Τ.Ε.Ι. ΚΡΗΤΗΣ

Σχολή Εφαρμοσμένων Επιστημών

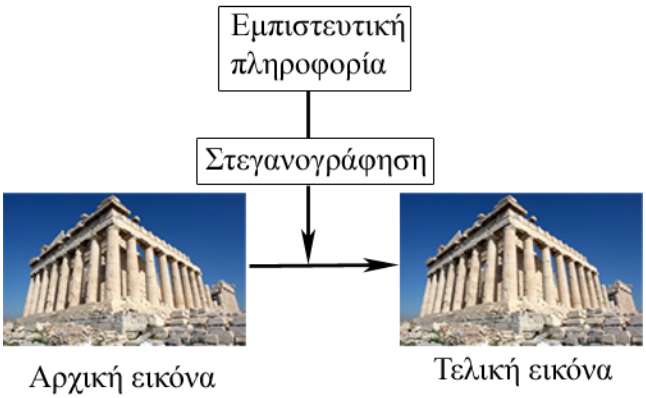
Τμήμα Ηλεκτρονικών Μηχανικών

Τεχνικές στεγανογραφίας και υλοποίηση εφαρμογής σε C#.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

του

Φουρναράκη Κωνσταντίνου



Επιβλέπων: **Δρ. Μηχ. Νικόλαος Σ. Πετράκης**
Καθηγητής Εφαρμογών

Χανιά,

Οκτώβριος 2016

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
Σχολή Εφαρμοσμένων Επιστημών
Τμήμα Ηλεκτρονικών Μηχανικών

ΠΑΡΟΥΣΙΑΣΗ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

με θέμα:

Τεχνικές στεγανογραφίας και υλοποίηση εφαρμογής σε C#.

Φουρναράκη Κωνσταντίνου

Τετάρτη 26 Οκτωβρίου 2016, ώρα 14:00, Αίθουσα 5

Εξεταστική Επιτροπή: Καθ. Εφαρμ. Νικόλαος Πετράκης (επιβλέπων)
Αναπλ. Καθ. Γεώργιος Φουσκιτάκης
Αναπλ. Καθ. Αντώνιος Κωνσταντάρας

Εν' συντομία

Στόχος της παρούσας πτυχιακής εργασίας είναι η μελέτη ικανού αριθμού τεχνικών στεγανογραφίας καθώς και η ανάπτυξη ενός εύχρηστου λογισμικού για στεγανογράφηση/αποστεγανογράφηση. Το λογισμικό αυτό γράφτηκε σε C#, τρέχει στα Windows και έχει την δυνατότητα να ενσωματώνει μέσα σε μια εικόνα πληροφορίες όπως, απλό κείμενο ή κάποιο αρχείο, χωρίς να γίνεται αντιληπτή από το ανθρώπινο μάτι αυτή η αλλαγή στην εικόνα. Στον χρήστη δίνονται αρκετές επιλογές κατά την στεγανογράφηση με σημαντικότερη την επιλογή της κρυπτογράφησης της πληροφορίας πριν την στεγανογράφηση. Η ανάκτηση της κρυμμένης πληροφορίας μπορεί να γίνει από τον παραλήπτη της εικόνας χρησιμοποιώντας το ίδιο λογισμικό από την στιγμή που γνωρίζει τόσο τις απαραίτητες ρυθμίσεις στεγανογράφησης όσο και τους κωδικούς πρόσβασης, στην περίπτωση που εφαρμόστηκε κρυπτογραφία.

Πίνακας Περιεχομένων

Περίληψη

Σκοπός της παρούσας πτυχιακής εργασίας είναι η μελέτη διαφόρων τεχνικών στεγανογραφίας και η υλοποίηση μιας εφαρμογής, σε γλώσσα C#, που να εφαρμόζει την στεγανογραφία σε συνδυασμό με την κρυπτογραφία. Η εφαρμογή είναι σχεδιασμένη με κύριο γνώμονα την ασφάλεια στην ανταλλαγή πληροφορίας ενώ παράλληλα προσφέρει ένα φιλικό προς τον χρήστη γραφικό περιβάλλον πλήρως παραμετροποιημένο. Με την εφαρμογή αυτή ο χρήστης έχει την δυνατότητα να κρύψει πληροφορία (απλό κείμενο ή κάποιο αρχείο) με ασφάλεια μέσα σε μια εικόνα της επιλογής του και στην συνέχεια να την στείλει στον επιθυμητό παραλήπτη. Ο παραλήπτης, για να ανακτήσει την κρυμμένη πληροφορία από την εικόνα, πρέπει να έχει στην κατοχή του το συγκεκριμένο λογισμικό και βέβαια τους κατάλληλους κωδικούς πρόσβασης.

Abstract

The aim of this final work is to study various steganography techniques, as well as, the software implementation in C# language, for use in steganography in conjunction with cryptography. The software is designed with main focus on security in information exchange while offering a user-friendly graphical interface fully parameterized. With this software the user has the ability to hide information (brut text or even a file) securely within an image of his choice and then send it to the desired recipient. The recipient, in order to recover the hidden information of the image, must be in possession of that software and the appropriate passwords.

1. Εισαγωγή

Η εποχή που διανύουμε έχει γνωρίσει τεράστια εξέλιξη στον τομέα της τεχνολογίας με αποκορύφωμα τον τομέα των τηλεπικοινωνιών. Καθημερινά ανταλλάσσονται δεδομένα ανάμεσα στους χρήστες του Διαδικτύου με ταχύτητες που αυξάνονται κάθε μήνα. Οι χρήστες του Διαδικτύου θα ήταν πρακτικά αβοήθητοι εάν δεν υπήρχε η κρυπτογραφία για να τους προστατέψει από τα χέρια τρίτων κρυπτογραφώντας τα δεδομένα που ανταλλάσουν. Ταυτόχρονα, η κρυπτογραφία διασφαλίζει την ακεραιότητα της πληροφορίας που ανταλλάσσεται ακόμα και μετά την αποκρυπτογράφηση τους.

Όμως, όταν ένας χρήστης του Διαδικτύου επιθυμεί να αποστείλει πληροφορία που δεν θέλει να φανεί ότι έχει σταλεί, η κρυπτογραφία δεν μπορεί να βοηθήσει. Αυτήν την ανάγκη κλήθηκε να καλύψει η στεγανογραφία. Στεγανογραφία είναι η τέχνη και η επιστήμη εισαγωγής δεδομένων σε διάφορα μέσα με σκοπό την απόκρυψή τους. Όταν η στεγανογραφία συνδυαστεί με την κρυπτογραφία τότε διασφαλίζεται η εμπιστευτικότητα των δεδομένων που έχουν κρυφθεί. Στην παρούσα πτυχιακή εργασία υλοποιήθηκε εφαρμογή η οποία συνδυάζει στεγανογραφία (δεδομένων σε εικόνα) και κρυπτογραφία (AES) για την κάλυψη ύψιστης ανάγκης ασφάλειας. Ο χρήστης της εφαρμογής έχει την δυνατότητα να προστατέψει (κρυπτογραφήσει) τα δεδομένα του και εν συνεχεία να τα κρύψει μέσα σε μία εικόνα. Η κρυπτογραφημένη αυτή εικόνα, μπορεί να αποσταλεί σε οποιονδήποτε, παραμένει όμως μια απλή εικόνα για όποιον δεν έχει στην κατοχή του την παρούσα εφαρμογή σε συνδυασμό με τον σωστό κωδικό πρόσβασης. Οποιοσδήποτε τρίτος αποκτήσει πρόσβαση στην εικόνα αυτή δεν θα είναι σε θέση να προσπελάσει τα κρυμμένα-κρυπτογραφημένα δεδομένα χωρίς τα κατάλληλα εργαλεία.

Στα κεφάλαια που ακολουθούν ο αναγνώστης της παρούσας εργασίας έχει την ευκαιρία να διευρύνει τις γνώσεις του για:

- Τις τεχνικές που χρησιμοποιούνται στην στεγανογραφία.
- Τις διάφορες μορφές και τον σκοπό που έχει η κρυπτογραφία.
- Την σχεδίαση μιας εφαρμογής που συνδυάζει τα δυο παραπάνω.
- Τον χειρισμό της υλοποιημένης εφαρμογής.

Πιο συγκεκριμένα, το παρόν κεφάλαιο αποτελεί μια σύντομη εισαγωγή της πτυχιακής εργασίας με βασικές και συγκεκριμένες αναφορές στα περιεχόμενα των υπόλοιπων κεφαλαίων. Στο κεφάλαιο 2 παρουσιάζονται και αναλύονται οι σημαντικότερες τεχνικές στεγανογραφίας για την απόκρυψη δεδομένων σε μέσα όπως εικόνα, βίντεο, ήχο, κείμενο, αρχείο και δίκτυο. Στο κεφάλαιο 3 μαθαίνουμε τι είναι ένας αλγόριθμος κρυπτογράφησης, από πού ξεκίνησε, πως είναι οι νεότεροι από αυτούς, σε τι κατηγορίες διαχωρίζονται και με ποια κριτήρια. Στο κεφάλαιο 4 παρουσιάζονται και εξηγούνται τα σημαντικότερα κομμάτια κώδικα που χρησιμοποιήθηκαν για την υλοποίηση της

εφαρμογής. Το κεφάλαιο 5 αποτελεί ένα εγχειρίδιο χρήσης της εφαρμογής ώστε ο χρήστης να είναι σε θέση να το λειτουργήσει χωρίς ιδιαίτερο κόπο ή εξειδικευμένες γνώσεις. Τέλος, στο κεφάλαιο 6 αναγράφονται τα συμπεράσματα της εργασίας αυτής ενώ παράλληλα προτείνονται μελλοντικές ενέργειες για την βελτίωση της εφαρμογής.

Η πλήρης μορφή του πηγαίου κώδικα της εφαρμογής βρίσκεται στο CD που κατατέθηκε με την παρούσα πτυχιακή εργασία στο οποίο παροτρύνεται να ανατρέξει όποιος επιθυμεί να μελετήσει την σχεδίαση της εφαρμογής.

Ο λόγος για τον οποίο επέλεξα να ασχοληθώ με αυτό το θέμα είναι η αυξημένη ανάγκη για την προστασία των προσωπικών δεδομένων στις μέρες μας. Η κυρίαρχη άποψη των νέων χρηστών τεχνολογίας σχετικά με το σαθρό επιχείρημα που προβάλλεται συχνά «δεν έχω κάτι να κρύψω οπότε δεν με ενδιαφέρει» αποτέλεσε θεμέλιο λίθο για την ανάπτυξη της παρούσας εφαρμογής που προσφέρει φιλικό προς τον χρήστη περιβάλλον χωρίς να θυσιάζει την ασφάλεια των δεδομένων του. Ένα εύστοχο παράδειγμα για αυτό το θέμα, έδωσε ο διαδικτυακός ακτιβιστής Edward Snowden λέγοντας: «Το να λέει κάποιος ότι δεν χρειάζεται ιδιωτικότητα επειδή δεν έχει κάτι να κρύψει είναι σαν να λέει ότι δεν χρειάζεται ελευθερία του λόγου επειδή δεν έχει κάτι να πει». [1][2][3][21]

2. Τεχνικές Στεγανογραφίας

Με τον όρο στεγανογραφία αναφερόμαστε στην απόκρυψη ενός μηνύματος μέσα σε κάποιο μέσο. Το μέσο αυτό μπορεί να είναι οποιοδήποτε υλικό ή άυλο αντικείμενο. Η στεγανογραφία δεν πρέπει να συγχέεται με την κρυπτογραφία η οποία έχει ως σκοπό την μετατροπή του μηνύματος σε μη κατανοητή μορφή, έτσι ώστε ακόμα και αν γίνει αντιληπτή η ύπαρξη της πληροφορίας να μην μπορεί να διαβαστεί από άτομα που δεν είναι κάτοχοι του κλειδιού αποκρυπτογράφησης. Οι δυο αυτές μέθοδοι μπορούν να συνδυαστούν καθώς έχουμε την δυνατότητα να κρυπτογραφήσουμε ένα μήνυμα και εν συνεχεία να το κρύψουμε μέσα σε ένα μέσο όπως για παράδειγμα μια εικόνα.[13]

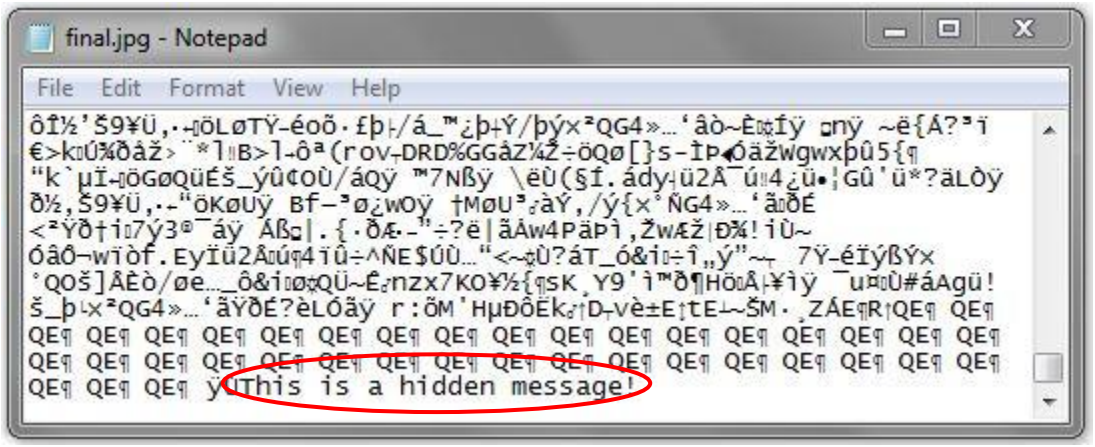
Στην σύγχρονη εποχή που διανύουμε υπάρχει πληθώρα τεχνικών στεγανογραφίας. Οι τομείς που συναντάμε τις πιο γνωστές τεχνικές στεγανογραφίας είναι οι εξής: εικόνας, κειμένου, ήχου, βίντεο, εκτελέσιμων αρχείων, δικτύων (HICCUPS) και εισαγωγής του μηνύματος σε όλων των τύπων αρχεία (injection method).

2.1 Εισαγωγή κειμένου σε αρχείο (Injection method)

Στα λειτουργικά συστήματα Windows έχουμε την δυνατότητα να κρύψουμε ένα αρχείο κειμένου μέσα σε ένα οποιοδήποτε αρχείο (στην παρούσα περίπτωση εικόνα) γράφοντας στην γραμμή εντολών:

>copy C:\cover.jpg/b+ C:\message.txt/b C:\final.jpg

Όπου *cover.jpg* είναι το αρχείο εικόνας μέσα στο οποίο θα κρύψουμε το *message.txt* που είναι ένα απλό αρχείο κειμένου. Το *final.jpg* είναι η εικόνα-συνδυασμός αυτών των δύο. Στην συνέχεια για να ανασύρουμε το κρυμμένο κείμενο απλώς ανοίγουμε το *final.jpg* με έναν κειμενογράφο, όπως φαίνεται στην Εικόνα 2.1.



Εικόνα 2.1: Κείμενο κρυμμένο σε αρχείο εικόνας.

