημάτων. Με αυτό το τρόπο παρέχεται ένα παράθυρο ευκαιρίας για να ξεκινήσει το όχημα, όχι με τη λήψη σημάτων σε πραγματικό χρόνο από ένα πλήκτρο του κλειδιού που μεταδίδεται, αλλά με την απόσπαση του σήματος από τη μνήμη του immobilizer. Εάν εντοπιστεί μια περιοχή μνήμης που περιέχει αυτές τις πληροφορίες, ο επιτιθέμενος πρέπει είτε να αποκτήσει γρήγορα πρόσβαση στο όχημα είτε να έχει μια συσκευή στο όχημα που μπορεί να ανταποκριθεί για να καταγράψει αυτές τις πληροφορίες.

1.3.3 Brute-Forcing a Key Code

Ορισμένοι κωδικοί απόκρισης μπορούν να προσεγγιστούν με την μέθοδο του brute force, αν και η σκοπιμότητα μιας τέτοιας επίθεσης εξαρτάται από το μήκος του κλειδιού και τον αλγόριθμο κρυπτογράφησης. Για να επιτύχει μια επίθεση brute force, ο επιτιθέμενος πρέπει να κατασκευάσει ένα ειδικά διαμορφωμένο λογισμικό για να αντικαταστήσει το κλειδί χρησιμοποιώντας μια συσκευή SDR όπως το HackRF One που χρησιμοποιήσαμε στο δικά μας σενάριο.

1.3.4 Forward-Prediction Attack

Εάν ένας εισβολέας είναι σε θέση να παρατηρήσει ανταλλαγές challenge-response που πραγματοποιούνται όταν πατηθεί το κουμπί που στέλνει ένα σήμα στο όχημα και ο αναμεταδότης του οχήματος αποκρίνεται, τότε ο επιτιθέμενος μπορεί να εκτελέσει μια forward-prediction attack. Σε μια τέτοια επίθεση, ο επιτιθέμενος παρατηρεί πολλαπλές προκλήσεις και από αυτές μπορεί να προβλέψει ποιο θα είναι το επόμενο σήμα που θα μεταδοθεί. Εάν η ψευδοτυχαία γεννήτρια αριθμού (PRNG) του πομποδέκτη είναι αδύναμη, θα έχει ως συνέπεια την επιτυχή επίτευξη της επίθεσης αυτής. Για την απλούστευση του παραδείγματος, εάν το PRNG βασίστηκε στο πότε το τηλεχειριστήριο πήρε την πρώτη απάντηση, ένας εισβολέας θα μπορούσε να εμφυτεύσει τη δική του γεννήτρια τυχαίων αριθμών με έναν αντίστοιχο χρόνο έναρξης. Μόλις ο εισβολέας συγχρονιστεί με τον στόχο, θα μπορεί να προβλέψει όλους τους μελλοντικούς κωδικούς.

1.3.5 Dictionary Attacks

Ομοίως, εάν ένας εισβολέας μπορεί να καταγράψει πολλές έγκυρες ανταλλαγές challenge-response μεταξύ του κλειδιού και του αναμεταδότη, και μπορεί να τις αποθηκεύσει σε ένα λεξικό, τότε στη συνέχεια θα είναι σε θέση να χρησιμοποιήσει τα συλλεγμένα ζεύγη κλειδιών για να ζητήσει επανειλημμένα προκλήσεις από τον αναμεταδότη, μέχρις ότου μια πρόκληση να ταιριάξει με μια απάντηση στο λεξικό αυτό. Αυτή η δύσκολη επίθεση είναι δυνατή μόνο όταν το σύστημα εισαγωγής κλειδιών δεν χρησιμοποιεί την επαλήθευση του αποστολέα για να βεβαιωθεί ότι οι απαντήσεις είναι έγκυρες. Ο επιτιθέμενος θα πρέπει επίσης να είναι σε θέση να ζητά συνεχώς την πιστοποίηση από τον αναμεταδότη. Για να εκτελεστεί μια επίθεση λεξικού, ο επιτιθέμενος θα χρειαστεί να δημιουργήσει ένα σύστημα για να ενεργοποιήσει το αίτημα του κλειδιού και να καταγράψει την ανταλλαγή με μια συσκευή SDR. Ένα Arduino συνδεδεμένο με το πάτημα πλήκτρων του έγκυρου κλειδιού του ερευνητή θα ήταν αρκετό. Υποθέτοντας ότι ο έλεγχος ταυτότητας πραγματοποιείται μέσω του δίαυλου CAN, είναι επίσης δυνατή η ανάκτηση του αναγνωριστικού πλήκτρων FOB σε εξαιρετικά υψηλή συχνότητα και η προσπάθεια συλλογής του ρεύματος κλειδιού με την αναπαραγωγή και την καταγραφή της επικοινωνίας μέσω του διαύλου CAN. Χρησιμοποιώντας προσαρμοσμένα εργαλεία, αυτό θα ήταν δυνατό να επαναληφθεί σε οποιοδήποτε BUS δίκτυο.

1.3.6 Dumping the Transponder Memory

Είναι συχνά πιθανό να πραγματοποιηθεί η συγκεκριμένη επίθεση εφόσον υπάρχει φυσική πρόσβαση στο κλειδί, ώστε να αποσπαστούν στοιχεία από τη μνήμη του αναμεταδότη με

σκοπό την απόκτηση του μυστικού κλειδιού.

1.4 Signal Modulation

Για την καλύτερη κατανόηση του σεναρίου που παρουσιάζεται παρακάτω και αφού γνωρίζουμε μέχρι αυτό το σημείο ότι έχουμε να αντιμετωπίσουμε ένα μεταβαλλόμενο σήμα, θα πρέπει στην συνέχεια να εξετάσουμε και τους τύπους διαμόρφωσής του, καθώς θα χρειαστεί να μετατρέψουμε το καταγεγραμμένο σήμα από αναλογικό σε ψηφιακό.

Για να εφαρμόσουμε τη σωστή μέθοδο αποδιαμόρφωσης λοιπόν, θα πρέπει πρώτα να είμαστε σε θέση να προσδιορίσουμε τον τύπο διαμόρφωσης που χρησιμοποιεί ένα σήμα. Η διαμόρφωση σήματος είναι ο τρόπος με τον οποίο παρουσιάζονται τα δεδομένα δυαδικής μορφής που μεταδίδονται από ένα ασύρματο σήμα και βρίσκει εφαρμογή όταν υπάρχει η δυνατότητα να διακρίνουμε τη διαφορά μεταξύ του ψηφιακού σήματος «1» και του ψηφιακού σήματος «0». Υπάρχουν δύο συνήθεις τύποι διαμόρφωσης ψηφιακού σήματος: Amplitude-Shift keying (ASK) και Frequency-Shift Keying (FSK)..

1.4.1 Amplitude-Shift keying (ASK)

Όταν χρησιμοποιείται ο τύπος διαμόρφωσης ASK, τα δυαδικά ψηφία χαρακτηρίζονται από το πλάτος του σήματος. Το παρακάτω σχήμα δείχνει μια γραφική παράσταση του σήματος που μεταδίδεται σε κύματα. Ένα φέρον κύμα αποτελεί το εύρος του φορέα και όταν δεν υπάρχει παλμός, αυτή είναι η κατάσταση ηρεμίας του σήματος. Όταν η γραμμή μεταφοράς είναι υψηλή για συγκεκριμένη διάρκεια, αυτή καταγράφεται ως κύμα και χαρακτηρίζεται από το δυαδικό «1». Όταν η γραμμή του φορέα βρίσκεται σε κατάσταση ηρεμίας για μικρότερη διάρκεια, τότε αυτό χαρακτηρίζεται από το δυαδικό «0».

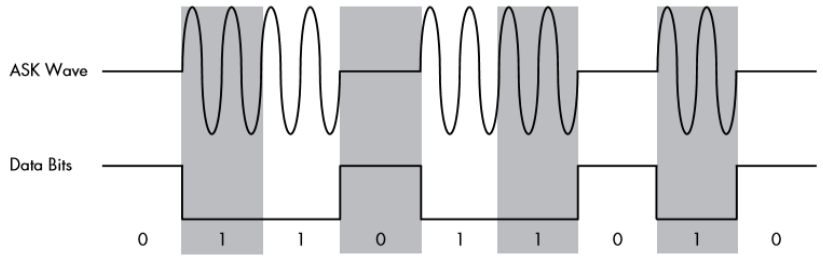

Figure 3: ASK Modulation

Η διαμόρφωση ASK είναι επίσης γνωστή ως On-Off Keying (OOK) και τυπικά χρησιμοποιεί ένα bit για την έναρξη και διακοπή του σήματος. Τα bits start-and-stop είναι συνηθισμένος τρόπος να ξεχωρίζεται από πού ένα μήνυμα ξεκινά και πού σταματά. Όπως παρουσιάζεται και στο παραπάνω σχήμα τα bits start-and-stop, μπορούν να αναπαρασταθούν από 9 bits: 0-1-1-0-1-1-0-1-0.

1.4.2 Frequency-Shift Keying

Αντίθετα με τη διαμόρφωση ASK, η FSK πάντοτε έχει έναν φορέα σήματος, αλλά σε αυτή τη περίπτωση το σήμα μετριέται με το πόσο γρήγορα αλλάζει την συχνότητα του.
Στην διαμόρφωση FSK, ένα σήμα υψηλής συχνότητας παρουσιάζεται από το δυαδικό «0», ενώ ένα σήμα χαμηλής συχνότητας από το δυαδικό «1». Όταν ένα φέρον κύμα είναι κοντό, τότε αυτό είναι «1», ενώ όταν απέχουν πολύ περισσότερο, τότε αυτό είναι το «0». Τα δυαδικά ψηφία στο παρακάτω σχήμα είναι πιθανόν να αποτελούν την ακολουθία: 1-0-0-1-0-0-1-0-1.

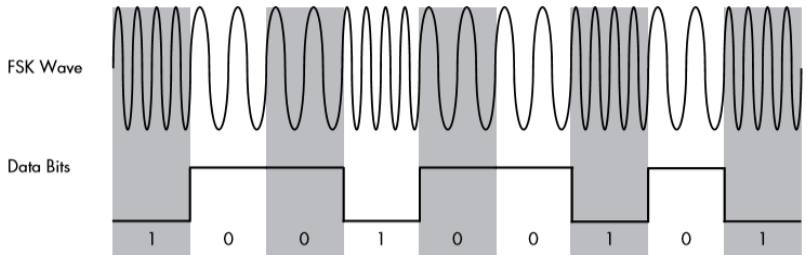

Figure 4: FSK Modulation

1.5 Volkswagen Schemes

Δεδομένου ότι στη παρούσα εργασία εξετάζουμε και αναλύουμε ένα rolling code σύστημα ενός οχήματος Audi, της οικογένειας Volkswagen, θα πρέπει να αναφέρουμε τις διάφορες κατηγορίες και την εξέλιξή τους μέσα στο βάθος των τελευταίων χρόνων. Παρακάτω αναφέρονται οι 4 βασικές και πιο διαδεδομένες κατηγορίες αυτών των συστημάτων μέσα από τις 7 συνολικά που έχουν εντοπιστεί μέχρι στιγμής.