Ο τρόπος αυτός εκμεταλλεύεται το γεγονός ότι το σύστημα παραβλέπει τυχόν δεδομένα που βρίσκονται μετά από το EOF (End Of File). Αυτό έχει ως αποτέλεσμα η εικόνα (στην παρούσα περίπτωση) να εκτελείται κανονικά και το κρυφό μήνυμα να ανασύρεται με έναν κειμενογράφο.[4]

2.2 Εισαγωγή κειμένου σε εικόνα JPEG

Η μορφοποίηση εικόνας JPEG χρησιμοποιεί τον διακριτό μετασχηματισμό συνημίτονου (DCT) (discrete cosine transform) για τον εντοπισμό 64 συντελεστών DCT σε διαδοχικά 8x8 μπλοκ εικονοστοιχείων. Για την εισαγωγή κειμένου μέσα σε μια εικόνα JPEG εκμεταλλευόμαστε την διαδικασία αυτή αλλοιώνοντας τις τιμές των DCT συντελεστών.

2.3 Εισαγωγή κειμένου σε εικόνα (LSB)

Η μέθοδος αυτή χρησιμοποιεί το λιγότερο σημαντικό δυαδικό ψηφίο (LSB = Least Significant Bit) της κάθε ψηφιολέξης (byte) που αποτελεί την εικόνα. Για παράδειγμα έχουμε μια εικόνα 24 μπιτ όπου κάθε εικονοστοιχείο (pixel) αποτελείται από 3 αριθμούς των 8-μπιτ οι οποίοι υποδηλώνουν τις τιμές των χρωμάτων του εικονοστοιχείου (RGB = Red Green Blue). Παρακάτω βλέπουμε τις τιμές τριών διαδοχικών εικονοστοιχείων.

	Κόκκινο	Πράσινο	Μπλε
1ο Εικονοστοιχείο	1001010**1**	0000110**1**	1100100**1**
2ο Εικονοστοιχείο	1001011**0**	0000111**1**	1100101**0**
3ο Εικονοστοιχείο	1001111**1**	0001000**0**	1100101**1**

Οι τιμές των LSB παρουσιάζονται υπερτονισμένες (bold). Έστω ότι θέλουμε να εισάγουμε τον χαρακτήρα 1 (ASCII) που στο δυαδικό έχει την τιμή **00110001**. Εισάγουμε τα δυαδικά ψηφία (bits) του χαρακτήρα αντικαθιστώντας τις ήδη υπάρχουσες τιμές των εικονοστοιχείων (pixel).

	Κόκκινο	Πράσινο	Μπλε
1ο Εικονοστοιχείο	1001010**0**	0000110**0**	1100100**1**
2ο Εικονοστοιχείο	1001011**1**	0000111**0**	1100101**0**
3ο Εικονοστοιχείο	1001111**0**	0001000**1**	1100101**1**

Όπως βλέπουμε στον παραπάνω πίνακα τα λιγότερο σημαντικά δυαδικά ψηφία (LSB) έχουν αντικατασταθεί με τα δυαδικά ψηφία του χαρακτήρα ‘1’ και τυχόν πλεονασμοί μένουν αναλλοίωτοι (υπογραμμισμένο bit).

Υποθετικά σε μια εικόνα διαστάσεων 1024x768 εάν κρύψουμε έναν χαρακτήρα σε κάθε τρία εικονοστοιχεία τότε μπορούμε να κρύψουμε έως και 262.144 χαρακτήρες.

Η μέθοδος αυτή μπορεί να λειτουργήσει μόνο με μορφοποιήσεις εικόνων, των οποίων τα περιεχόμενα είναι οι τιμές των χρωμάτων στα εικονοστοιχεία (π.χ. PNG, BMP). Αντιθέτως τα περιεχόμενα μιας εικόνας JPEG αποτελούν αριθμούς συμπίεσης και γι’ αυτό δεν υποστηρίζεται η παραπάνω μέθοδος. [14]

2.4 Στεγανογραφία φθαρμένων δικτύων (HICCUPS)

Η μέθοδος στεγανογραφίας δικτύων HICCUPS (Hidden Communication System for Corrupted Networks) παρουσιάστηκε από τον Krzysztof Szczypiorski το 2003 [5]. Με την μέθοδο αυτή υπάρχει η δυνατότητα μετάδοσης πληροφοριών ανάμεσα σε μια ομάδα ηλεκτρονικών υπολογιστών με πακέτα του πρωτοκόλλου TCP/IP τα οποία έχουν εσκεμμένα λάθος άθροισμα ελέγχου αρχείου (file checksum). Στην εικόνα 2.2 βλέπουμε μια τοπολογία στην οποία μια κρυφή ομάδα χρηστών ανταλλάσουν κατακερματισμένα πακέτα με λάθος άθροισμα ελέγχου αρχείου.

Εικόνα 2.2: Τοπολογία HICCUPS.

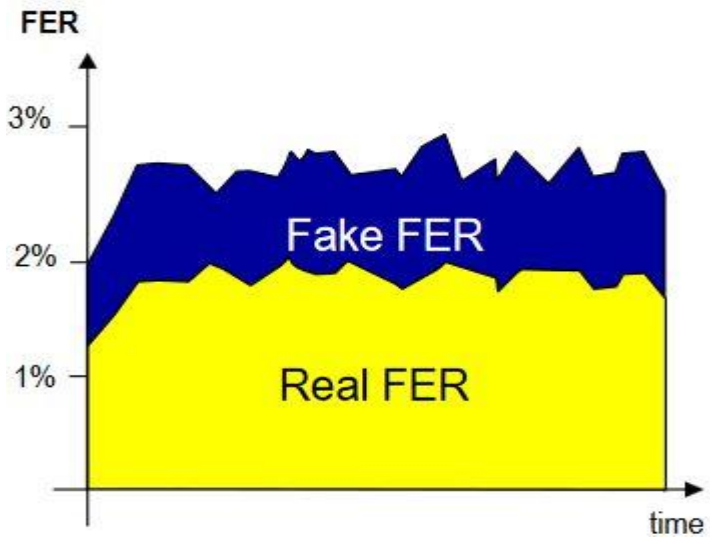
Η μέθοδος αυτή έχει ως επακόλουθο την αύξηση του ποσοστού σφάλματος πλαισίου FER (Frame Error Rate) καθώς κάνει αναμετάδοση (broadcast) κατεστραμμένων πακέτων (άχρηστα για τους συμβατικούς χρήστες αλλά χρήσιμα για την κρυφή ομάδα). Υποθέτουμε ότι έχουμε ένα ασύρματο δίκτυο IEEE 802.11b με εύρος ζώνης 11 Mbps και πραγματικό ποσοστό σφάλματος πλαισίου 1.5% και ταυτόχρονα μη πραγματικό 2.5%. Εάν χρησιμοποιείται το 40% του εύρους ζώνης τότε το ωφέλιμο εύρος ζώνης στεγανογραφικής επικοινωνίας θα είναι:

11 Mbps*40%*(2.5%-1.5%) = 44 kbps

Ομοίως για ένα δίκτυο 54 Mbps IEEE 802.11g το ωφέλιμο εύρος ζώνης στεγανογραφικής επικοινωνίας θα είναι:

54 Mbps*40%*(2.5%-1.5%) = 216 kbps

Στην εικόνα 2.3 παρουσιάζεται με κίτρινο χρώμα το πραγματικό ποσοστό σφάλματος πλαισίου και με μπλε το μη πραγματικό (προϊόν μεθόδου HICCUPS).



Εικόνα 2.3: Ποσοστά σφάλματος πλαισίου.

2.5 Στεγανογραφία ήχου

2.5.1 Μέθοδος λιγότερου σημαντικού δυαδικού ψηφίου (LSB coding)

Μια από τις πιο διαδεδομένες μεθόδους στην στεγανογραφία ήχου είναι αυτή του λιγότερου σημαντικού δυαδικού ψηφίου. Η μέθοδος αυτή έχει αρκετές ομοιότητες με την μέθοδο στεγανογραφίας εικόνας (2.3) που αναλύσαμε προηγουμένως. Στην εικόνα 2.4 παρουσιάζεται η στεγανογραφία του χαρακτήρα 'C' σε ένα αρχείο ήχου με ροή δειγματοληψίας 16 δυαδικά ψηφία (bit). Ο χαρακτήρας 'C' του ASCII σε δυαδική μορφή είναι '01000011'. [6]

Αρχείο ήχου με ροή δειγματοληψίας 16 δυαδικά ψηφία	Η δυαδική μορφή του χαρακτήρα 'C' του ASCII	Αρχείο ήχου με στεγανογραφημένο τον χαρακτήρα 'C' του ASCII
0110001011001101	0	0110001011001100
1100111000101010	1	1100111000101011
1010100101010101	0	1010100101010100
1010101010100110	0	1010101010100110
0101010101010101	0	0101010101010100
1010101010100101	0	1010101010100100
1110001010101000	1	1110001010101001
1010101111010101	1	1010101111010101

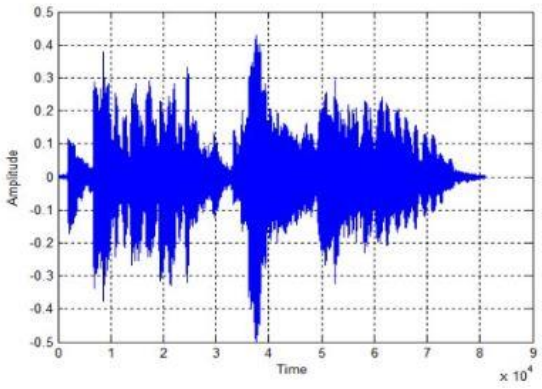
Εικόνα 2.4: Στεγανογραφία ενός ASCII χαρακτήρα με την μέθοδο λιγότερου σημαντικού ψηφίου (LSB)

Παρατηρούμε ότι τα κόκκινα ψηφία του τρίτου πίνακα είναι αυτά του δεύτερου πίνακα και τα υπογραμμισμένα είναι τα ψηφία του πρώτου πίνακα που δεν χρειάστηκε να μεταβληθούν.

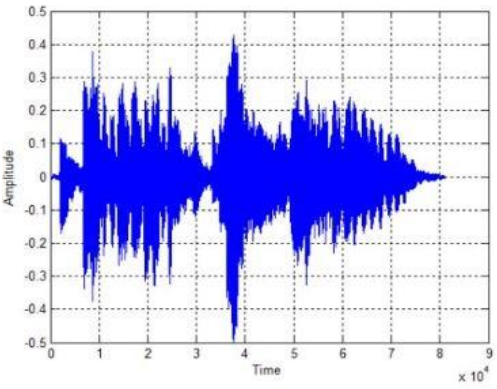
Η μεταβολή των λιγότερο σημαντικών δυαδικών ψηφίων σε ένα αρχείο ήχου δεν είναι αισθητή στην ανθρώπινη ακοή καθώς δεν αλλοιώνουν τις ακουστικές συχνότητες σημαντικά.

2.5.2 Μέθοδος εξάπλωσης φάσματος (Spread Spectrum)

Αυτή η μέθοδος στεγανογραφίας ήχου εξαπλώνει το κρυφό μήνυμα κατά μήκος των ακουστικών συχνοτήτων. Αποτέλεσμα αυτού του τρόπου λειτουργίας είναι το γεγονός ότι η αλλοίωση του αρχείου ήχου γίνεται εύκολα αντιληπτή από οποιονδήποτε το αναπαράγει. Στις εικόνες 2.5 και 2.6 βλέπουμε το αρχικό και το τροποποιημένο σήμα ήχου αντίστοιχα. Η διαφορά των ακουστικών συχνοτήτων είναι εμφανής. [7]



Εικόνα 2.5: Αρχικό σήμα ήχου



Εικόνα 2.6: Τροποποιημένο σήμα ήχου

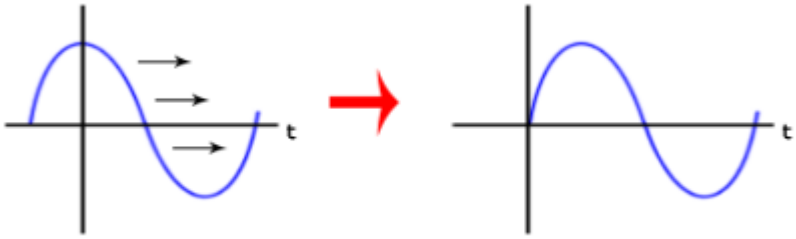
2.5.3 Μέθοδος κωδικοποίησης φάσης (Phase Coding)

Η τεχνική αυτή αντικαθιστά την φάση ενός αρχικού τμήματος του αρχείου ήχου με μια φάση αναφοράς που αντιπροσωπεύει την κρυφή πληροφορία. Η φάση στα υπόλοιπα τμήματα ρυθμίζεται προκειμένου να διατηρηθεί η διαφορά φάσης μεταξύ των τμημάτων. Στην εικόνα 2.7 παρουσιάζεται η μεταβολή φάσης ανάλογα με τα δυαδικά ψηφία (Bit) του κρυφού μηνύματος. [6]

$$\text{Μεταβολή Φάσης} = \begin{cases} \pi/2 \text{ εάν το κρυφό δυαδικό ψηφίο} = 0 \\ -\pi/2 \text{ εάν το κρυφό δυαδικό ψηφίο} = 1 \end{cases}$$

Εικόνα 2.7: Μεταβολή φάσης ανάλογα με το κρυφό δυαδικό ψηφίο

Στην εικόνα 2.8 παρατηρούμε την μεταβολή φάσης σε ένα σήμα ημιτόνου.



Εικόνα 2.8: **Αρχικό σήμα ημιτόνου** **Μετατοπισμένο κατά π/2 σήμα**

Από άποψη αναλογίας σήματος προς θόρυβο η μέθοδος κωδικοποίησης φάσης είναι η πιο αποτελεσματική. Όταν η μεταβολή φάσης είναι χαμηλή τότε μπορεί να επιτευχθεί στεγανογραφία μη αντιληπτή από το ανθρώπινο αυτί σε αντίθεση με άλλες μεθόδους που μεταβάλουν τις ακουστικές συχνότητες.

2.5.4 Μέθοδος κωδικοποίησης ισοτιμίας (Parity Coding)

Η μέθοδος κωδικοποίησης ισοτιμίας διαιρεί το σήμα σε περιοχές δειγμάτων, σε αντίθεση με την μέθοδο λιγότερου σημαντικού ψηφίου που λειτουργεί με μεμονωμένα δείγματα. Στην παρούσα μέθοδο γίνεται χρήση της άρτιας ισοτιμίας (Even Parity) σε κάθε μια από τις περιοχές δειγμάτων. Στο παρακάτω παράδειγμα βλέπουμε πως λειτουργεί η άρτια ισοτιμία για τον έλεγχο επιτυχούς λήψης. [6, 16]

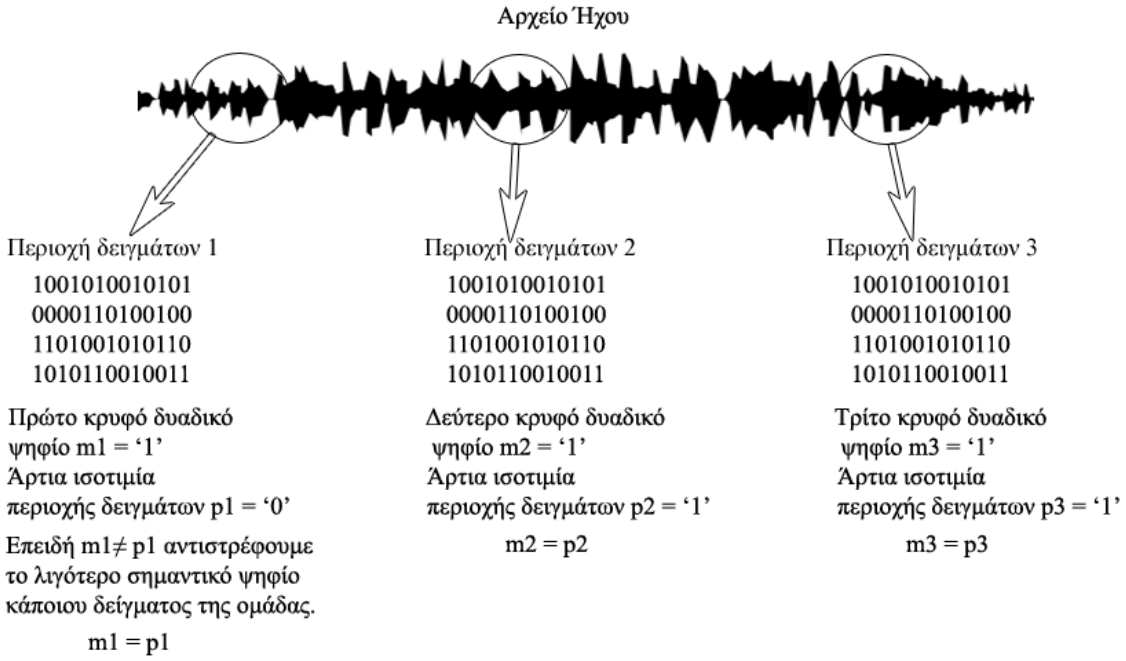
- Ο χρήστης Α θέλει να στείλει τον δυαδικό αριθμό: 1001
- Υπολογίζει την άρτια ισοτιμία του δυαδικού αριθμού: 1+0+0+1 (mod 2) = 0
- Εισάγει το αποτέλεσμα της πράξης (‘0’) στο τέλος του δυαδικού αριθμού ο οποίος γίνεται 10010 και στέλνει αυτό τον αριθμό στον χρήστη Β.