1.5.1 VW-1 Scheme

Το συγκεκριμένο σύστημα είναι το μοναδικό μεταξύ των υπολοίπων όπου ο κωδικός μεταδίδεται στα 433.92 MHz, ενώ όλα τα υπόλοιπα μεταδίδουν στα 434.4 MHz όπως θα δούμε και παρακάτω. Σε αντίθεση με τα καινούργια RKE (Remote Keyless Entry) συστήματα, η αρχή του εκάστοτε πακέτου που μεταδίδεται δεν αναγνωρίζεται από ένα μακρύ κωδικό γνωστός ως preamble, αλλά από ένα μοναδικό μοτίβο «1» και «0» και το οποίο διαρκεί 500μs τόσο κατά το μέγιστο σημείο εκπομπής παλμού, όσο και κατά το ελάχιστο. Μετά από αυτή την ακολουθία bits, στέλνονται τα bits των πραγματικών δεδομένων με LSB-first (Least Significant Bit) κωδικοποιημένο κατά παλμό (pulse-width). Το «0» υποδηλώνεται από έναν κοντό παλμό μέγιστης εκπομπής ακολουθούμενο από ένα μακρύ παλμό ελάχιστης εκπομπής, ενώ το «1» ακριβώς το αντίθετο (long high, short low). Εδώ αξίζει να σημειωθεί ότι τα 4 πρώτα bytes πραγματικού κώδικα που μεταδίδονται περιέχουν και το UID του κλειδιού σε κατακερματισμένη μορφή, όπου αρκετά bytes του πακέτου έχουν αλλοιωθεί με XOR. Τα επόμενα 3 bytes "lfsr" περιέχουν το firmware και τέλος το τελευταίο byte υποδηλώνει την λειτουργία του κουμπιού που πατήθηκε (κλείδωμα, ξεκλείδωμα).

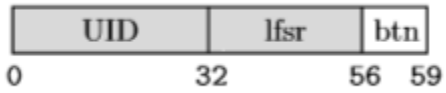

Figure 5: Packet structure of a rolling code for VW-1. Gray background indicates that the part is obfuscated o rholds the LFSR state. The start pulse is not shown

1.5.2 VW-2 and VW-3 Scheme

Η δομή του πακέτου του rolling code για το συγκεκριμένο σύστημα έχει την παρακάτω δομή:

1. Ένας κωδικός preamble από «0» και «1» σε μοτίβο
2. Ακολουθεί μία σταθερή ακολουθία start
3. Ένα κρυπτογραφημένο payload των 8 bytes
4. Τέλος ένα byte το οποίο υποδηλώνει την λειτουργία του κουμπιού που πατήθηκε.

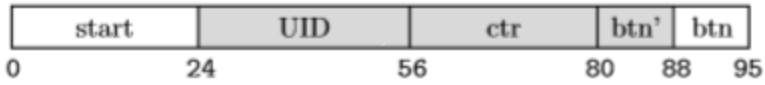

Figure 6: Packet structure of a rolling code for VW-2–4. Gray background indicates that the part is encrypted. Note that the fixed start pattern is shorter for VW-2.

Τα 8 byte payload που μεταδίδονται και φαίνονται και από το σχήμα παραπάνω με το σκιαγραφημένο μέρος δημιουργείται από τα ακόλουθα στοιχεία πρωτού αυτά κρυπτογραφηθούν:

1. 4-byte UID,
2. 3-byte μετρητή ctr,
3. 1-byte btn' που υποδηλώνει το κουμπί.

Το payload στη συνέχεια κρυπτογραφείται χρησιμοποιώντας ένα κατάλληλο αλγόριθμο κρυπτογράφησης (block cipher) όπου ανακτάται από το ECU firmware. Ο αλγόριθμος κρυπτογράφησης είναι ο AUT64.

Η λειτουργία του αλγόριθμου για ένα γύρο περιγράφεται στα δύο σχήματα που παρουσιάζονται παρακάτω:

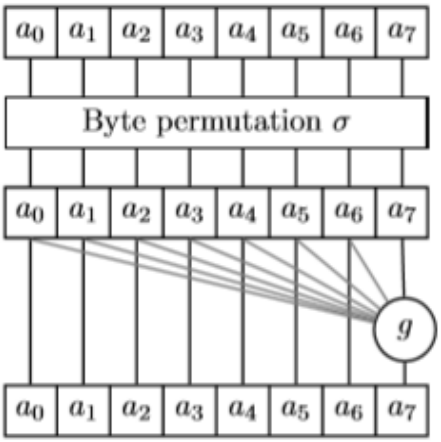

Figure 7: One round i of the AUT64 block cipher as used in VW-2 and VW-3. a0,...,a7 is the 8bytestateofthecipher, g(a0,...,a7,keyi) the round function.

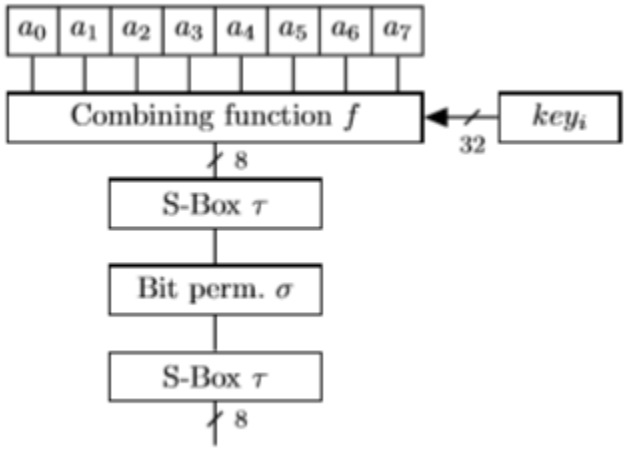

Figure 8: One round function g of the AUT64 block cipher as used in VW-2 and VW-3. a0,...,a7 is the 8-byte state of the cipher, key i the round key.

1.5.3 VW-4 Scheme

Στα νεότερα οχήματα όπως αυτά χαρακτηρίζονται περίπου από το 2009 και μετά, εντοπίστηκε ότι το RKE σύστημα που χρησιμοποιήθηκε είχε την ίδια κωδικοποίηση και δομή πακέτου με αυτή του VW-3 συστήματος, αλλά με διαφορετικό start pattern και την απουσία του αλγορίθμου κρυπτογράφησης AUT64. Η ανάλυση αυτού του συστήματος αποκάλυψε ότι το σχετικό ECU firmware χρησιμοποιούσε τον αλγόριθμο κρυπτογράφησης XTEA για την κρυπτογράφηση του πακέτου μετάδοσης με δομή ίδια με του VW-3, όπως και προαναφέρθηκε. Ο αλγόριθμος XTEA βασίζεται σε μια δομή 64 γύρων του Feistel με μέγεθος block 64-bits, καθώς και ένα κλειδί των 128-bit. Αξίζει να τονιστεί ότι η πιο γνωστή επίθεσης κρυπτανάλυσης κατά του XTEA βρίσκεται ακόμα σε θεωρητικό στάδιο.

Όπως θα δούμε και παρακάτω, στο σενάριο της συγκεκριμένης εργασίας το σύστημα που χρησιμοποιείται είναι το τελευταίο (VW-4 Scheme). Επίσης, σε αυτό το σημείο πρέπει να αναφερθεί ότι δε θα επεκταθούμε στον τρόπο κρυπτανάλυσης του αλγόριθμου κρυπτογράφησης του τηλεχειριστηρίου, καθώς γίνεται χρήση της μεθόδου brute force, οπότε δεν κρίνεται απαραίτητο.

Κεφάλαιο 2: Components

2.1 Hardware

Σε αυτό το κεφάλαιο λοιπόν θα αναφερθούν αναλυτικά τα επιμέρους εργαλεία, από την σκοπιά του hardware, που κρίθηκαν απαραίτητα με σκοπό την εκπόνηση της τρέχουσας εργασίας.

2.1.1 Laptop

Figure 9: Laptop

Με δεδομένο το γεγονός ότι το πείραμα λαμβάνει μέρος σε εξωτερικό χώρο, χρησιμοποιήθηκε ένα laptop δεκατεσσάρων ιντσών (14”) για ευχρηστία και εύκολη μεταφερσιμότητα, με έναν επεξεργαστή (intel i5 8ης γενιάς), για λόγους ισχυρής επεξεργαστικής ισχύς για την διαδικασία του brute force attack και σε συνδυασμό με μια μνήμη RAM των 8GB, καθώς το λειτουργικό σύστημα που χρησιμοποιήθηκε κρίθηκε αναγκαίο να εγκατασταθεί σε Virtual Machine.

2.1.2 HackRF One

Figure 10: HackRF One

Το HackRF One της εταιρίας "Great Scott Gadgets" είναι μια open source hardware πλατφόρμα, σχεδιασμένη για τον έλεγχο και την ανάπτυξη σύγχρονων και εξελιγμένων ραδιοτεχνολογιών. Αποτελεί μία περιφερειακή συσκευή SDR (Software Defined Radio) ικανή να λαμβάνει, αλλά και να εκπέμπει ραδιοσήματα από 1MHz έως και 6GHz με ρυθμό δειγματοληψίας έως και είκοσι εκατομμύρια δείγματα το δευτερόλεπτο. Η υποστήριξη τέτοιου μεγέθους δειγματοληψίας εξαρτάται άμεσα και από τον επεξεργαστή του laptop που συνδέεται. Για την ορθή λειτουργία του οι ελάχιστες απαιτήσεις είναι ένα καλώδιο micro USB καθώς και μια κεραία ANT500.

Αναλυτικότερα τα τεχνικά χαρακτηριστικά καταγράφονται παρακάτω:

- 1 MHz to 6 GHz operating frequency
- half-duplex transceiver
- up to 20 million samples per second
- 8-bit quadrature samples (8-bit I and 8-bit Q)
- compatible with GNU Radio, SDR#, and more
- software-configurable RX and TX gain and baseband filter
- software-controlled antenna port power (50 mA at 3.3 V)
- SMA female antenna connector
- SMA female clock input and output for synchronization
- convenient buttons for programming
- internal pin headers for expansion
- Hi-Speed USB 2.0
- USB-powered
- open source hardware

2.1.3 Antenna ANT500

Figure 11: ANT500

Η τηλεσκοπική κεραία ANT500 της εταιρίας "Great Scott Gadgets" είναι σχεδιασμένη να λειτουργεί από τα 75 MHz έως και το 1 GHz, με το μήκος της να φτάνει τα 20 cm έως και τα 88 cm. Δεδομένου του εύρους συχνοτήτων που ψάχνουμε η κεραία ANT500 είναι ιδανική.

2.1.4 Car Key Fob

Figure 12: Audi Key Fob

Όπως προαναφέρθηκε οι δοκιμές και η δειγματοληψία του σήματος πραγματοποιήθηκαν σε ένα τηλεχειριστήριο αυτοκινήτου της εταιρίας Audi (χρονολογίας 2011). Τα βασικά χαρακτηριστικά του είναι:

a. Εκπομπή σήματος σε συχνότητα ~434 Mhz.
b. Transponder Chip: Megamos Crypto ID48.
c. Modulation: ASK (Amplitude-shift keying) ~ OOK (On-Off Keying)
d. FCC ID.

Εάν υπάρχει φυσική πρόσβαση στο τηλεχειριστήριο, όλες οι βασικές πληροφορίες που προαναφέρθηκαν βρίσκονται στην εσωτερική μεριά του κύριου κλειδιού.

2.2 Software

Τα επιμέρους λογισμικά που έλαβαν χώρα για τον εναρμονισμένο συνδυασμό των προαναφερθέντων συσκευών, αλλά και για τα εκάστοτε στάδια του εντοπισμού, της συλλογής, ανάλυσης, καθώς και εκπομπής του σήματος ποικίλουν ανάλογα με την καταλληλόλητα, προτίμηση και χρηστικότητά τους.