❖ Ο χρήστης Β λαμβάνει τον αριθμό: 10010

❖ Υπολογίζει την άρτια ισοτιμία: 1+0+0+1+0 (mod 2) = 0

❖ Ο Β είναι σίγουρος ότι ο αριθμός που έλαβε είναι αυτός που του έστειλε ο Α καθώς το αποτέλεσμα που υπολόγισε είναι άρτιο.

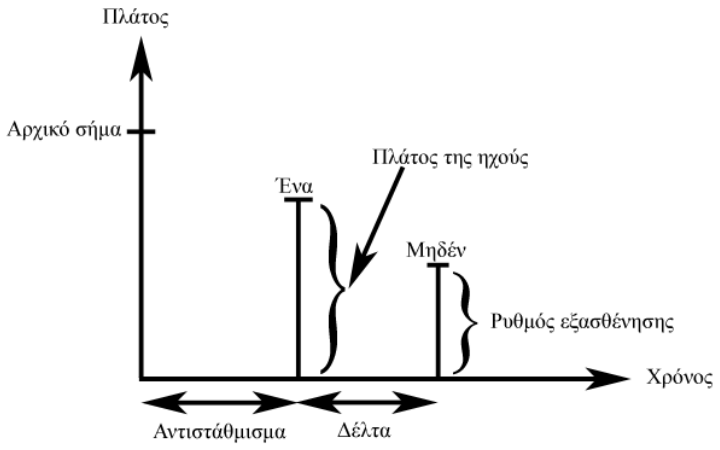
Εάν θέλουμε να κρύψουμε σε ένα αρχείο ήχου τα τρία πρώτα δυαδικά ψηφία ‘111’ του χαρακτήρα ‘Φ’ με αυτή την μέθοδο, θα ακολουθήσουμε τα παρακάτω βήματα όπως φαίνονται στην εικόνα 2.9.



Εικόνα 2.9: Αναπαράσταση στεγανογραφία δυαδικών ψηφίων '111' σε αρχείο ήχου με την μέθοδο κωδικοποίησης ισοτιμίας.

2.5.5 Μέθοδος απόκρυψης στην ηχώ (Echo Hiding).

Με την μέθοδο απόκρυψης στην ηχώ έχουμε την δυνατότητα να κρύψουμε τα δυαδικά ψηφία της πληροφορίας μας μέσα στην ηχώ ενός αρχείου ήχου. Πριν από την έναρξη της διαδικασίας κωδικοποίησης το αρχικό σήμα διαιρείται σε μικρότερα κομμάτια. Μετά το πέρας της διαδικασίας αυτής τα κομμάτια συνδέονται για να παράξουν το τελικό σήμα. Οι μεταβλητές παράμετροι που χαρακτηρίζουν την μέθοδο αυτή παρουσιάζονται παρακάτω στην εικόνα 2.10.



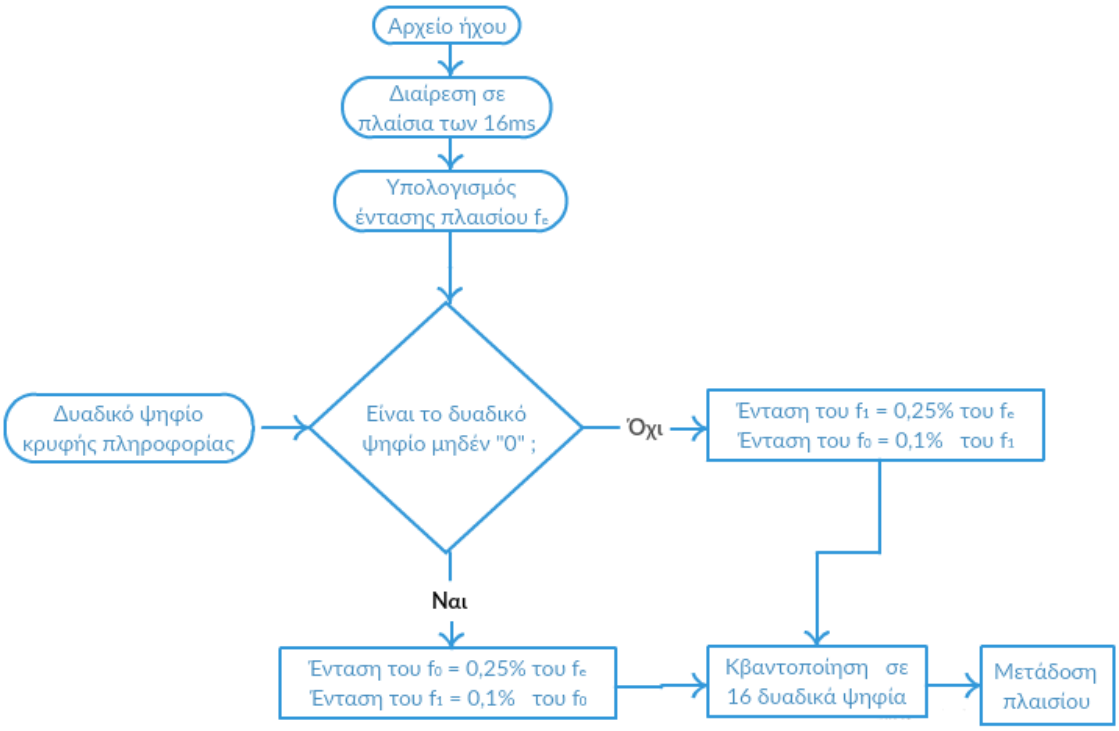
Εικόνα 2.10: Μεταβλητές παράμετροι της μεθόδου απόκρυψης στην ηχώ.

Για την κωδικοποίηση με αυτή την μέθοδο χρησιμοποιούνται δυο χρονοκαθυστερήσεις (delay times), η μια αντιπροσωπεύει τον δυαδικό αριθμό ένα με διάρκεια όσο το αντιστάθμισμα (offset) και η άλλη τον δυαδικό αριθμό μηδέν με διάρκεια όσο το αντιστάθμισμα συν την διάρκεια δέλτα.

Προτέρημα αυτής της μεθόδου είναι το γεγονός ότι δεν αλλοιώνεται το αρχείο ήχου με θόρυβο σε αντίθεση με άλλες μεθόδους. Εάν το αρχείο ήχου έχει μια μόνο ηχώ τότε μπορεί να κρυφτεί μόνο ένα δυαδικό ψηφίο πληροφορίας. Από τον τρόπο λειτουργίας της μεθόδου αυτής απορρέει και το κύριο μειονέκτημά της που είναι η περιορισμένη χωρητικότητα αποθήκευσης της κρυφής πληροφορίας. [8]

2.5.6 Μέθοδος εισαγωγής τόνου (Tone Insertion)

Το ανθρώπινο αυτί δεν έχει την ικανότητα να αντιληφθεί χαμηλής έντασης τόνους με την παρουσία υψηλότερων φασματικών συνιστωσών. Σε αυτό το γεγονός στηρίζεται η μέθοδος εισαγωγής τόνου. Η διαδικασία που ακολουθούμε σε αυτή την μέθοδο απεικονίζεται στην εικόνα 2.11.



Εικόνα 2.11: Διάγραμμα μεθόδου εισαγωγής τόνου.

Αρχικά το αρχείο ήχου διαιρείται σε πλαίσια διάρκειας 16 ms (χιλιοστά του δευτερολέπτου). Ύστερα για την κωδικοποίηση των δυαδικών ψηφίων κρυφής πληροφορίας ‘0’ και ‘1’ χρησιμοποιούμε δυο τόνους με συχνότητες f_0=1875 Hz και f_1=2625 Hz αντίστοιχα. Για κάθε πλαίσιο του αρχείου ήχου υπολογίζεται μια ένταση πλαισίου f_e. Με κάθε υπολογισμό της έντασης πλαισίου f_e εισάγουμε ένα δυαδικό ψηφίο κρυφής πληροφορίας στο αρχείο ήχου. Εάν το δυαδικό ψηφίο που θέλουμε να εισάγουμε είναι ‘0’ τότε η ένταση του τόνου συχνότητας f_0 ορίζεται στο 0,25% της έντασης πλαισίου f_e και η ένταση του f_1 στο 0,1% της έντασης πλαισίου f_0. Για το δυαδικό ψηφίο ‘1’ η ένταση του f_1 γίνεται 0,25% του f_e ενώ η ένταση του f_0 0,1% της f_1. Τα πλαίσια με τις φασματικές συνιστώσες τους στις συχνότητες των τόνων αποτελούν το τελικό αρχείο ήχου που εμπεριέχει την κρυφή πληροφορία. Η ταυτόχρονη ρύθμιση σημαντικών και εξαιρετικά χαμηλών εντάσεων των τόνων διευκολύνει την κρυφή εισαγωγή και την σωστή εξαγωγή δεδομένων. [9]

2.6 Στεγανογραφία βίντεο

Ένα αρχείο βίντεο αποτελείται από έναν αριθμό εικόνων (καρέ) και ήχων που αναπαράγονται ταυτόχρονα. Από αυτή και μόνο την πληροφορία συμπεραίνουμε ότι μπορούν να υποστούν στεγανογραφία οι εικόνες ή ο ήχος ή και τα δύο ταυτόχρονα. Σε αυτή την μέθοδο στεγανογραφίας έχουμε διπλό κέρδος καθώς λόγω των εναλλασσόμενων εικόνων το ανθρώπινο μάτι δεν αντιλαμβάνεται ακόμα και εμφανώς αλλοιωμένες εικόνες (καρέ). Επίσης, λόγω του μεγάλου μεγέθους των αρχείων βίντεο έχουμε άπλετο χώρο για την εισαγωγή κρυφών μηνυμάτων (ή αρχείων).

2.7 Στεγανογραφία κειμένου

Στην στεγανογραφία κειμένου το μέσο στο οποίο κρύβουμε κείμενο είναι ένα κείμενο. Οι κυριότερες μέθοδοι για την επίτευξη αυτού του στόχου περιλαμβάνουν την αλλαγή της μορφοποίησης ενός ήδη υπάρχοντος κειμένου, την αλλαγή των λέξεων στο κείμενο, την παραγωγή τυχαίων ακολουθιών χαρακτήρων ή την χρήση γραμματικής χωρίς συμφραζόμενα για την παραγωγή αναγνώσιμου κειμένου.

2.7.1 Μέθοδος βασισμένη στην μορφοποίηση (Format Based Method)

Η μέθοδος που βασίζεται στην μορφοποίηση κειμένου περιλαμβάνει την φυσική αλλαγή της μορφής του κειμένου για την απόκρυψη της πληροφορίας. Το μειονέκτημα της μεθόδου αυτής είναι ότι εάν το παραγόμενο κείμενο ανοιχτεί με έναν κειμενογράφο είναι πιθανό να ανιχνευτούν ορθογραφικά λάθη και πρόσθετα κενά ενώ το αλλαγμένο μέγεθος χαρακτήρα είναι διακριτό από το ανθρώπινο μάτι. Επιπρόσθετα το παραγόμενο κείμενο μπορεί να συγκριθεί με το αρχικό κείμενο (χωρίς κρυφή πληροφορία) και να εντοπισθούν διαφορές. Τα στοιχεία αυτά είναι ικανά να κινήσουν υποψίες ότι το κείμενο μπορεί να περιέχει κρυμμένη πληροφορία. [10]

Σε αυτή την μέθοδο το κρυφό δυαδικό ψηφίο (“0” ή “1”) κρύβεται με ποικίλους τρόπους. Ένας από αυτούς είναι η ολίσθηση γραμμής (Line Shift) όπου το κρυφό δυαδικό ψηφίο είναι ανάλογο της απόσταση από την μια γραμμή στην άλλη.

Ένας άλλος τρόπος είναι με την ολίσθηση λέξης (Word Shift) όπου εάν η λέξη για παράδειγμα έχει μετατοπιστεί προς τα δεξιά τότε το κρυφό δυαδικό ψηφίο είναι “1” (Εικόνα 2.12) ενώ προς τα αριστερά “0”.

Ένα μήλο.

Ένα μήλο.

Εικόνα 2.12: Ολίσθηση λέξης

Με ανάλογο τρόπο ορίζονται τα κρυφά δυαδικά ψηφία και στην συντακτική μέθοδο (Syntactic Method) στην οποία μπορεί το κόμμα να ορίζει το “1” και η τελεία το “0”. Το

πρόβλημα σε αυτή την μέθοδο έγκειται στο ότι όποιος διαβάσει το παραγόμενο κείμενο θα αντιληφτεί εύκολα την ακατάλληλη χρήση των σημείων στίξης.

Με την βοήθεια του κενού διαστήματος (White Space Steganography) μπορεί να γίνει μια παρόμοια διαδικασία στην οποία το δυαδικό μηδέν αντιπροσωπεύεται από ένα κενό διάστημα πριν την τελεία μιας πρότασης . Το δυαδικό ένα αντιπροσωπεύεται από δύο κενά διαστήματα πριν την τελεία της πρότασης. Την μέθοδο αυτή χρησιμοποιεί το διαδικτυακό εργαλείο spammimic [15] το οποίο κρύβει το μήνυμα της επιλογής μας σε ανεπιθύμητα κείμενα (spam).