Παρακάτω, καταγράφονται αναλυτικότερα το καθένα από αυτά.

2.2.1 Operating System

Το λειτουργικό σύστημα που προτείνεται για την λειτουργία του HackRF One είναι το distro του "pentoo" (gentoo-based), καθώς έχει προεγκατεστημένα όλα τα συστατικά στοιχεία που είναι συμβατά με το HackRF, όπως το GNU Radio, αλλά και τη βιβλιοθήκη του HackRF. Το λειτουργικό σύστημα που τελικά χρησιμοποιήθηκε είναι το Ubuntu

16.04.6 λόγω χρηστικότητας και καλύτερου χειρισμού.

2.2.2 Gqrx SDR

Το Gqrx είναι ένα SDR (Software Defined Radio) λογισμικό ανοιχτού κώδικα βασισμένο στο GNU Radio που υποστηρίζει αρκετές SDR συσκευές συμπεριλαμβανομένου τις Air spy, Fun cube Dongles, rtl-sdr, HackRF and USRP.

Κάποια βασικά χαρακτηριστικά του είναι τα εξής:

- Discover devices attached to the computer.
- Process I/Q data from the supported devices.
- Change frequency, gain and apply various corrections (frequency, I/Q balance).
- AM, SSB, CW, FM-N and FM-W (mono and stereo) demodulators.
- Special FM mode for NOAA APT.
- Variable band pass filter.
- AGC, squelch and noise blanker.
- FFT plot and waterfall.
- Record and playback audio to / from WAV file.
- Record and playback raw baseband data.
- Spectrum analyzer mode where all signal processing is disabled.
- Basic remote control through TCP connection.
- Streaming audio output over UDP.

Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε για τον εντοπισμό του σήματος, λόγω του καλά σχεδιασμένου user interface του.

2.2.3 Osmocom_fft

Άλλο ένα SDR λογισμικό που αποτελεί ακόμη ένα εργαλείο ανάλυσης φάσματος. Χρησιμοποιήθηκε σε συνδυασμό με το HackRF για την συλλογή/καταγραφή του ζητούμενου σήματος που εκπέμφθηκε από το κλειδί.

2.2.4 GNU Radio Companion

Το GNU Radio είναι ένα λογισμικό ανοιχτού κώδικα που παρέχει την δυνατότητα για την ανάλυση και επεξεργασία για την υλοποίηση ραδιοσημάτων. Μπορεί να χρησιμοποιηθεί

άμεσα με πολλών ειδών SDR εξοπλισμού χαμηλού κόστους με τελικό σκοπό την παραγωγή ραδιοσημάτων. Χρησιμοποιείται ευρέως σε περιβάλλοντα έρευνας, βιομηχανίας, ακαδημαϊκής κοινότητας και κυβερνητικά περιβάλλοντα για την υποστήριξη τόσο της έρευνας ασύρματων επικοινωνιών όσο και των ραδιοσυστημάτων στον πραγματικό κόσμο.

2.2.5 Inspectrum

Το inspectrum αποτελεί ένα εργαλείο για την ανάλυση ενός σήματος, κυρίως προερχόμενο από ένα δέκτη SDR, όπως αυτό του HackRF. Χρησιμοποιήθηκε για την εξαγωγή πληροφορίας του καταγεγραμμένου σήματος σε κατάλληλη μορφή τέτοια ώστε στη συνέχεια να μπορεί να μετατραπεί σε μια δυαδική ακολουθία από «0» και «1».

Αξίζει να σημειωθεί ότι μέσω της κοινότητας του Github υπάρχει διαθέσιμο και το "DSpectrum" που αποτελεί μια εμπλουτισμένη έκδοση του Inspectrum. Υπάρχει διαθέσιμος και ο οδηγός εγκατάστασης. Προσφέρει αυτοματοποιημένες διαδικασίες αναγνώρισης του Manchester encoder, καθώς επίσης και της μετατροπής του σήματος δειγματοληψίας σε δυαδική μορφή. Όλα αυτά προσφέρονται μέσω ενός καλοσχεδιασμένου web interface για την ευκολότερη ανάλυση του σήματος. Δυστυχώς, περιείχε αρκετά bugs και για αυτό το λόγο δεν χρησιμοποιήθηκε για την εκπόνηση της συγκεκριμένης εργασίας.

Τα λογισμικά που αναφέρθηκαν παραπάνω είναι αυτά που προτιμήθηκαν μέσα από μια πληθώρα παρόμοιων λογισμικών ανοιχτού κώδικα που ποικίλουν ανάλογα όχι μόνο με την SDR συσκευή της προτίμησης μας αλλά και άλλων συσκευών όπως για παράδειγμα το Raspberry Pi και Arduino.

Κεφάλαιο 3: Software Installation Guide

Για να επιτευχθεί λοιπόν ο τελικός μας στόχος θα πρέπει να ξεκινήσουμε από τα βασικά στάδια της εγκατάστασης των επιμέρους λογισμικών. Παρακάτω παρουσιάζεται ένας αναλυτικός οδηγός της εγκατάστασης και παραμετροποίησης τους.

Δεδομένου ότι έχουμε προμηθευτεί και εγκαταστήσει το λειτουργικό μας σύστημα Ubuntu 16.04.6 x64, ξεκινάμε ευθύς αμέσως με την εγκατάσταση των βιβλιοθηκών του HackRF One, αλλά και του GNU Radio, Gqrx, inspectrum και των υπόλοιπων προ απαιτούμενων συστατικών λογισμικών.

Αρκεί να ανοίξουμε ένα τερματικό και να τρέξουμε τις παρακάτω εντολές:

1. Εγκατάσταση των βιβλιοθηκών του HackRF One:
    - $sudo apt-get install hackrf -y

2. Εγκατάσταση του GNU Radio:
    - $sudo apt-get install gnuradio -y

3. Εγκατάσταση του git για την μετέπειτα χρήση άλλων software από τη κοινότητα του Github:
    - $sudo apt-get install git -y

4. Εγκατάσταση του Inspectrum:
    - $sudo apt-get install inspectrum -y

5. Εγκατάσταση του Gqrx:
    - $sudo add-apt-repository -y ppa:gqrx/gqrx-sdr
    - $sudo apt-get update
    - $sudo apt-get install gqrx-sdr

6. Εγκατάσταση Audacity:
    - $sudo add-apt-repository ppa:ubuntuhandbook1/audacity
    - $sudo apt-get update
    - $sudo apt-get install audacity

Λόγω της πολυπλοκότητας του GNU Radio αρκετές φορές εμφανίζονται και ζητήματα συμβατότητας με αλληλεξαρτώμενες βιβλιοθήκες και εργαλεία όπως αυτά της python,

όπου καθίσταται απαραίτητη η ύπαρξη της έκδοσης python 2.7, αλλά και αυτή του cmake, εργαλείου για την δημιουργία πακέτων λογισμικού όπως αυτά που θα παρουσιαστούν παρακάτω για την δημιουργία του block στο GNU Radio.

Η έκδοση του cmake θα πρέπει να είναι μεγαλύτερη ή ίση της version 3. Στην περίπτωση που δεν είναι προεγκατεστημένο στο σύστημα, αρκεί να τρέξουμε τις παρακάτω εντολές:

7. Εγκατάσταση του cmake:
    - $sudo apt-get install software-properties-common
    - $sudo add-apt-repository ppa:george-edison55/cmake-3.x
    - $sudo apt-get update

Κεφάλαιο 4: Attack Scenario

Σε αυτό το κεφάλαιο λοιπόν θα αποδομηθούν και αναλυθούν τα επιμέρους στάδια - σε πρώτη φάση, από τον εντοπισμό, την συλλογή/καταγραφή, αλλά και την ανάλυση του σήματος, καθώς επίσης - σε δεύτερη φάση από την εκ νέου παραγωγή του σήματος με την χρήση της εξαντλητικής μεθόδου brute force όπως προαναφέρθηκε και τελικά της εκπομπής του.

4.1 Locating the Signal

Ας περιγράψουμε βήμα προς βήμα την διαδικασία εντοπισμού του σήματος. Αυτό θα το επιτύχουμε με το λογισμικό του Gqrx ψάχνοντας προσεγγιστικά κοντά στην συχνότητα που γνωρίζουμε ότι εκπέμπει το κλειδί του οχήματος (~434).

Αφού έχουμε συνδέσει το HackRF One με το Laptop μας, εκτελούμε την παρακάτω εντολή μέσα σε ένα τερματικό που θα μας ανοίξει αρχικά ένα παράθυρο ώστε να ρυθμίσουμε τις παραμέτρους που δέχεται το Gqrx. Παραμετροποιούμε επομένως τις παρακάτω παραμέτρους ως εξής:

- ✓ τύπος συσκευής: HackRF One,
- ✓ ρυθμό δειγματοληψίας: 2 Msps (για λόγους επεξεργαστικής ισχύς),
- ✓ το εύρος (bandwidth) που θα αναζητήσουμε το σήμα: 1 MHz, καθώς γνωρίζουμε ήδη ότι είναι κοντά στα 434 MHz.

Με την εκτέλεση της εντολής $gqrx, ανοίγει το παρακάτω παράθυρο με τις επιθυμητές τιμές που προαναφέραμε:

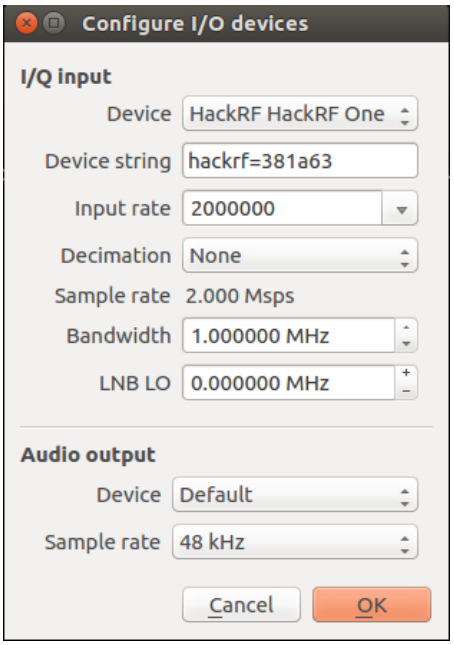

Figure 13: GQRX Configuration Menu

Αφού πατήσουμε "OK", ανοίγει το User Interface. Με ένα κλικ από το κουμπί του τηλεχειριστηρίου του κλειδιού αμέσως παρατηρούμε την ύπαρξη του σήματος:

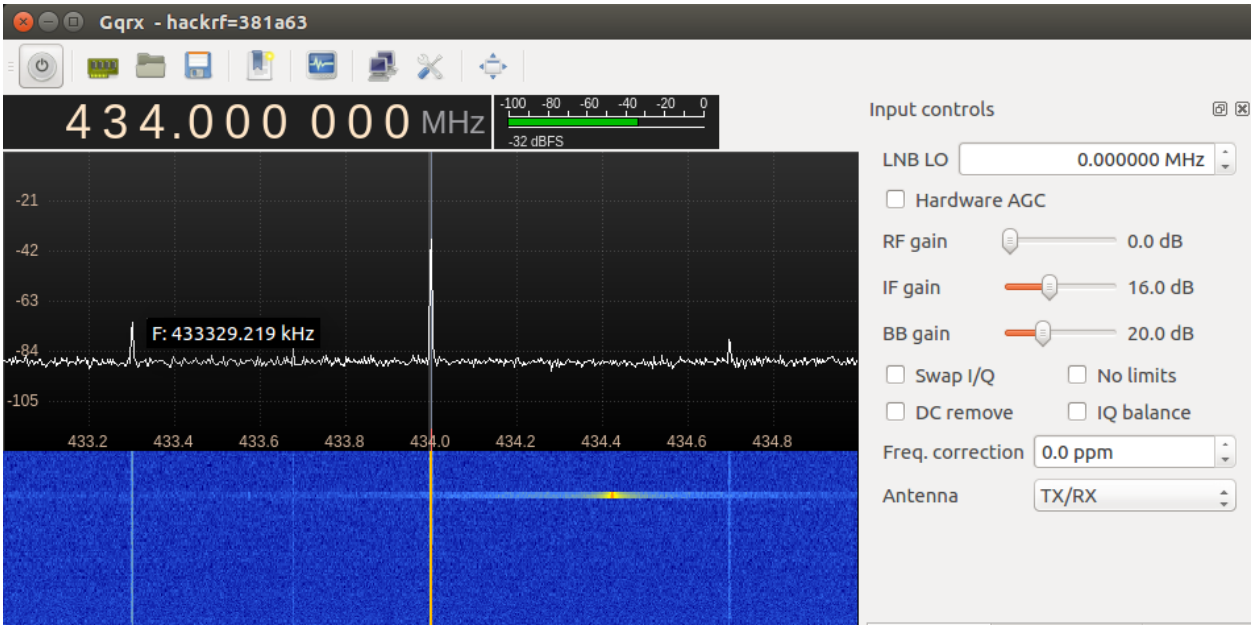

Figure 14: GQRX Capturing a Pulse

Με μια πρώτη ματιά, παρατηρούμε ότι το σήμα εκπέμπεται με συχνότητα κοντά στα 434.4 MHz. Προσαρμόζοντας επομένως τη συχνότητα δειγματοληψίας και πατώντας το κουμπί μερικές φορές ακόμα, καταλήγουμε στο συμπέρασμα ότι η πραγματική συχνότητα εκπομπής του σήματος είναι τα 434.43 MHz.

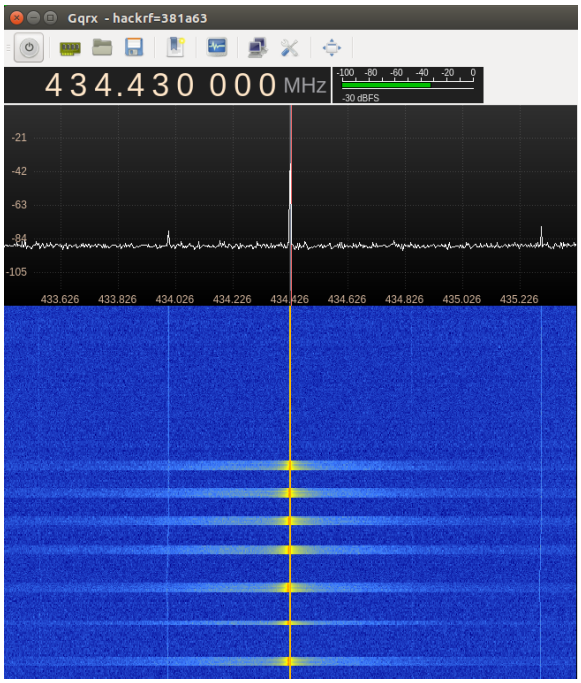

Figure 15: GQRX Key Fob's Frequency

4.2 Capturing the Signal

Το επόμενο βήμα αποτελεί την συλλογή του σήματος με στόχο την μετέπειτα ανάλυσή του. Αυτό το στάδιο θα πραγματοποιηθεί μέσω του λογισμικού osmocom_fft, πάλι εκτελώντας την παρακάτω εντολή, όπου θα μας ανοίξει το αντίστοιχο user interface (πρώτα θα πρέπει να κάνουμε reset το HackRF One, πατώντας απλά το κατάλληλο κουμπί πάνω στη συσκευή):

$osmocom_fft

Figure 16: Osmocom_fft Configuration

Και σε αυτό το σημείο θα χρειαστεί να παραμετροποιήσουμε τις παραμέτρους ως εξής:

- ✓ Center Frequency: 434.2 MHz (ο λόγος είναι ότι εάν θέσουμε την συχνότητα ακριβώς στην συχνότητα που εκπέμπει το τηλεχειριστήριο του κλειδιού το λαμβανόμενο σήμα θα επισκιαστεί. Αυτό οφείλεται, όπως παρατηρείται και από το παραπάνω σχήμα ότι το HackRF εκπέμπει στην κεντρική συχνότητα που του θέσαμε – όπως γίνεται στις Rolljam επιθέσεις).
- ✓ RF Gain (dB): 0
- ✓ IF Gain (dB): 16
- ✓ BB Gain (dB): 16
- ✓ Bandwidth: 1 MHz
- ✓ Sample Rate: 2 Msps
- ✓ File Name: /home/ubuntu/Desktop/lock.cfile (για λόγους ευκολίας ονομάσαμε το εκάστοτε σήμα που καταγράφουμε, ανάλογα με την λειτουργία του κουμπιού που πατάμε στο τηλεχειριστήριο).

Το αποτέλεσμα διαμορφώνεται ως εξής:

Figure 17: osmocom_fft Capturing the Signal

Καθώς επιθυμούμε να αποσπάσουμε την αναγκαία πληροφορία από το σήμα του τηλεχειριστηρίου, καταγράψαμε 10 δείγματα ανά λειτουργία κουμπιού. Οπότε διαμορφώθηκαν τα αρχεία σε:

- ✓ /home/ubuntu/Desktop/lock.cfile
- ✓ /home/ubuntu/Desktop/unlock.cfile
- ✓ /home/ubuntu/Desktop/trunk.cfile

4.3 Sampling the Signal

Αυτή η ενότητα είναι και η πιο σημαντική, καθώς θα πρέπει με όσα στοιχεία γνωρίζουμε

μέχρι στιγμής να μπορέσουμε να διακρίνουμε το μοτίβο των πακέτων που έχουμε συλλέξει. Το εργαλείο λογισμικού που θα μας βοηθήσει στην έρευνά μας είναι το inspectrum. Αλλά δεν είναι αυτό μόνο αρκετό. Χρειάστηκε επίσης να κατασκευάσουμε έναν κώδικα σε Python ώστε να μετατρέπει τα δεδομένα που προκύπτουν από το inspectrum σε δυαδικό κώδικα «0» και «1».

Κατά την καταγραφή των λογισμικών στο Κεφάλαιο 2, έγινε αναφορά στο DSpectrum, το οποίο μετέτρεπε με αυτοματοποιημένες διαδικασίες το δειγματοληπτημένο σήμα σε δυαδική μορφή.

Η διαδικασία που θα περιγράψουμε παρακάτω χρειάστηκε να πραγματοποιηθεί επανειλημμένα για κάθε ένα από τα σήματα που συλλέχθηκαν και για κάθε λειτουργία του κουμπιού του τηλεχειριστηρίου, καθώς πρέπει να γίνουν διακριτά τα:

- ✓ Preamble bytes
- ✓ Payload bytes
- ✓ Button bytes

Όπου,

για κάθε σήμα το preamble θα πρέπει να είναι μια σταθερή ευδιάκριτη ακολουθία από «0» και «1», ενώ η ακολουθία των bits στο payload θα πρέπει σε κάθε πάτημα του κουμπιού να διαφέρει κάθε φορά. Και τέλος, να διακρίνουμε τα bits εκείνα που χαρακτηρίζουν τη λειτουργία του κουμπιού – ίδια ακολουθία bits για την εκάστοτε λειτουργία.

Αρκεί επομένως να εκτελέσουμε την εντολή, πρώτα για το κουμπί του κλειδώματος:

$inspectrum /home/ubuntu/Desktop/lock.cfile

Μια πρώτη εικόνα που παίρνουμε είναι η εξής:

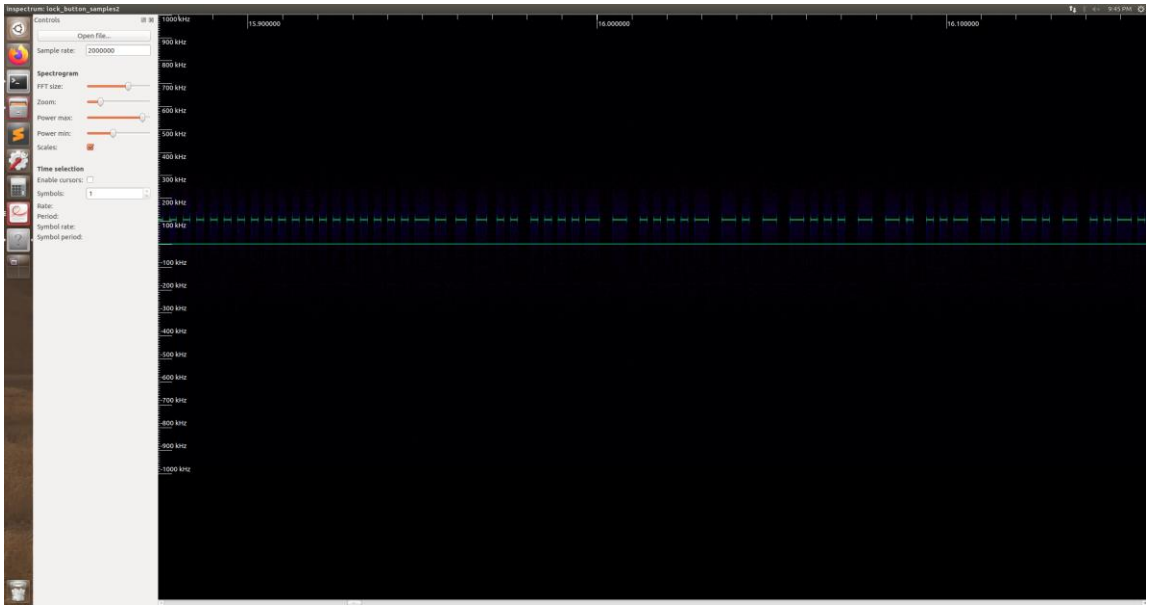

Figure 18:Inspectrum

Στην αριστερή στήλη παρατηρούμε ότι βρίσκονται κάποια πεδία που ακόμα δεν έχουν πάρει κάποια τιμή. Η τιμή αυτών θα αρχίσει να μεταβάλλεται όταν ξεκινήσει η διαδικασία δειγματοληψίας του σήματος. Τα πεδία αυτά είναι τα εξής:

- ✓ Rate
- ✓ Period
- ✓ Symbol rate
- ✓ Symbol period

Παρακάτω περιγράφεται η διαδικασία αυτή βήμα προς βήμα:

1. Εντοπίζουμε το παλμό με το μικρότερο μήκος και δειγματοληπτούμε βάσει αυτού.
2. Αφού εντοπίσουμε τον μικρότερο παλμό, στο πεδίο «symbols» συμπληρώνουμε τον αριθμό 5 για τα 5 πρώτα σύμβολα και τα ενεργοποιούμε επιλέγοντας το πεδίο «enable cursors».

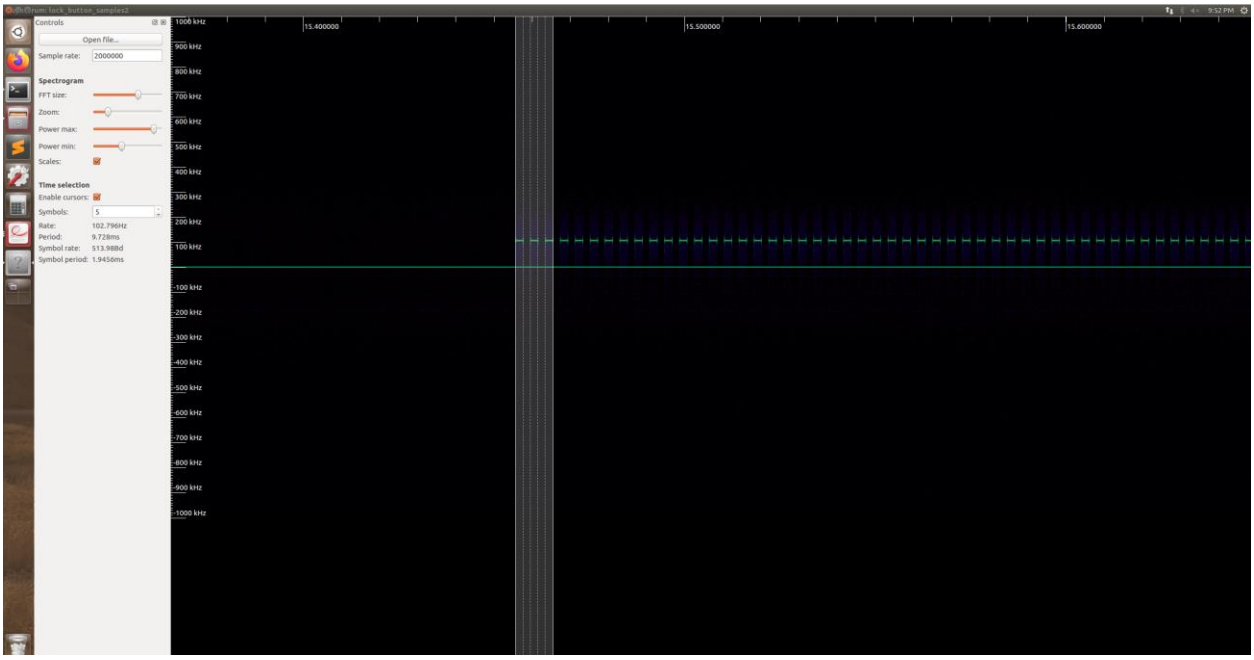

Figure 19: Inspectrum Sampling the Signal

Οι τιμές που θα διαμορφωθούν στο τέλος της διαδικασίας αυτής είναι κομβικές για την παραγωγή του σήματος στο επόμενο στάδιο.

3. Συνεχίζουμε να αυξάνουμε τον αριθμό των συμβόλων μέχρι να καλύψουμε ολόκληρο το σήμα. Αξίζει να σημειωθεί ότι θα πρέπει να είμαστε πολύ προσεχτικοί καθ' όλη την διάρκεια της διαδικασίας αυτής και κάθε σύμβολο θα πρέπει να εφάπτεται ακριβώς σε κάθε αρχή και τέλος του εκάστοτε παλμού, διαφορετικά θα εξάγουμε λανθασμένα αποτελέσματα.

Το τελικό αποτέλεσμα διαμορφώνεται ως εξής:

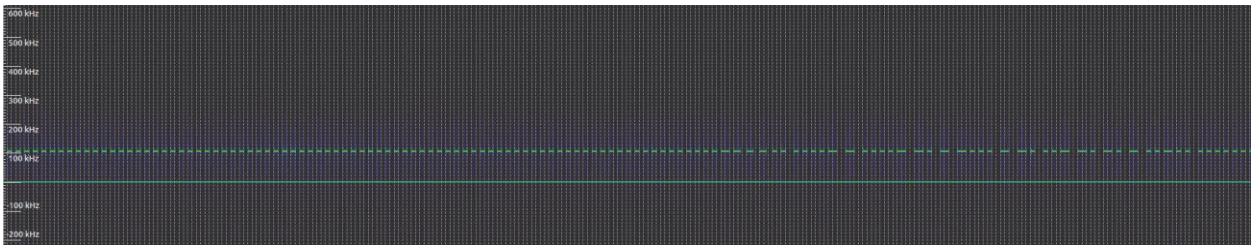

Figure 20: Sampling the Signal 2

4. Στην συνέχεια, ξέροντας ότι το σήμα είναι διαμορφωμένο κατά πλάτος (ASK) ή διαφορετικά ότι αντιπροσωπεύει την λειτουργία του On-Off Keying (OOK), κάνουμε δεξί κλίκ πάνω στο σήμα και επιλέγουμε «add amplitude plot» κάτω από την επιλογή «add derived plot», και προσαρμόζουμε την λεπτή κόκκινη γραμμή που εμφανίζεται να συμπίπτει απόλυτα με το λαμβανόμενο σήμα, όπως φαίνεται παρακάτω:

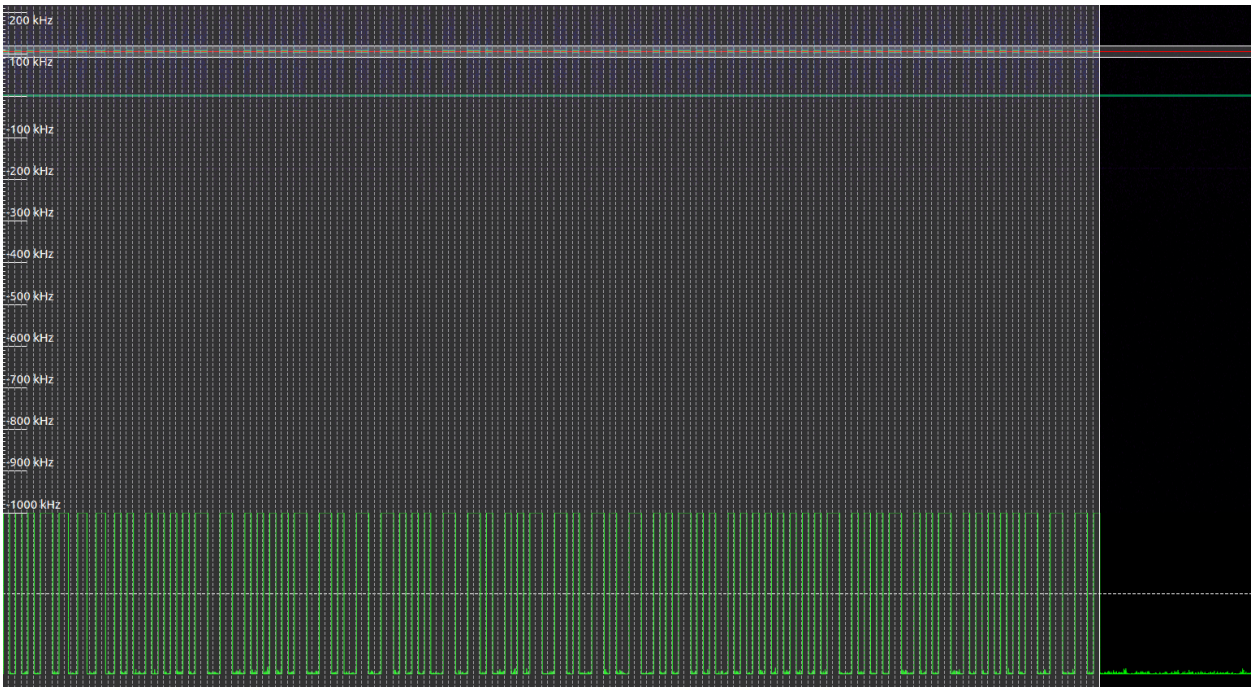

Figure 21: Inspectrum - Add Amplitude Plot

5. Πατάμε ξανά δεξί κλικ και επιλέγουμε «extract symbols» και στην συνέχεια «to stdout», όπου αυτό θα έχει ως αποτέλεσμα την εξαγωγή δεδομένων στο τερματικό μας.

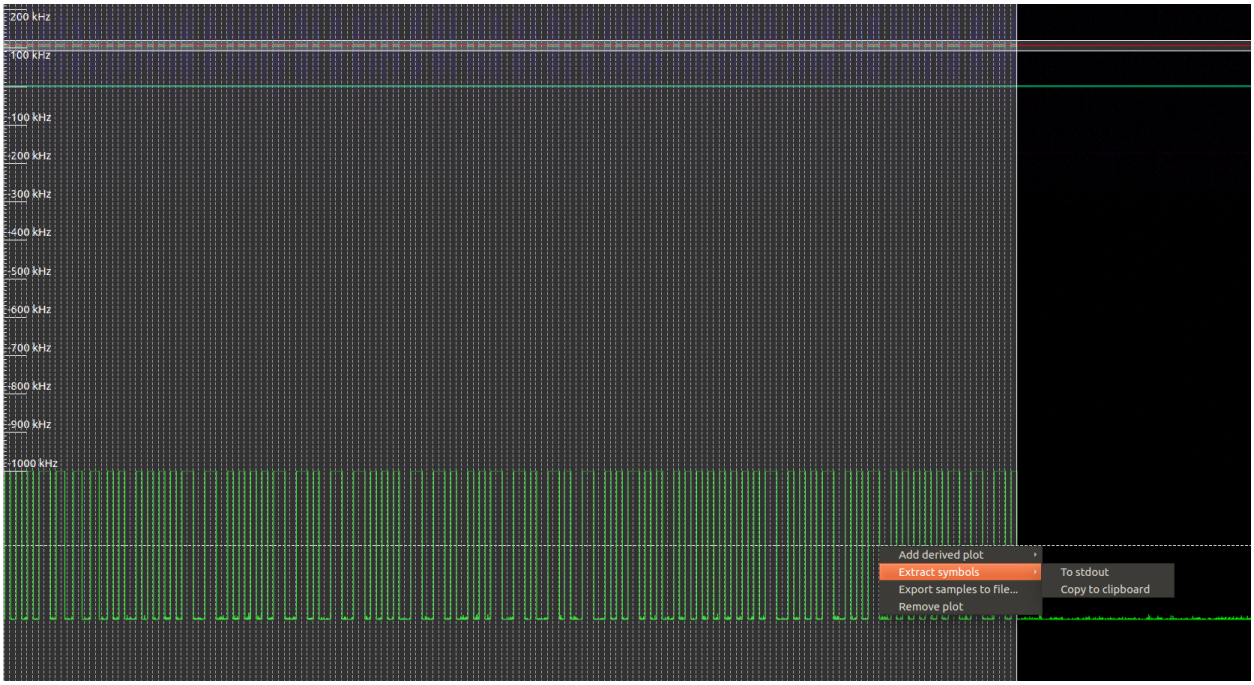

Figure 22: Inspectrum - Extracting Symbols

6. Στην εξερχόμενη πληροφορία που εμφανίζεται στο τερματικό μας, εάν παρατηρήσουμε προσεχτικά θα δούμε ότι υπάρχουν αρνητικές αλλά και θετικές τιμές αυτής:

Figure 23: Extracted Symbols

7. Αυτή η διαφοροποίηση των τιμών μας οδήγησε στην ανάγκη για τον σχεδιασμό ενός κώδικα σε Python κατά τον οποίο, όσες τιμές έχουν αρνητικό πρόσημο θα αναπαρίστανται με τον δυαδικό αριθμό «0», ενώ αντίθετα αυτές με θετικό πρόσημο με τον δυαδικό αριθμό «1». Ο κώδικας είναι ο παρακάτω:

```python
A = [1.34309, -0.999766, 1.47431, -0.990935,

B = []

i = 0
j = 0

for s in A:
    if s > 0:
        i=1
        j+=1
    else:
        i=0
        j+=1
    B.append (i)

print B
print (j)
```

Figure 24: Raw to Binary - Python Code

8. Στην συνέχεια εκτελώντας το κώδικα ($python raw_to_binary.py), προκύπτει η παρακάτω ακολουθία από bits την οποία και αποθηκεύουμε σε ένα αρχείο ώστε να έχουμε την δυνατότητα να συγκρίνουμε τα αποτελέσματα αργότερα:

```
ubuntu@ubuntu:~$ python raw_to_binary.py
[1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1
, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1
, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1
, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1
, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1,
0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1
, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0,
1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0,
 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1]
423
```

Figure 25: Binary Representation of Raw Input

9. Ακολουθήσαμε την ίδια διαδικασία και για τις τρεις διαφορετικές λειτουργίες του κλειδιού ώστε να κατανοήσουμε την δομή του σήματος.