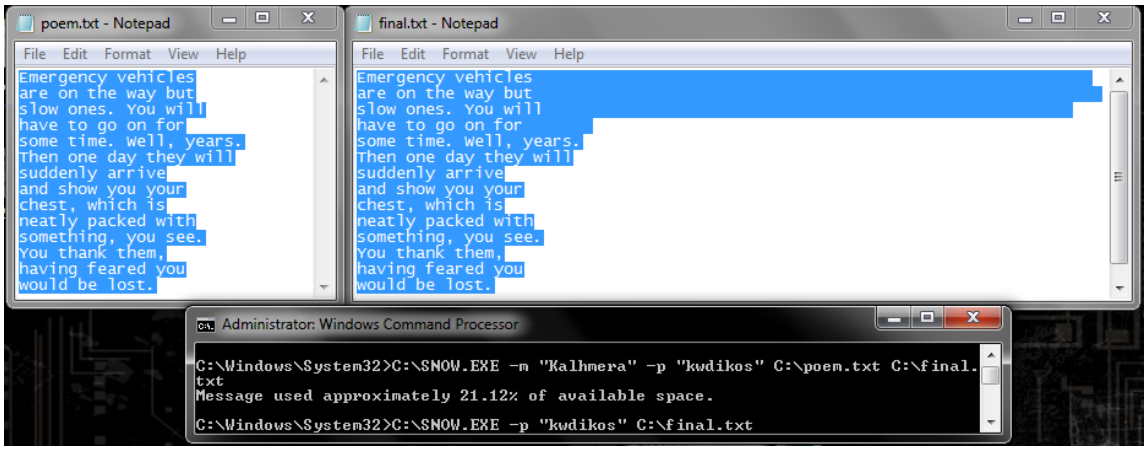
Στο παρακάτω παράδειγμα έχουμε κρύψει το μήνυμα 'Καλημέρα' με την τεχνική στεγανογραφίας κενού διαστήματος του spammimic.

Dear E-Commerce professional ; This letter was specially selected to be sent to you . We will comply with all removal requests . This mail is being sent in compliance with Senate bill 1622 , Title 4 , Section 303 . Do NOT confuse us with Internet scam artists ! Why work for somebody else when you can become rich inside 81 weeks ! Have you ever noticed nearly every commercial on television has a .com on in it and most everyone has a cellphone . Well, now is your chance to capitalize on this . WE will help YOU use credit cards on your website plus decrease perceived waiting time by 150% . You can begin at absolutely no cost to you . But don't believe us ! Mr Anderson of Georgia tried us and says "I was skeptical but it worked for me" ! We assure you that we operate within all applicable laws ! For the sake of your family order now ! Sign up a friend and you get half off ! God Bless ! Dear Friend , Especially for you - this cutting-edge news . We will comply with all removal requests ! This mail is being sent in compliance with Senate bill 2316 ; Title 2 , Section 306 . This is not a get rich scheme ! Why work for somebody else when you can become rich inside 57 months ! Have you ever noticed more people than ever are surfing the web plus people will do almost anything to avoid mailing their bills . Well, now is your chance to capitalize on this ! WE will help YOU deliver goods right to the customer's doorstep & SELL MORE . You can begin at absolutely no cost to you ! But don't believe us . Mrs Jones of New Jersey tried us and says "Now I'm rich, Rich, RICH" ! We assure you that we operate within all applicable laws ! We beseech you - act now ! Sign up a friend and you'll get a discount of 10% ! God Bless .

Παρατηρήστε τα περιττά κενά ανάμεσα στα σημεία στίξης και τις λέξεις.

Μια άλλη πτυχή αυτής της μεθόδου στεγανογραφίας καλύπτει η εφαρμογή Snow [17] η οποία στο τέλος κάθε γραμμής εισάγει κενούς χαρακτήρες με τους οποίους κωδικοποιεί την κρυφή πληροφορία. Στην εικόνα 2.13 παρατηρούμε αριστερά το αρχείο poem.txt που αποτελεί το αρχείο εισόδου, δεξιά το αρχείο final.txt το οποίο είναι το τελικό αρχείο κειμένου με το μήνυμα "Kalhmera" κρυμμένο στα κενά διαστήματα στο τέλος των τεσσάρων πρώτων γραμμών όπως φαίνεται. Το τρίτο παράθυρο είναι ένα τερματικό με τις εντολές που χρειάστηκε να εκτελεστούν. Αξίζει να σημειωθεί ότι αυτή η εφαρμογή σε αντίθεση με το spammimic προηγουμένως , μας δίνει την δυνατότητα να

κρυπτογραφήσουμε το μήνυμά μας με έναν κωδικό της επιλογής μας που στην παρούσα περίπτωση είναι το "kwdikos".



Εικόνα 2.13: Στεγανογραφία μηνύματος σε κείμενο με την εφαρμογή Snow.

Αρκετό ενδιαφέρον παρουσιάζει και η μέθοδος ετικέτας γλώσσας σήμανσης υπερκειμένου (HTML Tag) [11] στην οποία τα κρυφά δυαδικά ψηφία απεικονίζονται όπως φαίνεται παρακάτω.

Για την κωδικοποίηση του δυαδικού "0"

<img src=image.jpg></img>

Για την κωδικοποίηση του δυαδικού "1"

**

Για την κωδικοποίηση της δυαδικής ακολουθίας "010110"

<img src=image1.jpg></img>

**

<img src=image3.jpg></img>

**

**

<img src=image6.jpg></img>

Οι αλλαγές αυτές δεν έχουν κανένα αντίκτυπο στην εκτέλεση του κώδικα και η παραγόμενη ιστοσελίδα σε σχέση με την αρχική είναι φαινομενικά ταυτόσημη.

Αξίζει να αναφερθεί και η μέθοδος στεγανογραφίας σε αριθμητικούς πίνακες αποτελεσμάτων (Cricket Match Scorecard) [12]. Σχετικά απλή μέθοδος στην οποία τα κρυφά δυαδικά ψηφία κρύβονται μπροστά από τους αριθμούς του πίνακα με την βοήθεια του μηδέν. Πιο συγκεκριμένα εάν το κρυφό δυαδικό ψηφίο είναι μηδέν "0" τότε ο αριθμός μέσα στον τελικό πίνακα θα είναι ο ίδιος με τον αριθμό στον αρχικό πίνακα ενώ αν το ψηφίο είναι "1" τότε εισάγουμε ένα μηδέν μπροστά από τον αριθμό του τελικού πίνακα. Στους παρακάτω πίνακες φαίνεται ένα παράδειγμα.

Παίκτης No1	8
Παίκτης No2	6
Παίκτης No3	2

Πίνακας 1: Αρχικός πίνακας αποτελεσμάτων

Παίκτης No1	08
Παίκτης No2	6
Παίκτης No3	02

Πίνακας 2: Τελικός πίνακας με κρυμμένη την δυαδική ακολουθία "101".

2.7.2 Μέθοδος τυχαίας και στατιστικής παραγωγής (Random and Statistical Generation)

Σε αυτή την μέθοδο εξαλείφεται ο κίνδυνος σύγκρισης του παραγόμενου με το αρχικό κείμενο. Αυτό συμβαίνει γιατί δεν χρησιμοποιούμε κάποιο ήδη υπάρχον κείμενο για την απόκρυψη της πληροφορίας αλλά παράγουμε τυχαία ένα. Η παραγωγή του αρχικού κειμένου μπορεί να γίνει είτε με τυχαίες ακολουθίες χαρακτήρων, είτε με ήδη υπάρχουσες λέξεις με στατιστικές ιδιότητες (π.χ. λέξεις με ομοιοκαταληξία για την παραγωγή ενός κείμενου που θα μοιάζει με ποίημα).

2.7.3 Γλωσσική στεγανογραφία (Linguistic Steganography)

Η γλωσσική στεγανογραφία χρησιμοποιεί τις γλωσσικές ιδιότητες του κειμένου και εισάγει την κρυφή πληροφορία στην γλωσσική δομή του κειμένου. Παραδείγματος χάριν οι λέξεις «αμείβω» και «πληρώνω» έχουν την ίδια έννοια οπότε η μία μπορεί να χρησιμοποιηθεί για να αντιπροσωπεύει το ΄1΄ και η άλλη το ΄0΄ της κρυφής πληροφορίας. Το παραγόμενο κείμενο είναι συντακτικά αψεγάδιαστο αλλά υπάρχει η πιθανότητα να είναι σημασιολογικά ακατανόητο πράγμα που αποτελεί και το μοναδικό μειονέκτημα της μεθόδου αυτής.

3 Κρυπτογραφία

3.1 Ορισμός

Η λέξη κρυπτογραφία (cryptography) προέρχεται από τα συνθετικά «κρυπτός» και «γραφή». Είναι ο επιστημονικός κλάδος που ασχολείται με τη μελέτη, την ανάπτυξη και τη χρήση τεχνικών κρυπτογράφησης και αποκρυπτογράφησης, με σκοπό τη διαφύλαξη του απορρήτου των πληροφοριών που ανταλλάσσονται. Η κρυπτογραφία μαζί με την κρυπτανάλυση αποτελούν τους δύο κλάδους της κρυπτολογίας, η οποία ασχολείται με τη μελέτη της ασφαλούς επικοινωνίας. Η σημασία της κρυπτογραφίας είναι τεράστια στους τομείς της ασφάλειας υπολογιστικών συστημάτων και τηλεπικοινωνιών. Ο κύριος στόχος της είναι να παρέχει μηχανισμούς ώστε να ανταλλάσσονται με ασφάλεια δεδομένα μεταξύ αποστολέα και παραλήπτη χωρίς να έχει πρόσβαση σε αυτά κάποιος τρίτος. Ιστορικά, η κρυπτογραφία χρησιμοποιήθηκε για τη μετατροπή της πληροφορίας των μηνυμάτων από μια κατανοητή μορφή σε έναν «γρίφο», που χωρίς τη γνώση του κρυφού μετασχηματισμού η πληροφορία θα παρέμενε ακατανόητη. [1, 2, 3, 15]

3.2 Ιστορική αναδρομή

Η πρώτη στρατιωτική χρήση της κρυπτογραφίας αποδίδεται στους Σπαρτιάτες. Γύρω στον 5^{ο} π.Χ. αιώνα εφηύραν την «σκυτάλη», την πρώτη κρυπτογραφική συσκευή, στην οποία χρησιμοποίησαν για την κρυπτογράφηση τη μέθοδο της μετάθεσης. Όπως αναφέρει ο Πλούταρχος, η «Σπαρτιατική Σκυτάλη» (Εικόνα 3.1), ήταν μια ξύλινη ράβδος, ορισμένης διαμέτρου, γύρω από την οποία ήταν τυλιγμένη ελικοειδώς μια λωρίδα περγαμηνής. Το κείμενο ήταν γραμμένο σε στήλες, ένα γράμμα σε κάθε έλικα, όταν δε ξετύλιγαν τη λωρίδα, το κείμενο ήταν ακατάληπτο εξαιτίας της αναδιάταξης των γραμμάτων. Το «κλειδί» ήταν η διάμετρος της σκυτάλης.

Εικόνα 3.1: Η Σπαρτιατική Σκυτάλη, μια πρώιμη συσκευή κρυπτογράφησης

Από τις αρχές του 20^{ου} αιώνα εμφανίστηκε ραγδαία ανάπτυξη στον τομέα της κρυπτογραφίας. Αυτό ήταν αποτέλεσμα των δύο παγκοσμίων πολέμων καθώς υπήρξε εξαιρετικά μεγάλη ανάγκη για ασφάλεια κατά τη μετάδοση ζωτικών πληροφοριών μεταξύ των στρατευμάτων των χωρών. Τα κρυπτοσυστήματα αυτής της περιόδου αρχίζουν να γίνονται πολύπλοκα, και να αποτελούνται από μηχανικές και ηλεκτρομηχανικές κατασκευές, οι οποίες ονομάζονται «κρυπτομηχανές». Η κρυπτανάλυση τους, απαιτεί μεγάλο αριθμό προσωπικού, το οποίο εργαζόταν επί μεγάλο χρονικό διάστημα ενώ ταυτόχρονα γίνεται εξαιρετικά αισθητή η ανάγκη για μεγάλη υπολογιστική ισχύ. Παρά την πολυπλοκότητα που αποκτούν τα συστήματα

κρυπτογράφησης κατά τη διάρκεια αυτής της περιόδου η κρυπτανάλυση τους είναι συνήθως επιτυχημένη. Οι Γερμανοί έκαναν εκτενή χρήση παραλλαγών ενός συστήματος γνωστού ως Enigma (Εικόνα 3.2).

Εικόνα 3.2: Η κρυπτομηχανή Enigma.

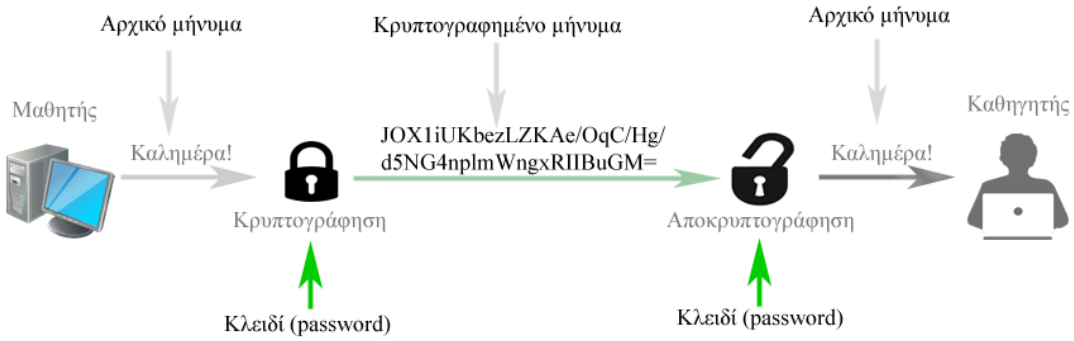
Το 1949 ο Claude Shannon δημοσίευσε το άρθρο «Θεωρία επικοινωνίας των συστημάτων μυστικότητας». Αυτό, εκτός από τις άλλες εργασίες του επάνω στη θεωρία δεδομένων και επικοινωνίας καθιέρωσε μια στερεά θεωρητική βάση για την κρυπτογραφία και την κρυπτανάλυση. Ο Claude Shannon ονομάστηκε από πολλούς ο πατέρας των μαθηματικών συστημάτων κρυπτογραφίας για την συμβολή του αυτή. Εκείνη την εποχή η κρυπτογραφία εξαφανίζεται και φυλάσσεται από τις μυστικές υπηρεσίες κυβερνητικών επικοινωνιών όπως η NSA. Πολύ λίγες εξελίξεις δημοσιοποιήθηκαν ξανά μέχρι τα μέσα της δεκαετίας του '70. Στις 17 Μαρτίου 1975 δημοσιεύτηκε από την IBM το σχέδιο του προτύπου κρυπτογράφησης DES (Data Encryption Standard). Αυτή ήταν μια προσπάθεια να αναπτυχθούν ασφαλείς ηλεκτρονικές εγκαταστάσεις επικοινωνίας για επιχειρήσεις όπως τράπεζες και άλλες μεγάλες οικονομικές οργανώσεις. Το 1977 αφού τροποποιήθηκε από την NSA, ο DES γίνεται ο πρώτος δημόσια προσιτός αλγόριθμος κρυπτογράφησης που εγκρίνεται από μια εθνική αντιπροσωπεία όπως η NSA. Λόγω του ότι ο DES αποκαλύφθηκε ότι ήταν ευάλωτος σε επιθέσεις ωμής βίας (Brute Force Attack), το εθνικό ινστιτούτο προτύπων και τεχνολογίας της Αμερικής (NIST) διεξήγαγε έναν ανοικτό διαγωνισμό για την εύρεση ενός αλγορίθμου κρυπτογράφησης που θα μπορούσε να αντικαταστήσει τον DES. Ο αλγόριθμος που επιλέχθηκε ήταν ο Rijndael, τον οποίο δημιούργησαν δυο βέλγοι ο Vincent Rijmen και ο Joan Daemen. Ο Rijndael τροποποιήθηκε από την NSA και μετονομάστηκε σε AES (Advanced Encryption Standard) (Προηγμένο Πρότυπο Κρυπτογράφησης). Έτσι το 2001 ο DES αντικαταστάθηκε επίσημα από τον AES. [1, 18]

3.3 Κατηγοριοποιήσεις αλγορίθμων κρυπτογράφησης

Οι μοντέρνοι αλγόριθμοι κρυπτογράφησης διακρίνονται σε δύο κατηγορίες, τους συμμετρικούς και τους ασύμμετρους. Στους συμμετρικούς, για την διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης χρησιμοποιείται ένα κλειδί. Αυτό είναι και το μειονέκτημα των συμμετρικών αλγορίθμων, θα πρέπει δηλαδή ο αποστολέας να στείλει στον παραλήπτη το κρυπτογραφημένο μήνυμα αλλά και το κλειδί με το οποίο έχει κρυπτογραφήσει. Μόλις ο παραλήπτης αποκτήσει το κλειδί, τότε θα είναι σε θέση να αποκρυπτογραφήσει το μήνυμα. Στους ασύμμετρους αλγόριθμους κρυπτογράφησης υπάρχουν 2 διαφορετικά κλειδιά, το δημόσιο και το ιδιωτικό. Το δημόσιο χρησιμοποιείται για την κρυπτογράφηση του μηνύματος ενώ το ιδιωτικό για την αποκρυπτογράφηση. Έτσι ο αποστολέας κρυπτογραφεί ένα μήνυμα με το δημόσιο κλειδί του παραλήπτη. Μόλις ο παραλήπτης λάβει το μήνυμα προχωρά στην αποκωδικοποίησή του χρησιμοποιώντας το ιδιωτικό του κλειδί. Το δημόσιο κλειδί όπως υποδηλώνει το όνομά του μπορεί να εκδοθεί ακόμα και σε εφημερίδα, επιτρέποντας στον οποιονδήποτε να κρυπτογραφεί μηνύματα με αυτό. Αντιθέτως το ιδιωτικό κλειδί πρέπει να φυλάσσεται καθώς εάν κάποιος τρίτος το αποκτήσει θα μπορεί να αποκρυπτογραφήσει τα μηνύματά μας. [1, 2, 3]

3.4 Συμμετρική Κρυπτογραφία

Ένα παράδειγμα συμμετρικής κρυπτογραφίας απεικονίζεται στην εικόνα 3.3 όπου ο μαθητής θέλει να στείλει με ασφάλεια ένα μήνυμα στον καθηγητή.



Εικόνα 3.3: Παράδειγμα συμμετρικής κρυπτογραφίας.

Αρχικά ο μαθητής κρυπτογραφεί το μήνυμα που θέλει να στείλει με έναν συμμετρικό αλγόριθμο κρυπτογράφησης (π.χ. AES) χρησιμοποιώντας ένα κλειδί της επιλογής του. Το κρυπτογραφημένο πια μήνυμα είναι δυσανάγνωστο και έτοιμο να αποσταλεί με ασφάλεια με οποιοδήποτε μέσο (π.χ. E-mail, SMS, ταχυδρομείο). Όπως προαναφέρθηκε το κύριο μειονέκτημα της συμμετρικής κρυπτογραφίας είναι το γεγονός ότι με κάποιο τρόπο ο παραλήπτης πρέπει να λάβει και το κλειδί κρυπτογράφησης. Μόλις ο καθηγητής λάβει με επιτυχία το κρυπτογραφημένο μήνυμα και το κλειδί, είναι έτοιμος να προχωρήσει στην αποκρυπτογράφηση και να ανακτήσει το αρχικό μήνυμα.

Οι συμμετρικοί αλγόριθμοι χωρίζονται σε δυο κατηγορίες: τους αλγόριθμους ομάδας (block ciphers) και τους αλγόριθμους στοιχειοσειράς (stream ciphers). Στους αλγόριθμους ομάδας το μήνυμα τεμαχίζεται σε ομάδες από δυαδικά ψηφία (bits) οι

οποίες κρυπτογραφούνται. Αντίθετα στους συμμετρικούς αλγόριθμους κρυπτογραφείται το μήνυμα αυτούσιο χωρίς τεμαχισμό. Ο πίνακας 3.1 παρουσιάζει μερικούς από τους πιο σημαντικούς συμμετρικούς αλγόριθμους.

Αλγόριθμος	**Περιγραφή**	**Μέγεθος κλειδιού (δυαδικά ψηφία)**
Blowfish	Αλγόριθμος ομάδας υλοποιημένος από τον Bruce Schneier.	32 - 448 bits
DES	Ο αλγόριθμος ομάδας που υιοθετήθηκε το 1977 ως κυβερνητικό πρότυπο των ΗΠΑ. Δημιουργήθηκε από την IBM.	56 bits
IDEA	Αλγόριθμος ομάδας υλοποιημένος από τους Massey και Xuejia.	128 bits
MARS	Ήταν ανάμεσα στους διαγωνιζόμενους για τον τίτλο του AES. Δημιουργός του είναι η IBM.	128, 192, 256 bits
RC2	Αλγόριθμος ομάδας υλοποιημένος το 1987 από τον Ron Rivest.	8 - 1024 bits
RC4	Αλγόριθμους στοιχειοσειράς υλοποιημένος το 1987 από τον Ron Rivest.	40 - 2048 bits
RC5	Αλγόριθμος ομάδας, δημοσιοποιήθηκε το 1994 από τον Ron Rivest.	0 - 2040 bits
RC6	Αλγόριθμος ομάδας που διεκδίκησε τον τίτλο του AES. Δημιουργήθηκε από την RSA Labs.	128, 192, 256 bits
Rijndael	Ο αλγόριθμος ομάδας που επιλέχθηκε από την NIST για να αποτελέσει τον AES. Δημιουργήθηκε από τους Vincent Rijmen και Joan Daemen.	128, 192, 256 bits
Serpent	Αλγόριθμος ομάδας που διεκδίκησε τον τίτλο του AES. Δημοσιεύτηκε το 1998 από τους Ross Anderson, Eli Biham και Lars Knudsen.	128, 192, 256 bits
Triple-DES	Αλγόριθμος ομάδας, τριπλή εφαρμογή του αλγορίθμου DES.	56, 112, 168 bits
Twofish	Αλγόριθμος ομάδας που δημοσιοποιήθηκε το 1998 και δημιουργήθηκε από τον Bruce Schneier.	128, 192, 256 bits

Πίνακας 3.1: Συμμετρικοί αλγόριθμοι κρυπτογραφίας.

3.5 Κρυπτογραφικός αλγόριθμος ομάδας (Block Cipher)

Οι αλγόριθμοι ομάδας ανήκουν στην κατηγορία της συμμετρικής κρυπτογράφησης. Μετατρέπουν μια ομάδα μη κρυπτογραφημένου κειμένου (plaintext), σε ομάδες κρυπτογραφημένου κειμένου (ciphertext) ομοίου μήκους. Η μετατροπή αυτή πραγματοποιείται με την βοήθεια ενός μυστικού κλειδιού. Το καθορισμένο μήκος των κρυπτογραφημένων ομάδων ονομάζεται μέγεθος ομάδας (block size) και οι περισσότεροι αλγόριθμοι διαθέτουν 128, 192 και 256 δυαδικά ψηφία (bits). Όταν χρησιμοποιούμε έναν αλγόριθμο ομάδας για να κρυπτογραφήσουμε ένα μήνυμα ακαθόριστου μήκους, χρησιμοποιούμε τεχνικές που είναι γνωστές ως καταστάσεις λειτουργίας (modes) του αλγορίθμου ομάδας. Οι γνωστότερες καταστάσεις λειτουργίας είναι οι παρακάτω:

- Electronic Codebook (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter (CTR)
- Propagating Cipher Block Chaining (PCBC)
- XOR-encrypt-XOR (XEX)
- XEX-based tweaked-codebook mode with ciphertext stealing (XTS)

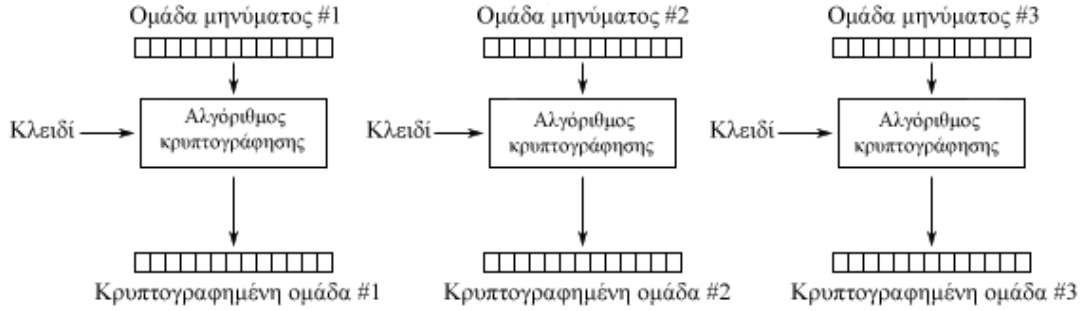
Επειδή οι αλγόριθμοι ομάδας λειτουργούν με ομοίου μήκους ομάδες (π.χ. 128 bits) προκύπτει η ανάγκη για γέμισμα της τελευταίας ομάδας εάν έχει μικρότερο μήκος από το μήκος ομάδας (π.χ. 100 bits). Η διαδικασία γεμίσματος ομάδας λέγεται «πλήρωση» (padding). Η συνηθέστερη διαδικασία γεμίσματος είναι η εισαγωγή λογικού ‘1’ και μιας ακολουθίας τόσων λογικών μηδέν ‘0’ στο τέλος του μη κρυπτογραφημένου κειμένου (plaintext) μέχρι να γεμίσει η ομάδα. Στον πίνακα 3.2 παρουσιάζεται ένα παράδειγμα.

Μη κρυπτογραφημένο κείμενο μεγέθους 104 bits
01001011011000010110110001101000011011010110010101110010011000010010000 00111001101100001011100110010000 1
Γεμισμένο μη κρυπτογραφημένο κείμενο μεγέθους 128 bits
01001011011000010110110001101000011011010110010101110010011000010010000 00111001101100001011100110010000 1**10000000000000000000000**

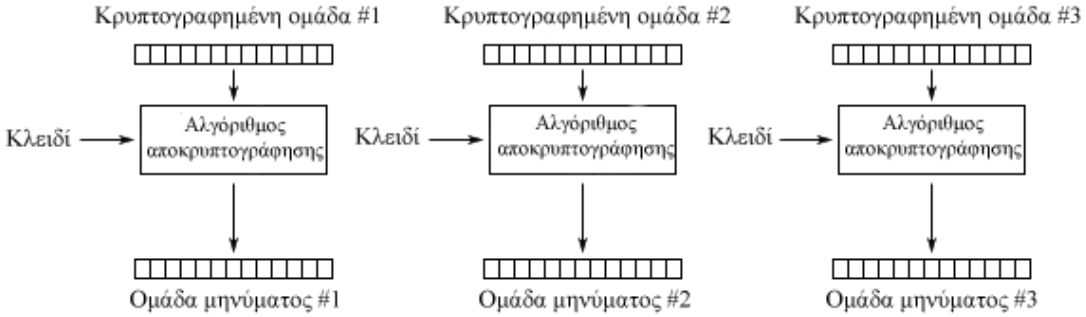
Πίνακας 3.2: Διαδικασία γεμίσματος ομάδας (padding).

Electronic Codebook (ECB)

Η πιο απλή κατάσταση λειτουργίας είναι η ECB στην οποία το μήνυμα διαιρείται σε ομάδες και κάθε ομάδα κρυπτογραφείται ξεχωριστά. Η διαδικασία της κρυπτογράφησης και αποκρυπτογράφησης παρουσιάζεται στις εικόνες 3.4 και 3.5 αντίστοιχα.



Εικόνα 3.4: Κατάσταση λειτουργίας Electronic Codebook (ECB), κρυπτογράφηση.



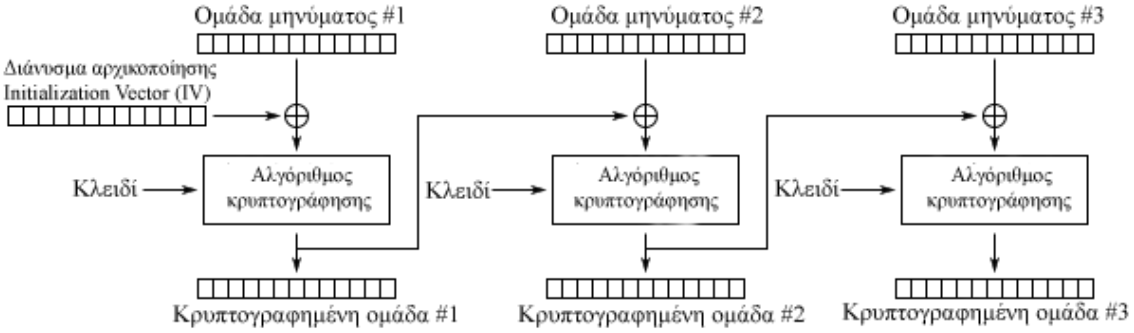
Εικόνα 3.5: Κατάσταση λειτουργίας Electronic Codebook (ECB), αποκρυπτογράφηση.

Κύριο μειονέκτημα αυτής της κατάστασης λειτουργίας είναι ότι η κρυπτογράφηση δύο ίδιων ομάδων μηνύματος μετασχηματίζει δύο πανομοιότυπες κρυπτογραφημένες ομάδες. Πλεονέκτημα αποτελεί ότι η διαδικασία κρυπτογράφησης και αποκρυπτογράφησης μπορούν να γίνουν παράλληλα.

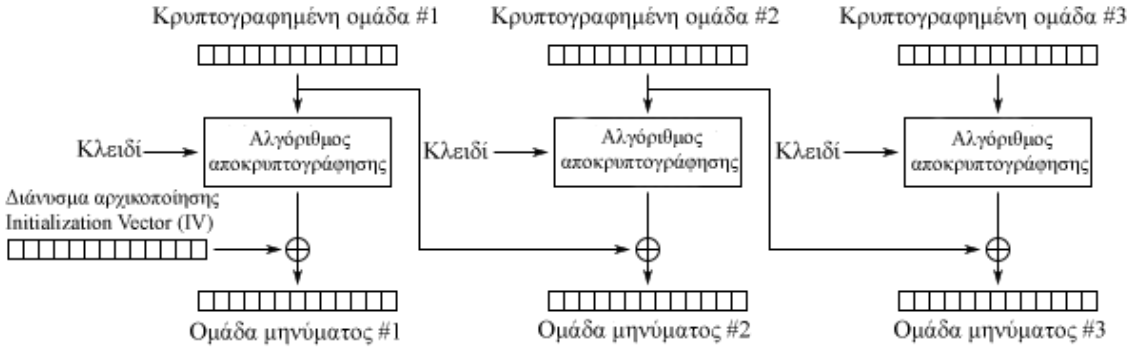
Cipher Block Chaining (CBC)

Οι Ehrsam, Meyer, Smith και Tuchman εφηύραν την κατάσταση λειτουργίας Cipher Block Chaining (CBC) το 1976. Η λειτουργία αυτή διορθώνει το μειονέκτημα της ECB καθώς όταν υπάρχουν όμοιες ομάδες μηνύματος (plaintext) δεν μετατρέπονται σε όμοιες κρυπτογραφημένες ομάδες. Αυτό επιτυγχάνεται υλοποιώντας XOR μιας ομάδας μηνύματος με την προηγούμενη κρυπτογραφημένη ομάδα όπως φαίνεται στην εικόνα 3.6. Επειδή η πρώτη ομάδα μηνύματος δεν έχει προηγούμενο κρυπτογραφημένο μήνυμα, εισάγουμε ένα διάνυσμα αρχικοποίησης (Initialization Vector) (IV) και υλοποιούμε XOR μεταξύ τους. Το διάνυσμα αρχικοποίησης πρέπει να είναι πάντα τυχαία τιμή. Στην αποκρυπτογράφηση εφαρμόζεται η ανάστροφη διαδικασία για την οποία χρειάζεται εκτός των άλλων και το διάνυσμα αρχικοποίησης. Συνήθως εισάγεται πριν την πρώτη κρυπτογραφημένη ομάδα και μεταφέρεται στον δέκτη μαζί με το κρυπτογραφημένο μήνυμα. Ο τρόπος αυτός για την μεταφορά του διανύσματος

αρχικοποίησης δεν μειώνει την ασφάλεια της κρυπτογράφησης καθώς η δουλειά του ήταν να βοηθήσει ώστε η πρώτη κρυπτογραφημένη να μην παίρνει την ίδια τιμή για ίδιο κλειδί και ομάδα μηνύματος. Η εικόνα 3.7 απεικονίζει την διαδικασία αποκρυπτογράφησης με την κατάσταση λειτουργίας CBC.