4.4 Analyzing the Signal

Αφού συλλέξαμε όσα περισσότερα δείγματα μπορούσαμε, προσπαθήσαμε στην συνέχεια να διαχωρίσουμε τα bits εκείνα που αποτελούν το preamble, την μεταβαλλόμενη κρυπτογραφημένη ακολουθία όπου και θα εφαρμόσουμε αργότερα την εξαντλητική μέθοδο brute force, καθώς επίσης και το μέρος της ακολουθίας που καθορίζει την λειτουργία του τηλεχειριστηρίου.

Παρατηρώντας και αναλύοντας την ακολουθία των bits που αποτελούν το σήμα καταλήξαμε στα παρακάτω αποτελέσματα:

1. Preamble: 256 bits
2. Start pattern: 24 bits
3. Payload/secret key:128 bits
4. Button: 15 bits

Όπου,

1. Preamble: μια ακολουθία σε μοτίβο «10» * 128 (σύνολο 156 bits),
2. Start pattern: μια σταθερή ακολουθία και για τις τρεις λειτουργίες του τηλεχειριστηρίου (1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0),
3. Payload/secret key: διαφορετικό για κάθε σήμα, καθώς εφαρμόζεται ο αλγόριθμος κρυπτογράφησης,
4. Button: σταθερό για κάθε διαφορετική λειτουργία του τηλεχειριστηρίου, όπως φαίνεται παρακάτω,
    a. Lock button: 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1
    b. Unlock button: 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1
    c. Trunk button: 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1

Αρκεί πλέον, να κατασκευάσουμε τον δικό μας κώδικα εξαντλητικής μεθόδου brute force, να το συνδυάσουμε με ένα custom block που θα δημιουργήσουμε με την βοήθεια του GNU Radio ώστε να μπορέσουμε τελικά να δημιουργήσουμε το σήμα από την αρχή και να το μεταδώσουμε μέσω του HackRF One.

4.5 Creating Blocks with GNU Radio

Το GNU Radio μας παρέχει ποικίλες δυνατότητες για την συλλογή, επεξεργασία και μετάδοση ενός σήματος. Σε αυτή την υποενότητας λοιπόν θα παρουσιάσουμε την διαδικασία με την οποία μπορούμε να παράξουμε τα δικά μας "out of tree modules", καθώς επίσης και τον κώδικα Python που παράγει την εξαντλητική δυαδική ακολουθία για την εύρεση του κλειδιού.

Πριν ξεκινήσουμε με την διαδικασία κατασκευής του module, πρέπει διακρίνουμε και να καθορίσουμε τα χαρακτηριστικά του:

1. Είσοδο: 0. Δεν απαιτείται κάποια παράμετρο εισόδου στο module μας, καθώς ο κώδικας θα παράγεται εντός του. Το module μας λοιπόν αποτελεί ένα source block.
2. Έξοδος: 1. Η έξοδος του block θα είναι η ακολουθία από «0» και «1». Κατά την κατασκευή του block θα πρέπει να δηλώσουμε και τον τύπο δεδομένων που θα εξάγονται. Αφού θέλουμε να εξάγουμε bits, ο τύπος δεδομένων που θα δηλώσουμε θα είναι byte.
3. Παράμετροι: 1. Η παράμετρος που θα ορίσουμε θα είναι το πλήθος των bits της ζητούμενης ακολουθίας.

Αφού καθορίσαμε τα χαρακτηριστικά του module μας, ας αναλύσουμε τους 4 διαφορετικούς τύπους των blocks που μπορούμε να κατασκευάσουμε. Αξίζει να σημειωθεί ότι κάθε τύπος block έχει και διαφορετική δομή, οπότε θα πρέπει να επιλέξουμε προσεκτικά την επιλογή μας. Οι 4 διαφορετικοί τύποι blocks είναι οι εξής:

1. Synchronous Block: Το synchronous block επιτρέπει στους χρήστες να γράφουν μπλοκ που εξάγουν και παράγουν ίσο αριθμό αντικειμένων ανά θύρα. Ένα synchronous block μπορεί να έχει οποιοδήποτε αριθμό εισόδων ή εξόδων. Όταν ένα synchronous block έχει μηδενικές εισόδους, ονομάζεται source, ενώ όταν αυτό έχει μηδενικές εξόδους, ονομάζεται sink.
2. Decimation Block: Το decimation block είναι ένας άλλος τύπος block σταθερού ρυθμού, όπου ο αριθμός των στοιχείων εισόδου είναι ένα σταθερό πολλαπλάσιο του αριθμού των στοιχείων εξόδου (input items = noutput_items*decimation).
3. Interpolation Block: Το interpolation block είναι ένας άλλος τύπος block σταθερού ρυθμού, όπου ο αριθμός των στοιχείων εξόδου είναι σταθερό πολλαπλάσιο του αριθμού των στοιχείων εισόδου (input items = noutput_items/interpolation).
4. Basic Block: Το basic block δεν παρέχει καμία σχέση μεταξύ του αριθμού των στοιχείων εισόδου και του αριθμού των στοιχείων εξόδου. Όλα τα άλλα block είναι απλώς απλοποιήσεις του basic block. Οι χρήστες θα πρέπει να επιλέξουν να

κληρονομήσουν από το basic block, όταν τα υπόλοιπα block δεν είναι κατάλληλα.

Για την επίτευξη της δημιουργίας του module και κατ' επέκταση του block μας, θα χρησιμοποιήσουμε ένα εργαλείο που μας προσφέρει το GNU Radio, το gr_modtool.

Ανοίγουμε ένα τερματικό και ακολουθούμε τα παρακάτω βήματα:

1. $ gr_modtool newmod brute4ce
    a. Θα δημιουργήσει ένα module (φάκελο) με το όνομα gr-tutorial στο τρέχον directory.
2. $ gr_modtool add -t source bruteforce_b
    a. Θα δημιουργήσει τις κατάλληλες default συναρτήσεις για το block μας με μηδενική είσοδο, κάνοντας έτσι όπως υποδεικνύει η παράμετρος ένα source block. Η κατάληξη «_b» υποδηλώνει ότι ο τύπος εξόδου το block μας είναι byte.
    b. Κατά την εκτέλεση της παραπάνω εντολής θα μας ζητηθούν τα εξής:
        i. Language (python/cpp): επιλέγουμε python, καθώς ο κώδικας που θα γραφτεί το block μας θα είναι σε Python.
        ii. Enter valid argument list, including default arguments: keylength, όπου keylength ο αριθμός του κλειδιού των μεταβαλλόμενων bits.
        iii. Add Python QA code? [Y/n]: Y, όπου δημιουργεί και παραμετροποιεί καταλλήλως τα αρχεία: python/bruteforce_b.py, python/qa_bruteforce_b.py, python/CMakeLists.txt, grc/add_bruteforce_b.xml, grc/CMakeLists.txt.
3. Τα αρχεία που θα χρειαστεί να παραμετροποιήσουμε καταλλήλως είναι τα: "python/bruteforce_b.py" και "grc/tutorial_bruteforce_b.xml ", τα οποία θα αναλύσουμε στην επόμενη υποενότητα.

4.6 Editing the bruteforce_b.py

Σε αυτή την υποενότητα θα παρουσιάσουμε τον κώδικα Python που θα περιέχει την εξαντλητική μέθοδο παραγωγής του δυαδικού κώδικα για την εύρεση του κλειδιού, τις συναρτήσεις που χρησιμοποιήθηκαν αλλά και την δομή των αρχείων όπως αυτά κατασκευάστηκαν από το gr_modtool.

Παρακάτω παρατίθεται ο κώδικας του bruteforce_b:

#Start

```
import numpy as np
from itertools import product
from gnuradio import gr

class bruteforce_b(gr.sync_block):
    """
    docstring for block bruteforce_b
    """
    def __init__(self, keylength):
        gr.sync_block.__init__(self,
                name="bruteforce_b",
                in_sig=None,
                out_sig=[np.int8])

        self.keylength = keylength

    def work(self, input_items, output_items):

        binary = '01'
        spacing = 3

        remaining = []

        room = (len(output_items[0]) - len(remaining))

         if room <= 0:
                output_items[0][:] = np.array(remaining[:len(output_items[0])])
                remaining = remaining[len(output_items[0]):]
                return output_items[0]

        packet = []
        acc_len = 0

        while len(packet) < room:
                to_attempt = product(binary, repeat=self.keylength)
                for attempt in to_attempt:
                        out = np.asarray(''.join(attempt))
                        print str(out)
                        acc_len += len(str(out))
                        packet += str(out)
                        packet += [2] * spacing
        output_items[0][:] = np.array(remaining + packet[:room])
        if acc_len > room:
                remaining = packet [room:]
```

```
        else:
            remaining = []

        return (len(output_items[0]))

#End
```

Ας αναλύσουμε τώρα τις διάφορες συναρτήσεις που αποτελούν το bruteforce_b.py.

1. def __init__(self, keylength): Προκαθορισμένη συνάρτηση που δημιουργείται κατά την κατασκευή του block εκτελώντας την εντολή gr_modtool. Στη συγκεκριμένη συνάρτηση δηλώνουμε τις αρχικές μεταβλητές του κώδικά μας, όπως το όνομα του block (bruteforce_b), την είσοδο του block (in_sig) που όπως παρατηρούμε έχει την προκαθορισμένη τιμή «None» (την κληρονόμησε από την παράμετρο "-t source"). Επιπλέον, έχουμε την μεταβλητή «out_sig». Έχοντας ως τύπο εξόδου byte, θέτουμε με την βοήθεια της βιβλιοθήκης «numpy» τον τύπο της μεταβλητής ίσο με «np.in8», όπου int8 υποδηλώνει Byte (-128 to 127). Τέλος, όπως βλέπουμε η συνάρτηση παίρνει σαν παράμετρο και την μεταβλητή «keylength» που είχαμε δηλώσει κατά την δημιουργία του block.