Εικόνα 3.6: Κατάσταση λειτουργίας Cipher Block Chaining (CBC), κρυπτογράφηση.



Εικόνα 3.7: Κατάσταση λειτουργίας Cipher Block Chaining (CBC), αποκρυπτογράφηση.

Η μαθηματική συνάρτηση για την διαδικασία της κρυπτογράφησης είναι η παρακάτω:

$$\mathbf{Ci=E_K(P_i \oplus C_{i-1})}$$

$$\mathbf{C_0=IV}$$

Όπου **i** ο αριθμός ομάδας, **E** η διαδικασία κρυπτογράφησης (encryption), **K** το κλειδί κρυπτογράφησης (key), **P** η ομάδα μηνύματος (plaintext), $\oplus$ η λογική πράξη XOR, **C** η κρυπτογραφημένη ομάδα (ciphertext). Το $\mathbf{C_0}$ αποτελεί το διάνυσμα αρχικοποίησης (**IV**).

Αντίστοιχα η διαδικασία αποκρυπτογράφησης (**D**) είναι η ακόλουθη:

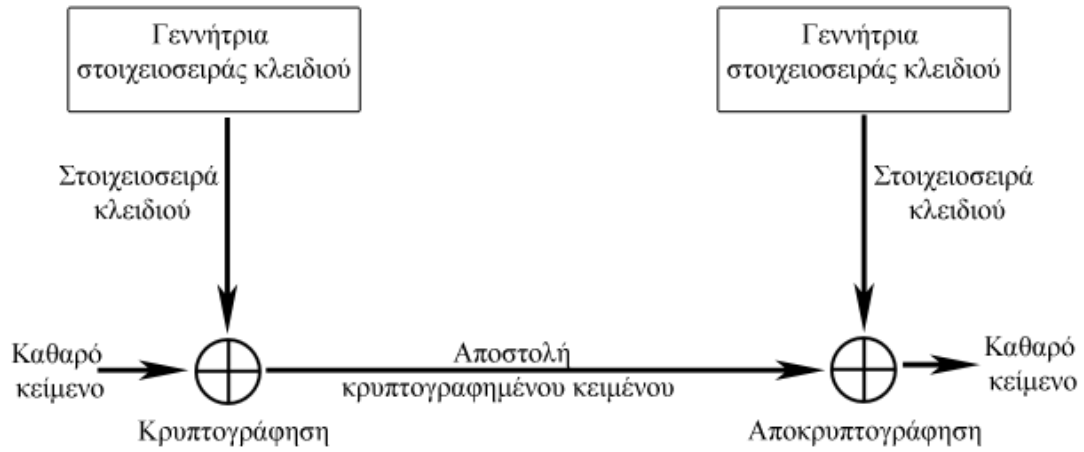
$$\mathbf{Pi=D_K(C_i) \oplus C_{i-1}}$$

$$\mathbf{C_0=IV}$$

Στην κατάσταση λειτουργίας CBC η διαδικασία της αποκρυπτογράφησης δεν μπορεί να γίνει παράλληλα σε αντίθεση με την ECB που υποστηρίζει και παράλληλη κρυπτογράφηση.

3.6 Κρυπτογραφικός αλγόριθμος ροής (Stream Cipher)

Ο κρυπτογραφικός αλγόριθμος ροής ή στοιχειοσειράς αποτελεί ένα είδος συμμετρικού αλγόριθμου κρυπτογράφησης. Αναπτύχθηκαν έτσι ώστε να είναι πολύ γρηγορότεροι από οποιονδήποτε κρυπτογραφικό αλγόριθμο ομάδας. Οι αλγόριθμοι ροής δημιουργούν μια ακολουθία από δυαδικά ψηφία που χρησιμοποιείται ως κλειδί κρυπτογράφησης, η ακολουθία αυτή ονομάζεται στοιχειοσειρά κλειδιού ή κλειδοροή (key stream). Στην εικόνα 3.8 παρουσιάζεται η κρυπτογράφηση και αποκρυπτογράφηση με έναν τέτοιου είδους αλγόριθμο.



Εικόνα 3.8: Κρυπτογράφηση και αποκρυπτογράφηση με αλγόριθμο στοιχειοσειράς (Stream Cipher).

Έστω ότι το καθαρό κείμενο είναι το γράμμα "G" που στο δυαδικό σύστημα έχει την τιμή 01000111 η γεννήτρια στοιχειοσειράς κλειδιού παράγει την ακολουθία 10001010.

Στην κρυπτογράφηση το κάθε δυαδικό ψηφίο του "G" θα υποστεί XOR με το αντίστοιχο του στην ακολουθία κλειδιού. Παρακάτω έχουμε τον πίνακα αληθείας της XOR (πίνακας 3.3) και δεξιά του την διαδικασία κρυπτογράφησης (πίνακας 3.4).

Είσοδος		Έξοδος
A	**B**	
0	0	0
0	1	1
1	0	1
1	1	0

Πίνακας 3.3: Πίνακας αληθείας XOR.

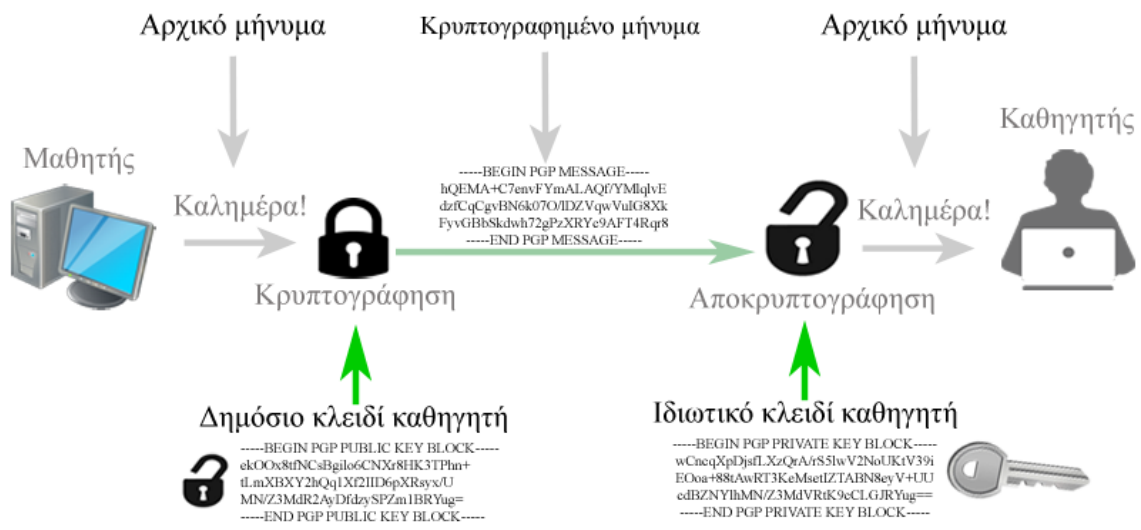
"G"	01000111
Κλειδί	10001010
XOR	11001101

Πίνακας 3.4: Η λογική πράξη XOR του κλειδιού και του "G".

3.7 Ασύμμετρη Κρυπτογραφία

Η ασύμμετρη κρυπτογραφία (Asymmetric Cryptography) γνωστή και ως κρυπτογράφηση δημοσίου κλειδιού (Public Key Cryptography) εφευρέθηκε στο τέλος της δεκαετίας του 1970 από τους Whitfield Diffie και Martin Hellman και αποτελεί ένα εντελώς διαφορετικό μοντέλο διαχείρισης των κλειδιών κρυπτογράφησης συγκριτικά με την κρυπτογράφηση συμμετρικού κλειδιού.

Όπως και στο παράδειγμα της συμμετρικής κρυπτογράφησης έτσι και σε αυτό ο μαθητής θέλει να στείλει κρυπτογραφημένο μήνυμα στον καθηγητή. Με την βοήθεια μιας γεννήτριας κλειδιών ο κάθε ένας τους αποκτά ένα ζευγάρι κλειδιών. Το ζευγάρι αποτελείται από το δημόσιο κλειδί (public key) και το ιδιωτικό κλειδί (private key). Το δημόσιο κλειδί θα μπορούσε να παρομοιαστεί σαν ένα ανοικτό λουκέτο το οποίο ανοίγει μόνο με το ιδιωτικό κλειδί. Ένα δημόσιο κλειδί μπορεί να αναρτηθεί δημόσια σε ιστοσελίδες χωρίς να επηρεαστεί η ασφάλεια της κρυπτογραφίας. Στην εικόνα 3.9 παρουσιάζεται ένα παράδειγμα χρήσης του ασύμμετρου αλγορίθμου. Αρχικά ο μαθητής χρησιμοποιεί το δημόσιο κλειδί του καθηγητή για να κρυπτογραφήσει το μήνυμα που θέλει να αποστείλει.



Εικόνα 3.9: Κρυπτογράφηση μηνύματος με ασύμμετρο αλγόριθμο.

Εν συνεχεία το κρυπτογραφημένο μήνυμα στέλνεται στον καθηγητή ο οποίος με την σειρά του το αποκρυπτογραφεί με το ιδιωτικό του κλειδί για να διαβάσει τελικά το μήνυμα.

Οι αλγόριθμοι ασύμμετρης κρυπτογραφίας αποτελούν τον πυρήνα στην ασφάλεια κρυπτοσυστημάτων, εφαρμογών και πρωτοκόλλων. Είναι ο θεμέλιος λίθος των διαδικτυακών προτύπων όπως Transport Layer Security (TLS), S/MIME (Secure/Multipurpose Internet Mail Extensions), Pretty Good Privacy (PGP), GNU Privacy Guard (GnuPG ή GPG), Secure Shell (SSH). Κάποιοι από τους σημαντικότερους αλγορίθμους του είδους τους είναι ο Diffie–Hellman και ο RSA. Ο αλγόριθμος Diffie–Hellman δημιουργήθηκε από τους Whitfield Diffie και Martin

Hellman το 1976 και αποτέλεσε την θεμελίωση μεταγενέστερων αλγορίθμων. Ο RSA δημιουργήθηκε από τους Ron Rivest, Adi Shamir και Leonard Adleman το 1977 και η ασφάλειά του έγκειται στην δυσκολία της παραγοντοποίησης του αποτελέσματος δυο μεγάλων πρώτων αριθμών. Άξιος αναφοράς είναι και ο αλγόριθμος ElGamal ο οποίος βασίζεται στον Diffie–Hellman και χρησιμοποιείται στα διαδικτυακά πρότυπα GPG. Ο ElGamal παρουσιάστηκε από τον Taher Elgamal το 1985.

3.8 Πρότυπο Κρυπτογράφησης AES

Το πρότυπο AES (Advanced Encryption Standard) (FIPS-197) περιγράφει έναν συμμετρικό αλγόριθμο ομάδας ο οποίος υποστηρίζει κλειδιά μήκους 128, 192, 256 δυαδικών ψηφίων. Το μήκος ομάδας του αλγορίθμου είναι 128 δυαδικά ψηφία. Ο αλγόριθμος κατά την κρυπτογράφηση εκτελεί έναν αριθμό επαναλήψεων ανάλογα με το μήκος κλειδιού που του έχει ανατεθεί (πίνακας 3.5).

Μέγεθος κλειδιού	128 bits	192 bits	256 bits
Αριθμός επαναλήψεων	10	12	14

Πίνακας 3.5: Επαναλήψεις ανάλογα με το μέγεθος κλειδιού.

Στην ανάπτυξη της παρούσας εφαρμογής επιλέχθηκε για την κρυπτογράφηση των δεδομένων ο AES σε κατάσταση λειτουργίας CBC. Ως συμπληρωματικό χαρακτηριστικό ασφαλείας επιλέχθηκε η τεχνική ***Χρησιμοποίησης Περεταίρω Πληροφορίας ΧΠΠ*** (**salting**) για το κλειδί του αλγορίθμου κρυπτογράφησης. Έτσι, το κλειδί κρυπτογράφησης αποτελείται από δυο μέρη. Το πρώτο μέρος είναι το κλειδί που εισάγει ο χρήστης και το δεύτερο είναι μια σταθερή ακολουθία αλφαριθμητικών που δημιουργείται από την εφαρμογή. Τα δυο αυτά μέρη συνδυάζονται και παράγουν το κλειδί κρυπτογράφησης. Η τεχνική αυτή προστατεύει τα κρυπτογραφημένα δεδομένα από επιθέσεις τύπου εξαντλητικής αναζήτησης (Brute Force) και επιθέσεις με την χρήση λεξικού (Dictionary Attack).

Η επίθεση εξαντλητικής αναζήτησης είναι αρκετά απλή αλλά απαιτεί σχετικά μεγάλη υπολογιστική ισχύ για να έχει 100% επιτυχία. Εάν για παράδειγμα έχουμε κρυπτογραφήσει δεδομένα χρησιμοποιώντας για κλειδί την ημερομηνία γέννησής μας π.χ. «1990», η επίθεση θα αρχίσει την αναζήτηση του σωστού κλειδιού μέχρι να τα καταφέρει ξεκινώντας από το 0. Μετά από ελάχιστες (για έναν υπολογιστή) διαδοχικές προσπάθειες (0, 1, 2, … 9, 00, 01, 02, ... 99, … 0000, … 1111, … 1989) το κλειδί θα έχει βρεθεί. Αντίστοιχα εάν το κλειδί απαρτίζεται από μικρούς χαρακτήρες η επίθεση ξεκινάει από το «α» και τελειώνει στο «ωωωωωωωωωω». Φυσικά ο αριθμός των «ω» ορίζεται από τον επιτιθέμενο και τους πόρους τους οποίους είναι διατεθειμένος να διαθέσει. Ομοίως και με τα κεφαλαία, σύμβολα, αλλόγλωσσους χαρακτήρες. Ένα θεωρητικά ασφαλές κλειδί πρέπει να απαρτίζεται από τουλάχιστον 8 χαρακτήρες εκ των οποίων να υπάρχει τουλάχιστον ένα σύμβολο, ένας αριθμός, ένας κεφαλαίος και ένας μικρός χαρακτήρας.