2. def work(self, input_items, output_items): Και αυτή η συνάρτηση είναι προκαθορισμένη από την εντολή gr_modtool. Η συνάρτηση work() αποτελεί την κύρια συνάρτηση μέσα στην οποία θα εκτελεστεί ο κώδικας του brute force. Για την υλοποίηση του κώδικα χρειάστηκε να χρησιμοποιήσουμε τις διαθέσιμες συναρτήσεις από τις βιβλιοθήκες των numpy και itertools. Η βασική συνάρτηση που δημιούργησε τις εξαντλητικές προσπάθειες για την κατασκευή των ακολουθιών από «0» και «1» είναι η εξής:
    a. product(binary, repeat=self.keylength), στην οποία εισαγάγαμε την μεταβλητή binary, όπου υποδηλώνει στην συνάρτηση product() ότι όλοι συνδυασμοί με μέγεθος keylength θα αποτελούνται από τους χαρακτήρες «0» και «1».
    b. Την εξαγωγή της παραπάνω συνάρτησης στην συνέχεια την εισαγάγαμε σε ένα πίνακα με την βοήθεια των συναρτήσεων που μας παρέχονται από την βιβλιοθήκη numpy, όπου στη συγκεκριμένη περίπτωση είναι η np.asarray().
    c. Ο υπόλοιπος κώδικας είναι υπεύθυνος για τον διαχωρισμό των ακολουθιών σε πακέτα ώστε να είναι ευδιάκριτα κατά την αποστολή του σήματος.

4.7 Editing the tutorial_bruteforce_b.xml

Η δημιουργία του «bruteforce_b.py» δεν επαρκεί μόνο για την σωστή λειτουργία του module μας. Αφού έχει κατασκευαστεί ο κώδικάς μας, θα πρέπει τώρα να είναι και διαθέσιμος μέσα από την κονσόλα του GRC. Αυτό το σκοπό εξυπηρετεί η ύπαρξη του XML αρχείου «tutorial_bruteforce_b.xml».

Άλλος ένας λόγος επεξεργασίας του προαναφερθέντος αρχείου XML είναι ότι στις περισσότερες περιπτώσεις, το εργαλείο gr_modtool δεν μπορεί να υπολογίσει όλες τις παραμέτρους από μόνο του. Στη δική μας περίπτωση βέβαια, επειδή η δομή του block δεν έχει ιδιαίτερη πολυπλοκότητα, αρκούν κάποιες βασικές μετατροπές που θα αναφερθούν αμέσως τώρα.

Ο κώδικας του tutorial_bruteforce_b.xml είναι ο κάτωθεν:

```
<?xml version="1.0"?>
<block>
	<name>bruteforce_b</name>
	<key>tutorial_bruteforce_b</key>
	<category>tutorial</category>
	<import>import tutorial</import>
	<make>tutorial.bruteforce_b($keylength)</make>

	<param>
		<name>Keylength</name>
		<key>keylength</key>
		<type>int</type>
	</param>

	<source>
		<name>out</name>
		<type>byte</type>
	</source>
</block>
```

Τα διακριτά πεδία όπως φαίνονται με μια πρώτη ματιά στον παραπάνω κώδικα είναι τα «param» και «source».

1. Στο πεδίο «param», αρκεί να δηλώσουμε τα χαρακτηριστικά της μεταβλητής μας για το πλήθος των bits που θα εφαρμοστεί η εξαντλητική μας μέθοδος. Τα

χαρακτηριστικά λοιπόν είναι τα εξής:

- ✓ Name: το όνομα του πεδίου της παραμέτρου μας που θα είναι ορατό στο block εντός του GRC.
- ✓ Key: το όνομα της μεταβλητής μας όπως εμείς την δηλώσαμε κατά την κατασκευή του module.
- ✓ Type: ο τύπος δεδομένων που θα δέχεται η μεταβλητή μας καθώς εισέρχεται στο πρόγραμμα.

2. Στο πεδίο «source» αρκεί να δηλώσουμε τα χαρακτηριστικά της εξόδου του block:
   - ✓ Name: το όνομα του πεδίου της παραμέτρου εξόδου.
   - ✓ Type: τον τύπο δεδομένων που θα εξάγονται από το block. Καθώς έχουμε να κάνουμε με bits, ο τύπος εξόδου θα είναι bytes.

Σε αυτό το σημείο, έχοντας ετοιμάσει τον κώδικά μας λοιπόν και έχουμε βεβαιωθεί για την ορθότητά του, σαν τελευταίο βήμα θα χρειαστεί να εκτελέσουμε την διαδικασία εγκατάστασής του. Αυτό θα πραγματοποιηθεί με την βοήθεια του «cmake».

Ανοίγουμε λοιπόν πάλι το τερματικό μας και αφού εισέλθουμε εντός της διεύθυνσης του module μας, δηλαδή /gr-tutorial, εκτελούμε τα παρακάτω:

- ✓ $mkdir build
- ✓ $cd build
- ✓ $cmake ../
- ✓ $make
- ✓ $sudo make install
- ✓ $sudo ldconfig

4.8 GNU Radio Flowgraph

Για τον τελικό σχεδιασμό του γράφου μέσω του GNU Radio θα χρειαστεί να συνδυάσουμε όλες τα δεδομένα και πληροφορίες που συλλέξαμε και προκύψανε κατά την διαδικασία ανάλυσης του σήματος.

Το GNU Radio μας παρέχει μια πληθώρα δυνατοτήτων μέσα από τα block του για τον σχεδιασμό ενός σήματος. Εμείς απλώς θα αναλύσουμε εκείνα τα block που κρίθηκαν απαραίτητα για τον σχηματισμό και μετάδοση του δικού μας σήματος.

Ο τελικός γράφος ροής που προέκυψε κατά την έρευνα παρουσιάζεται στην παρακάτω εικόνα.

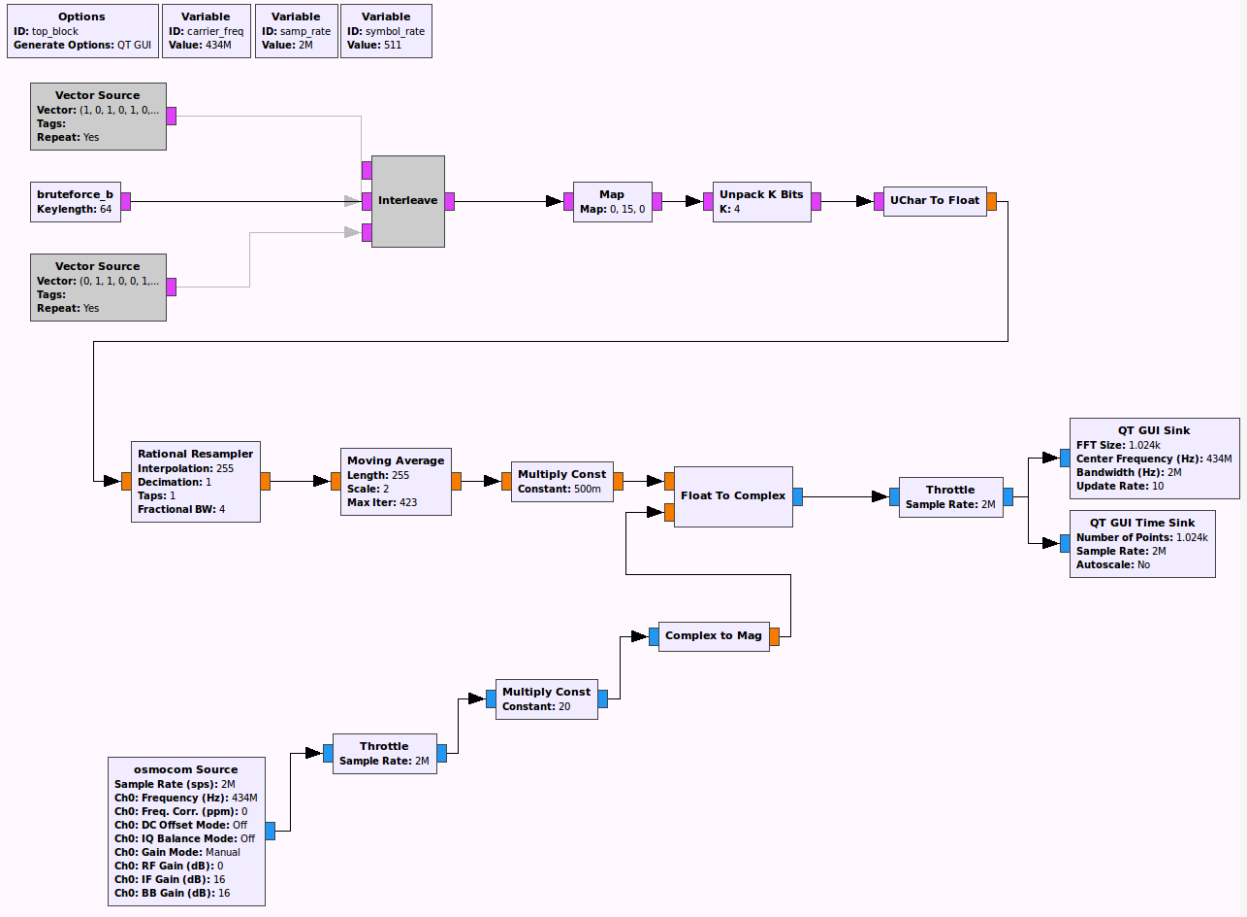

Figure 26: GNU Radio Flowgraph

Παρακάτω αναλύονται ένα προς ένα εκείνα τα block που είναι υπεύθυνα για την δημιουργία του σήματος μαζί με τα χαρακτηριστικά τους:

1. Variables:
    a. Carrier_freq: η μεταβλητή αυτή υποδηλώνει την συχνότητα εκπομπής του σήματος (carrier_freq = 434 MHz).
    b. Sample_rate: η μεταβλητή δήλωσης του ρυθμού δειγματοληψίας (sample_rate = 2MHz).
    c. Symbol_rate: η μεταβλητή που όπως δηλώνει και το όνομά της υποδηλώνει το πλήθος των δειγμάτων που στέλνω ανά σύμβολο. Η τιμή αυτού προέκυψε κατά την διαδικασία ανάλυσης του σήματος μέσω του inspectrum (symbol_rate = 511).

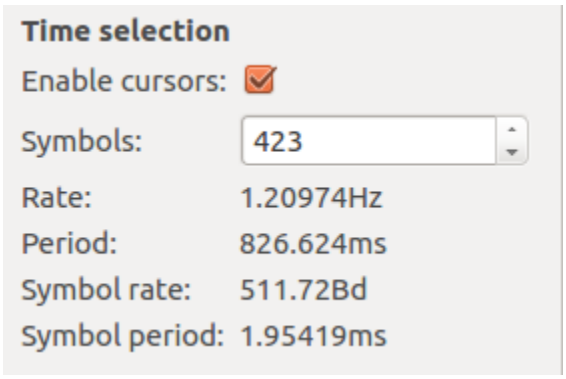

Figure 27: Inspectrum Information About The Signal

2. Sources:
    a. Vector Source(): Χρησιμοποιήσαμε 2 block αυτού του τύπου. Το ένα εξυπηρετεί την αποστολή του preamble, ενώ το δεύτερο την αποστολή των bits του button (λειτουργία των κουμπιών του τηλεχειριστηρίου).
    b. Bruteforce_b(): Το block αυτό αντιπροσωπεύει το δικό μας block.
3. Interleave: το συγκεκριμένο block εξυπηρετεί την συνένωση των τριών πηγαίων block.
4. Map(): Το block αυτό είναι υπεύθυνο για την αποτύπωση των bits. Όταν έχουμε παλμό κατά την μετάδοση, τότε αυτό σημαίνει ότι εκπέμπεται το bit «1», ενώ όταν έχουμε κενό κατά την μετάδοση, τότε αυτό σημαίνει ότι εκπέμπεται το bit «0» (0 → 0x0, 1 → 0xf).
5. Unpack K bits(): όπως υποδηλώνει και το όνομα του block, η λειτουργία του είναι να ξεπακετάρει εκείνο το πλήθος του bits που του έχουμε δηλώσει ως παράμετρο.
6. Uchar to Float(): μετατροπέας του τύπου δεδομένων από byte σε float. Εξυπηρετεί στην επεξεργασία του σήματος από τα επόμενα block.
7. Rational Reasembler(): επανασυναρμολόγηση του σήματος (Interpolation = symbol_rate/2).
8. Moving Average(): για τον έλεγχο των δεδομένων αποστολής (Length = symbol_rate/2, Scale = 2).
9. Multiply Const(): καθορίζει το πλάτος/ένταση του σήματος (Constant = 0.5).
10. Float to Complex(): μετατροπέας του τύπου δεδομένων από float σε complex (διακριτό).
11. Throttle(): απαραίτητο block για τον έλεγχο των δειγμάτων που στέλνονται μέσα από την κεραία. Χρησιμοποιείται κυρίως για την προστασία του επεξεργαστή του υπολογιστή (Sample_rate = sample_rate = 2 MHz).
12. QT GUI Sink(): block υπεύθυνο για την απεικόνιση των συχνοτήτων.
13. QT GUI Time Sink(): block υπεύθυνο για την απεικόνιση του σήματος μέσα στο διάστημα του χρόνου.