Η επίθεση με την χρήση λεξικού είναι πιο εκλεπτυσμένη καθώς ο επιτιθέμενος δοκιμάζει για κλειδί έτοιμες λέξεις. Χρησιμοποιεί λίστες με τις πιο διαδεδομένες ακολουθίες κλειδιών. Οι λίστες αυτές είναι της μορφής:

123456, password, 12345678, 1234, qwerty, 111111, dragon, abc123, football …

Εφαρμόζοντας την τεχνική ***ΧΠΠ*** (**salting**) διασφαλίζουμε ότι όσο κακό και να είναι το κλειδί που θα εισάγει ο χρήστης η ασφάλεια της κρυπτογράφησης θα μείνει ανέπαφη. Για παράδειγμα εάν η συμπληρωματική ακολουθία (salt) είναι «Pc[W#/_q.8u~Jk\eY.sE@#jh» (τυχαίοι χαρακτήρες) και το κλειδί που εισάγει ο χρήστης «123456» τότε ο επιτιθέμενος θα πρέπει να μαντέψει την ακολουθία «123456Pc[W#/_q.8u~Jk\eY.sE@#jh». Για να καταλάβουμε πόσο όφελος έχουμε από την τεχνική αυτή, χρησιμοποιούμε ένα διαδικτυακό εργαλείο [16] για να υπολογίσουμε πόσο χρόνο θα χρειαστεί ο επιτιθέμενος για να παραβιάσει τον αλγόριθμο.
Εάν ο επιτιθέμενος έχει την δυνατότητα να ‘μαντεύει’ χίλια (1000) κλειδιά το δευτερόλεπτο τότε:

- Για να ανακαλύψει το κλειδί «123456» θα χρειαστεί περίπου 19 λεπτά.
- Ενώ για το κλειδί «123456Pc[W#/_q.8u~Jk\eY.sE@#jh» θα χρειαστεί … 6.897.000.000.000.000.000.000.000.000.000.000.000.000.000.000 χιλιετίες !

Άρα είναι προφανές το όφελος της χρήσης μια τέτοιας τεχνικής. [1, 2, 3]

4. Υλοποίηση της εφαρμογής με την γλώσσα προγραμματισμού C#

Η εφαρμογή υλοποιήθηκε στο Microsoft Visual Studio 2015 στην γλώσσα C#, βασίστηκε στο .NET Framework 4 και αποτελείται από τέσσερις (4) φόρμες και μια κλάση:

1. **DragAndDrop.cs** (εισαγωγή εικόνας)
2. **EncryptForm.cs** (στεγανογράφηση της δοσμένης εικόνας)
3. **DecryptForm.cs** (αποστεγανογράφηση της δοσμένης εικόνας)
4. **OptionsForm.cs** (παραμετροποιήσεις επιλογών από τον χρήστη)
5. **AES.cs** (κλάση κρυπτογράφησης-αποκρυπτογράφησης AES)

Όλες οι φόρμες αποτελούνται από συναρτήσεις που είναι υπεύθυνες για την αισθητική απεικόνιση των φορμών αυτών (Themes). Οι συναρτήσεις αυτές ανασύρουν τις τιμές χρωματικού κώδικα που ο χρήστης επιλέγει με την βοήθεια μιας συνάρτησης της φόρμας **OptionsForm.cs**. Οι τιμές αυτές είναι αποθηκευμένες στον φάκελο AES_Steganography ο οποίος βρίσκεται στον προσωρινό φάκελο των λειτουργικών συστημάτων Windows (%temp%). Μόλις οι τιμές ανασυρθούν τότε οι συναρτήσεις αλλάζουν τα στοιχεία της φόρμας με τα ανάλογα χρώματα. Παρακάτω ακολουθεί η εξήγηση λειτουργίας τους.
ChangeBackColor(): Η συνάρτηση αλλάζει το χρώμα φόντου των φορμών της εφαρμογής.
ChangeTextColor(): Όπως δηλώνει και το όνομά της αλλάζει το χρώμα των χαρακτήρων της φόρμας, όμως όχι των χαρακτήρων που πληκτρολογεί ο χρήστης.
ChangeBorderColor(): Αλλάζει το χρώμα της μπάρας τίτλου της φόρμας.
ChangeThemeColor(): Η συνάρτηση αυτή θέτει σε λειτουργία τις παραπάνω τρείς συναρτήσεις.

Επίσης υπάρχει η συνάρτηση **ReadOptions()** η οποία ευθύνεται για την ανάγνωση δύο επιλογών του χρήστη. Οι επιλογές αυτές είναι:

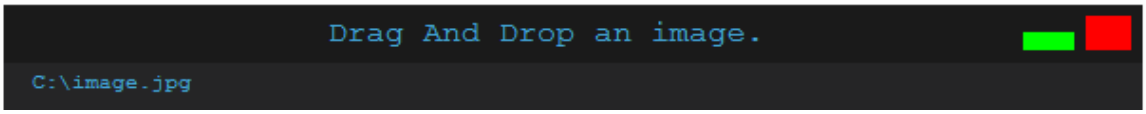
1. Επιλογή του προεπιλεγμένου μεγέθους κλειδιού κρυπτογράφησης.
2. Επιλογή μορφής διαθέσιμου χώρου στην εικόνα.

Η συνάρτηση ανασύρει τις επιλογές και τις εφαρμόζει στα κατάλληλα πεδία κατά την εκκίνηση της φόρμας που την περιέχει.

Ιδιαίτερο ενδιαφέρον αποτελεί το γεγονός ότι από όλες τις φόρμες έχει αφαιρεθεί η προεπιλεγμένη μπάρα τίτλου των Windows (Εικόνα 4.1).

Εικόνα 4.1: Προεπιλεγμένη μπάρα τίτλου στα Windows 7.

Όλες οι φόρμες αποτελούνται και από συναρτήσεις (Theme Components) που είναι υπεύθυνες για την λειτουργικότητα της παραμετροποιημένης μπάρας τίτλου (Εικόνα 4.2).



Εικόνα 4.2: Παραμετροποιημένη μπάρα τίτλου.

Στα παρακάτω υποκεφάλαια ακολουθούν οι τέσσερις φόρμες και η κλάση της εφαρμογής με εκτενέστερο σχολιασμό.

4.1 Η αρχική φόρμα (DragAndDrop.cs)

Στην εικόνα 4.3 παρατηρούμε την μορφή της αρχικής φόρμας DragAndDrop.cs μετά την εισαγωγή μιας εικόνας.

Εικόνα 4.3: Όψη αρχικής φόρμας (DragAndDrop.cs) μετά την εισαγωγή κάποιας εικόνας.

Η φόρμα αυτή έχει ως σκοπό να καλέσει τις φόρμες **EncryptForm.cs** ή **DecryptForm.cs** βάσει των ενεργειών που θα υποδείξει ο χρήστης. Μόλις εισαχθεί μια εικόνα και πατηθεί το κουμπί Encrypt ή Decrypt τότε καλείται η αντίστοιχη φόρμα ή οποία θα πάρει σαν όρισμα την διεύθυνση της εικόνας.

Παρακάτω ακολουθεί ο κώδικας της φόρμας αυτής με κάποια σχόλια.

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace AES_Steganography_Chania
{
    public partial class DragAndDrop : Form
    {
        //timerBlink variable for blinking label
        int i = 6;
        Point LastClick;
        public static string GlobalImagePath;
        string BackColorPath = Path.GetTempPath() +
"\\AES_Steganography\\BackgroundColor.txt";
        string BorderColorPath = Path.GetTempPath() +
"\\AES_Steganography\\BorderColor.txt";
        string TextColorPath = Path.GetTempPath() +
"\\AES_Steganography\\TextColor.txt";

        public DragAndDrop()
        {
            InitializeComponent();
            // Read and Apply Theme Colors
            ChangeThemeColor();
        }
        /// Custom Theme Components
        #region Theme Components
```

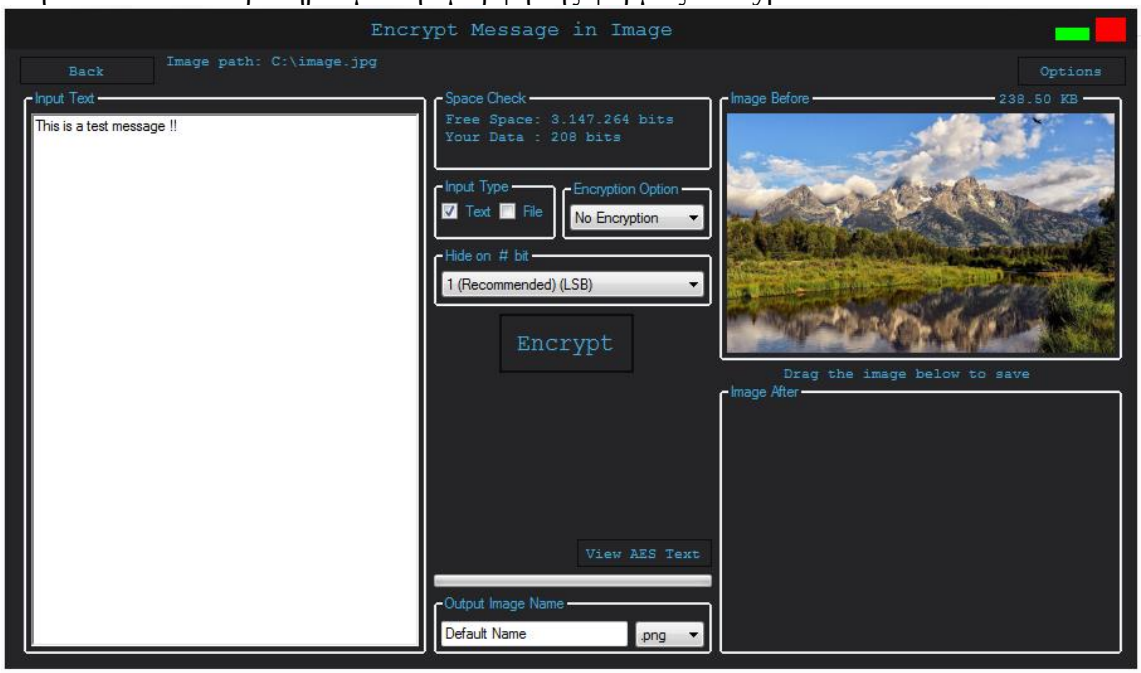
```
        private void CustomExit_Click(object sender, EventArgs e)
        {}
        private void CustomBorder_MouseDown(object sender, MouseEventArgs e)
        {}
        private void CustomBorder_MouseMove(object sender, MouseEventArgs e)
        {}
        private void CustomLabel_MouseDown(object sender, MouseEventArgs e)
        {}
        private void CustomLabel_MouseMove(object sender, MouseEventArgs e)
        {}
        private void CustomMinimize_Click(object sender, EventArgs e)
        {}
        #endregion Theme Components

        /// Read and Apply Theme Colors
        #region Theme Colors
        private void ChangeThemeColor()
        {}
        private void ChangeBackColor()
        {}
        private void ChangeBorderColor()
        {}
        private void ChangeTextColor()
        {}
        #endregion Theme Colors

        /// Drag and Drop Components
        #region Drag and Drop
        private void DragAndDrop_DragDrop(object sender, DragEventArgs e)
        {}
        private void DragAndDrop_DragEnter(object sender, DragEventArgs e)
        {}
        #endregion Drag and Drop

        /// Buttons
        private void ButtonEncrypt_Click(object sender, EventArgs e)
        {
            // If image is available send the path on Encrypt Form
            if (LabelImagePath.Visible == true)
            {
                EncryptForm frm = new EncryptForm(GlobalImagePath);
                frm.Show();
                this.Hide();
            }
            // If image is not available Blink label "No Image added..."
            else
            {
                timerBlink.Interval = 250;
                timerBlink.Start();
            }
        }
        private void ButtonDecrypt_Click(object sender, EventArgs e)
        {
            // If image is available send the path on Decrypt Form
            if (LabelImagePath.Visible == true)
            {
                DecryptForm frm = new DecryptForm(GlobalImagePath);
                frm.Show();
                this.Hide();
            }
            // If image is not available Blink label "No Image added..."
            else
            {
                timerBlink.Interval = 250;
                timerBlink.Start();
```

```
            }
        }

        // Timer for blinking label
        private void timerBlink_Tick(object sender, EventArgs e)
        {
            if (i > 0)
            {
                i--;
                if (labelNoImageAdded.ForeColor == Color.Red)
                {
                    labelNoImageAdded.ForeColor = Color.FromArgb(68, 169, 221);
                }
                else
                {
                    labelNoImageAdded.ForeColor = Color.Red;
                }
            }
            else
            {
                timerBlink.Stop();
            }
        }
    }
}
```

Η συνάρτηση **`timerBlink_Tick`**, που υπάρχει στο τέλος, υλοποιεί ένα χρονόμετρο (timer) για να αναβοσβήνει ένα μήνυμα στην περίπτωση που ο χρήστης πατήσει κάποιο κουμπί χωρίς να έχει εισάγει εικόνα. Για ευνόητους λόγους εδώ παρουσιάζονται εκείνα τα αποσπάσματα του κώδικα που έχουν περισσότερο ενδιαφέρον, ενώ ολόκληρος ο κώδικας ευρίσκεται στο CD που κατατέθηκε μαζί με την παρούσα πτυχιακή εργασία.

4.2 Η φόρμα στεγανογράφησης (EncryptForm.cs)

Στην εικόνα 4.4 παρατηρούμε την μορφή της φόρμας EncryptForm.cs.



Εικόνα 4.4: Όψη φόρμας στεγανογράφησης (EncryptForm.cs).

Η φόρμα αυτή αποτελείται από κουτιά επιλογής (checkboxes, comboboxes), ετικέτες (labels) των οποίων οι συναρτήσεις λειτουργίας έχουν αφαιρεθεί από τον παρακάτω κώδικα για να είναι πιο εύκολα αναγνώσιμος. Το μέγεθος της συμβολοσειράς **defaultsalt** έχει μειωθεί από 2048 σε 76 χαρακτήρες.