Παράλληλα όμως, για να βεβαιωθούμε ότι το σήμα που στέλνουμε ταιριάζει και είναι πανομοιότυπο με αυτό που στέλνεται από το τηλεχειριστήριο, προσθέσαμε ακόμα 4 block. Το πρώτο όπως φαίνεται αποτελεί την πηγή του σήματος, το οποίο του υποδεικνύουμε να το λαμβάνει μέσω του osmocom, εργαλείο που υποστηρίζει το HackRF One. Θέσαμε τις βασικές τιμές, όπως ρυθμό δειγματοληψίας και την συχνότητα που περιμένει να λάβει το σήμα. Στην συνέχεια αυτό περνάει μέσα από ένα throttle για τον έλεγχο της ροής των δεδομένων, ακολουθούμενο από ένα πολλαπλασιαστή πλάτους/έντασης σήματος και τέλος από έναν μετατροπέα τύπου δεδομένων.

Το αποτέλεσμα λοιπόν του γράφου αποτυπώνεται στο παρακάτω σχήμα. Το σήμα που κατασκευάσαμε εμείς εκ νέου προβάλλεται με χρώμα μπλε, ενώ το σήμα που εκπέμπεται από το τηλεχειριστήριο με χρώμα κόκκινο. Οι ελάχιστες διαφορές που έχουν οφείλονται στην ύπαρξη και μόνο του θορύβου. Αφού το σήμα μας παράγεται από την αρχή χωρίς να παρεμβάλλεται κάποια πηγή θορύβου κατά την διάρκεια αποστολής του, φαίνεται να είναι καθαρό. Σε αντίθεση με το σήμα του τηλεχειριστηρίου, όπου έχει να αντιμετωπίσει την εκπομπή μέσω του αέρα αλλά και την ανάμειξή του με διαφορετικά σήματα που ενδέχεται να εκπέμπονται την ίδια χρονική στιγμή.

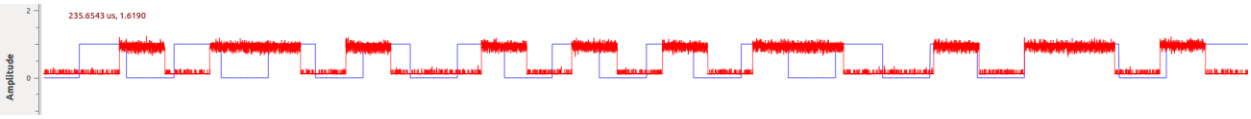

Figure 28: Comparing the Actual Signal with the our Signal

Κεφάλαιο 5: Conclusion – Countermeasures

Τα δεδομένα και η συνδεσιμότητα που καθιστούν δυνατή την είσοδο στο όχημα χωρίς την χρήση κλειδιού, αποτελούν τη βάση για την άμβλυνση της αμέριστης απειλής. Τα οχήματα παράγουν δεδομένα σχετικά με το ασύρματο σύστημα εισόδου τους. Για παράδειγμα, τα δεδομένα του οχήματος μπορούν να υποδηλώνουν συστηματικές συμπεριφορές, όπως μια πόρτα που ανοίγει ή όταν ο κινητήρας αρχίζει όταν αισθάνεται την παρουσία του ασύρματου κλειδιού fob. Παρακολουθώντας τα δεδομένα από ένα συνδεδεμένο αυτοκίνητο χρησιμοποιώντας τις σωστές γνώσεις, μπορεί κάποιος να εντοπίσει αυτές τις συμπεριφορές. Οι επιθέσεις ασύρματης εισόδου μπορούν δυνητικά να ανιχνευθούν με την παρακολούθηση των δεδομένων των οχημάτων και την εκμάθηση της συμπεριφοράς τους, προκειμένου να ανιχνευθούν ανωμαλίες που θα μπορούσαν να υποδηλώνουν την ύπαρξη μιας κακόβουλης κίνησης στο ασύρματο σύστημα.

Μια λύση στον κυβερνοχώρο για τα αυτοκίνητα μπορεί να εντοπίσει αυτή την ανωμαλία και να αναλάβει δράση. Μπορεί να είναι κατάλληλη μια σειρά από απαντήσεις σε περιστατικά, ανάλογα με τον τύπο ασύρματης προσπέλασης εισόδου. Σε μια επίθεση αναμετάδοσης (replay attack), ένα δυνητικό αντίμετρο θα μπορούσε να λάβει χώρα κατά το οποίο μπορεί να ειδοποιείται ο κατασκευαστής του οχήματος ή ο Πάροχος Υπηρεσιών Τηλεματικής (TSP) για την ύποπτη δραστηριότητα. Οι ιδιοκτήτες των οχημάτων με την σειρά τους να μπορούν ειδοποιηθούν μέσω μηνύματος κειμένου ή μέσω εφαρμογής. Με αυτόν τον τρόπο, εάν οι ιδιοκτήτες αυτοκινήτων γνωρίζουν το όχημα τους έχει παραβιαστεί, μπορούν να καλέσουν την αστυνομία ή ακόμα και να ενεργοποιήσουν μια διαδικασία που θα απενεργοποιεί τον κινητήρα αναγκάζοντας τον εισβολέα να εγκαταλείψει τελικά το όχημα.

Μια άλλη κατά των επιθέσεων της εξαντλητικής μεθόδου (brute force attack) θα μπορούσε να αποτελεί η προσθήκη στη πλευρά του συστήματος του δέκτη ενός μηχανισμού που να χρησιμοποιεί παρόμοιο μηχανισμό με αυτόν ενός Anti DDOS συστήματος εντοπισμού και αποτροπής (όπως των Firewall) λήψης σημάτων μετά από ένα εύλογο πλήθος προσπαθειών.

Υπάρχουν καταγεγραμμένα πλήθος ακόμα αντιμέτρων στην προσπάθεια των κατόχων να προστατεύσουν τα οχήματά τους, χωρίς αυτά να αποτελούν επίσημη πρόταση από το κλάδο των αυτοκινητοβιομηχανιών.

Κεφάλαιο 6: References

[1] Remote Keyless System, https://en.wikipedia.org/wiki/Remote_keyless_system
[2] Rolling Codes, https://en.wikipedia.org/wiki/Rolling_code
[3] Modulation, https://en.wikipedia.org/wiki/Modulation
[4] Amplitude Shift Keying, https://en.wikipedia.org/wiki/Amplitude-shift_keying
[5] Frequency Shift Keying, https://en.wikipedia.org/wiki/Frequency-shift_keying
[6] On – Off Keying, https://en.wikipedia.org/wiki/On%E2%80%93off_keying
[7] HackRF One, https://greatscottgadgets.com/hackrf/one/
[8] Audi Key Fob, https://www.autoremotecontrols.com/audi-a3-flip-remote-key
[9] GQRX, http://gqrx.dk/
[10] Inspectrum, https://github.com/miek/inspectrum
[11] GNU Radio, https://www.gnuradio.org/about/
[12] GNU Radio blocks, https://www.gnuradio.org/doc/sphinx-3.7.0/blocks/index.html
[13] GNU Radio coding guide, https://wiki.gnuradio.org/index.php/BlocksCodingGuide
[14] GNU Radio Out of Tree Modules, https://wiki.gnuradio.org/index.php/OutOfTreeModules
[15] GNU Radio building blocks, https://wiki.gnuradio.org/index.php/GNURadioCompanion
[16] Basic Signal Decoding and Replay, Mike Czumak, September 1 2017, https://www.securitysift.com/ook-signal-decoding-replay/
[17] Software Defined Radio (SDR) and Decoding On-off Keying (OOK), Cyrill Brunschwiler, September 1 2016, https://blog.compass-security.com/2016/09/software-defied-radio-sdr-and-decoding-on-off-keying-ook/
[18] HackRF Replay Attack Jeep, https://calebmadrigal.com/hackrf-replay-attack-jeep/
[19] Hacking Car Key Fobs with SDR, Luciano Ferrari, October 22 2018, https://www.lufsec.com/hacking-car-key-fobs-with-sdr/
[20] Jam and Replay Attacks on Vehicular Keyless Entry Systems, https://s34s0n.github.io/2019/07/18/Jam-and-Replay-Attacks-on-Vehicular-Keyless-Entry-Systems/
[21] The Car Hacker's Handbook: A Guide for the Penetration Tester - Craig Smith (2016), Chapter 12. ATTACKING WIRELESS SYSTEMS WITH SDR, https://publicism.info/engineering/penetration/13.html
[22] Jam Intercept and Replay Attack against Rolling Code Key Fob Entry Systems using RTL-SDR, http://spencerwhyte.blogspot.com/2014/03/delay-attack-jam-intercept-and-replay.html
[23] Reverse engineering static key remotes with gnuradio and rfcat, https://leonjza.github.io/blog/2016/10/02/reverse-engineering-static-key-remotes-with-gnuradio-and-rfcat/
[24] On-Off Keying (ASK) with SDR, https://zeta-two.com/radio/2015/06/23/ook-ask-sdr.html
[25] From RF to bytes with an RTL-SDR, https://nada-labs.net/2017/rf-to-bytes-rtl-sdr/
[26] Numpy, https://numpy.org/
[27] Python Coding, https://www.w3schools.com/Python/default.asp
[28] Aram Verstegen, Press to unlock: Analysis, reverse-engineering and implementation of HITAG2-based Remote Keyless Entry systems (August 2018)
[29] Omar Adel Ibrahim, Ahmed Mohamed Hussain, Gabriele Oligeri, and Roberto Di Pietro, Key Is In The Air: Hacking Remote Keyless Entry Systems
[30] Flavio D. Garcia and David Oswald, Lock It and Still Lose It – On the (In)Security of Automotive Remote Keyless Entry Systems
[31] Flavio Garcia, Automotive Cyber Security: Lessons Learned and Research Challenges

[32] Robert Leale, RKE (Key Fob) Attack Using Roll Jam Technique
[33] THE HOBBYIST'S GUIDE TO RTL-SDR: REALLY CHEAP SOFTWARE DEFINED RADIO A GUIDE TO RTL-SDR AND CHEAP SOFTWARE DEFINED RADIO BY THE AUTHORS OF THE RTL-SDR.COM BLOG