```
using System;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Imaging;
using System.IO;
using System.Globalization;

namespace AES_Steganography_Chania
{
    public partial class EncryptForm : Form
    {
        /// Variable Declaration
        Point LastClick;
        private Bitmap bmp1 = null;
        private Bitmap StealthBMP = null;
        int KeySize = 256;
        private string AEStext;
        // The salt string for AES cipher
        private string defaultsalt = "4-jjMb*khDzRb#V*YUm8WRN02i*CfWVaRlPnUlXD*
KnyL+sS+?FZbS8nmp28#Vw7j^Y9a#LT%";
        string BackColorPath = Path.GetTempPath() +
"\\AES_Steganography\\BackgroundColor.txt";
        string BorderColorPath = Path.GetTempPath() +
"\\AES_Steganography\\BorderColor.txt";
        string TextColorPath = Path.GetTempPath() +
"\\AES_Steganography\\TextColor.txt";
        string AESoptionFile = Path.GetTempPath() +
"\\AES_Steganography\\AESOptions.txt";
        string Base64File = null;
        int TotalPixels = 0;
        int mask = 1;
        private string ImagePath;
        string SpaceFormat;
        public static bool optionsflag = false;


        ///Form Initialization
        public EncryptForm(string GlobalImagePath)
        {
            InitializeComponent();
            // Get and save the selected image path from previous form
            LabelImagePath.Text = "Image path: " + GlobalImagePath;
            // Print Image size on label eg. 164.32 KB
            labelImageSize.Text = CalculateImageSize(GlobalImagePath, null);
            // Preview Image before
            bmp1 = new Bitmap(GlobalImagePath);
            pictureBoxImageBefore.Image = bmp1;
            pictureBoxImageBefore.SizeMode = PictureBoxSizeMode.StretchImage;
            // Save image path on string(ImagePath)
            ImagePath = GlobalImagePath;
            // Set default value for image format ComboBox
            comboBoxFormat.SelectedIndex = 0;
            // Set default value for Encryption ComboBox
            comboBoxEncryptOptions.SelectedIndex = 0;
```

```
            // Set default value for Mask ComboBox
            comboBoxMask.SelectedIndex = 0;
            // Load Options
            ReadOptions();
            // Read and Apply Theme Colors
            ChangeThemeColor();
            // Check Free Space of image
            CalculateFreeSpace();
            // Check input data size
            CalculateTextSize();
        }

        /// Custom Theme Components
        #region Theme Components
        private void CustomExit_Click(object sender, EventArgs e)
        {}
        private void CustomBorder_MouseDown(object sender, MouseEventArgs e)
        {}
        private void CustomBorder_MouseMove(object sender, MouseEventArgs e)
        {}
        private void CustomMinimize_Click(object sender, EventArgs e)
        {}
        private void CustomLabel_MouseDown(object sender, MouseEventArgs e)
        {}
        private void CustomLabel_MouseMove(object sender, MouseEventArgs e)
        {}
        #endregion Theme Components

        /// Read and Apply Theme Colors
        #region Theme Colors
        private void ChangeThemeColor()
        {}
        private void ChangeBackColor()
        {}
        private void ChangeBorderColor()
        {}
        private void ChangeTextColor()
        {}
        #endregion Theme Colors

        /// Buttons
        private void ButtonEncrypt_Click(object sender, EventArgs e)
        {
            Cursor.Current = Cursors.WaitCursor;
            try
            {
                if (labelDataSpace.ForeColor != Color.Red)
                {
                    // Insert Text
                    if (checkBoxTextInput.Checked == true)
                    {
                        Encrypt(richTextBoxInput.Text);
                        Cursor.Current = Cursors.Default;
                    }
                    // Insert File
                    else
                    {
                        Encrypt(Base64File);
                        Cursor.Current = Cursors.Default;
                    }
                }
```

```
                else
                {
                    MessageBox.Show("Insufficient storage! ");
                }
            }
            catch
            {
                Cursor.Current = Cursors.Default;
            }
        }
        // Button to display ciphertext
        private void ButtonAEStext_Click(object sender, EventArgs e)
        {
            MessageBox.Show(AEStext, "Cipher Text");
        }
        private void buttonBack_Click(object sender, EventArgs e)
        {}
        private void buttonGeneralOptions_Click(object sender, EventArgs e)
        {}

        // Drag image to save
        private void pictureBoxImageAfter_MouseDown(object sender, MouseEventArgs e)
        {}

        // Erase invalid chars on savename textbox
        private void textBoxSaveName_TextChanged(object sender, EventArgs e)
        {}

        // Encryption Options ComboBox
        private void comboBoxEncryptOptions_SelectedIndexChanged(object sender,
EventArgs e)
        {}

        // Mask ComboBox
        private void comboBoxMask_SelectedIndexChanged(object sender, EventArgs e)
        {}

        // Input text size
        private void richTextBoxInput_TextChanged(object sender, EventArgs e)
        {
            CalculateTextSize();
        }
        // Calculate Text Size
        private void CalculateTextSize()
        {}

        // Calculate Image Size on KBytes, MBytes

        private string CalculateImageSize(string ImagePath, Image ByteImage)
        {}

        // Input <Double> -> Output <String> with format <00.00 B/KB/MB>
        private string DoubleToKB_MB(double inputFileSize)
        {}

        // Load Options from temp folder
        // Read Options from "/Temp/AES_Steganography/AESOptions.txt" and save the
string to OptionsText
        private void ReadOptions()
        {}
```

```
        #region CheckBoxes
        // Drop file for encryption
        private void textBoxFileDrop_DragEnter(object sender, DragEventArgs e)
        {}
        private void textBoxFileDrop_DragDrop(object sender, DragEventArgs e)
        {}

        // If this checkBox is checked, TEXT will be encrypted.
        private void checkBoxTextInput_MouseUp(object sender, MouseEventArgs e)
        {}
        // If this checkBox is checked, FILE will be encrypted.
        private void checkBoxFileInput_MouseUp(object sender, MouseEventArgs e)
        {}
        #endregion CheckBoxes

        // calling AES encryption
        private void Encrypt(string Input)
        {
            // TempBMP ===> StealthBMP
            Bitmap TempBMP = new Bitmap(ImagePath);

            if (Input.Equals(""))
            {
                MessageBox.Show("The text you want to hide can't be empty",
"Warning");
            }
            else
            {
                if (comboBoxEncryptOptions.Text == "No Encryption")
                {
                    StealthBMP = InsertText((Input), TempBMP, mask);
                    pictureBoxImageAfter.Image = StealthBMP;
                    pictureBoxImageAfter.SizeMode = PictureBoxSizeMode.StretchImage;
                    labelImageSizeAfter.Visible = true;
                    labelImageSizeAfter.Text = CalculateImageSize(null, StealthBMP);
                }
                else if (comboBoxEncryptOptions.Text == "No Encryption Greek")
                {
                    if (checkBoxTextInput.Checked != true)
                    {
                        comboBoxEncryptOptions.SelectedIndex = 0;
                    }
                    else
                    {
                        // (UTF8)text to base64 generic combatibility Greek letters
(10bit char ===> 8bit char)
                        var plainTextBytes =
Encoding.UTF8.GetBytes(richTextBoxInput.Text);
                        string Base64Text = Convert.ToBase64String(plainTextBytes);
                        StealthBMP = InsertText((Base64Text), TempBMP, mask);
                        pictureBoxImageAfter.Image = StealthBMP;
                        pictureBoxImageAfter.SizeMode =
PictureBoxSizeMode.StretchImage;
                        labelImageSizeAfter.Visible = true;
                        labelImageSizeAfter.Text = CalculateImageSize(null,
StealthBMP);
                    }
                }
                else if ((comboBoxEncryptOptions.Text != "No
Encryption")&(comboBoxEncryptOptions.Text != "No Encryption Greek"))
                {
```

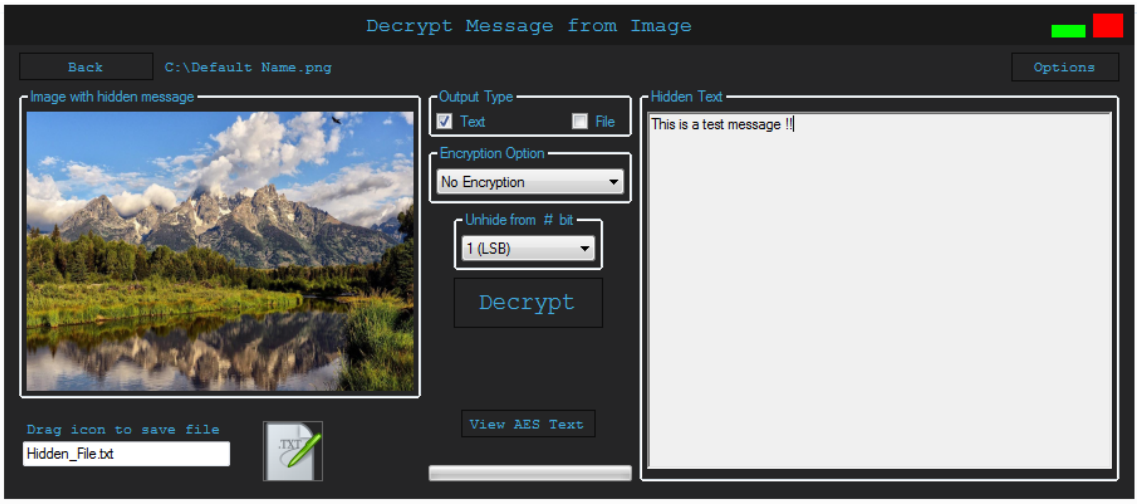
```
                    if (richTextBoxPassword.Text.Length < 6)
                    {
                        MessageBox.Show("Please enter a password with at least 6
characters", "Warning");
                    }
                    else
                    {
                        AEStext = AES.EncryptStringAES(Input,
richTextBoxPassword.Text, defaultsalt, KeySize);
                        StealthBMP = InsertText(AEStext, TempBMP, mask);
                        pictureBoxImageAfter.Image = StealthBMP;
                        pictureBoxImageAfter.SizeMode =
PictureBoxSizeMode.StretchImage;
                        labelImageSizeAfter.Visible = true;
                        labelImageSizeAfter.Text = CalculateImageSize(null,
StealthBMP);
                    }
                }
            }
        }

        // Calculate available space of input image and print the result on
Label(labelFreeSpace)
        private void CalculateFreeSpace()
        {}

        // Calculate size of input data and print the result on Label(labelDataSpace)
        private void CalculateDataSpace(string InputString)
        {}

        // Input <integer> -> Output <string> with format <0.00
bits/Kbits/Mbits/BytesKBytesMBytes>
        private string ValueFormater(int input)
        {}

        // Function with string input and binary array output (10010110)
        private string StringToBinary(string data)
        {}

        /// Insert Data to image Core (Steganography)
        public Bitmap InsertDataToIMG(string text, Bitmap bmpA, int mask)
        {
            return bmp;
        }
    }
}
```

Η συνάρτηση **ButtonEncrypt_Click** καλείται όταν ο χρήστης πατήσει το κουμπί *Encrypt*. Η συνάρτηση αυτή με την σειρά της καλεί την συνάρτηση **Encrypt** δίνοντάς της ως όρισμα το κείμενο ή το αρχείο που έχει εισάγει ο χρήστης. Αν ο χρήστης έχει επιλέξει κρυπτογράφηση τότε η συνάρτηση **Encrypt** καλεί την συνάρτηση **EncryptStringAES** από την κλάση **AES.cs** (που περιγράφεται στο υποκεφάλαιο 4.5) για να κρυπτογραφήσει τα δεδομένα. Στην συνέχεια η **Encrypt** στέλνει τα κρυπτογραφημένα (ή μη-κρυπτογραφημένα) δεδομένα μαζί με την εικόνα και την μάσκα που έχει επιλέξει ο χρήστης, στην συνάρτηση **InsertDataToIMG** η οποία

επιστρέφει την στεγανογραφημένη εικόνα. Ο πλήρης κώδικας της **InsertDataToIMG** βρίσκεται στο CD που παραδόθηκε με την παρούσα πτυχιακή εργασία.

4.3 Η φόρμα αποστεγανογράφησης (DecryptForm.cs)

Στην εικόνα 4.5 παρατηρούμε την μορφή της φόρμας DecryptForm.cs.



Εικόνα 4.5: Όψη φόρμας αποστεγανογράφησης (DecryptForm.cs).

Όπως και στις υπόλοιπες φόρμες έτσι κι εδώ υπάρχουν συναρτήσεις που έχουν απλοποιηθεί από τον παρακάτω κώδικα. Οι σημαντικότερες συναρτήσεις περιγράφονται στο τέλος του παρόντος υποκεφαλαίου.

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;

namespace AES_Steganography_Chania
{
    public partial class DecryptForm : Form
    {
        /// Variable Declaration
        Point LastClick;
        int KeySize = 256;
        private Bitmap bmp1 = null;
        private string AESstext;
        private string defaultsalt = "4-jjMb*khDzRb#V*YUm8WRN02i*CfWVaRlPnUlXD*
KnyL+sS+?FZbS8nmp28#Vw7j^Y9a#LT%";
        string BackColorPath = Path.GetTempPath() +
"\\AES_Steganography\\BackgroundColor.txt";
        string BorderColorPath = Path.GetTempPath() +
"\\AES_Steganography\\BorderColor.txt";
        string TextColorPath = Path.GetTempPath() +
"\\AES_Steganography\\TextColor.txt";
        string Base64File = null;
        int mask = 1;
        public static bool optionsflag = false;

        public DecryptForm(string GlobalImagePath)
        {
```

```
            InitializeComponent();
            // Get and save the selected image path from previous form
            LabelImagePath.Text = GlobalImagePath;
            // Preview Image
            bmp1 = new Bitmap(GlobalImagePath);
            pictureBoxEncryptedImage.Image = bmp1;
            pictureBoxEncryptedImage.SizeMode = PictureBoxSizeMode.StretchImage;
            // Set default value for Encryption ComboBox
            comboBoxEncryptOptions.SelectedIndex = 0;
            // Set default value for Mask ComboBox
            comboBoxMask.SelectedIndex = 0;
            //Read Options
            ReadOptions();
            // Read and Apply Theme Colors
            ChangeThemeColor();
        }

        /// Custom Theme Components
        #region Theme Components
        private void CustomExit_Click(object sender, EventArgs e)
        {}
        private void CustomBorder_MouseDown(object sender, MouseEventArgs e)
        {}
        private void CustomBorder_MouseMove(object sender, MouseEventArgs e)
        {}
        private void CustomMinimize_Click(object sender, EventArgs e)
        {}
        private void CustomLabel_MouseDown(object sender, MouseEventArgs e)
        {}
        private void CustomLabel_MouseMove(object sender, MouseEventArgs e)
        {}
        #endregion Theme Components

        /// Read and Apply Theme Colors
        #region Theme Colors
        private void ChangeThemeColor()
        {}
        private void ChangeBackColor()
        {}
        private void ChangeBorderColor()
        {}
        private void ChangeTextColor()
        {}
        #endregion Theme Colors

        /// Buttons
        private void ButtonDecrypt_Click(object sender, EventArgs e)
        {
            Cursor.Current = Cursors.WaitCursor;
            try
            {
                bmp1 = (Bitmap)pictureBoxEncryptedImage.Image;
                if (comboBoxEncryptOptions.Text == "No Encryption")
                {
                    string PlainText = WithdrawDataFromIMG(bmp1, mask);
                    if (checkBoxTextOutput.Checked == true)
                    {   try
                        {
                            richTextBoxOutput.Text = PlainText;
                        }
                        catch
```

```
                        {
                        }
                        Cursor.Current = Cursors.Default;
                    }
                    else
                    {   // file output
                        Base64File = PlainText;
                        richTextBoxOutput.Text = "Drag the icon to save the file !";
                        Cursor.Current = Cursors.Default;
                    }
                }
                else if (comboBoxEncryptOptions.Text == "No Encryption Greek")
                {
                    string Base64Text = WithdrawDataFromIMG(bmp1, mask);

                        try
                        {   // text output from (base64 input text)
                            richTextBoxOutput.Text =
System.Text.Encoding.UTF8.GetString(Convert.FromBase64String(Base64Text));
                        }
                        catch
                        {
                            richTextBoxOutput.Text = Base64Text;
                        }
                        Cursor.Current = Cursors.Default;
                }
                else if ((comboBoxEncryptOptions.SelectedIndex == 2)|
(comboBoxEncryptOptions.SelectedIndex == 3) | (comboBoxEncryptOptions.SelectedIndex
== 4))
                {
                    if (richTextBoxPassword.Text.Length < 6)
                    {
                        MessageBox.Show("Please enter a password with at least 6
characters", "Warning");
                        Cursor.Current = Cursors.Default;
                    }
                    else
                    {
                        try
                        {
                            AEStext = WithdrawDataFromIMG(bmp1, mask);

                            if (checkBoxTextOutput.Checked == true)
                            {   // text output
                                richTextBoxOutput.Text =
AES.DecryptStringAES(AEStext, richTextBoxPassword.Text, defaultsalt, KeySize);
                                Cursor.Current = Cursors.Default;
                            }
                            else
                            {   // file output
                                Base64File = AES.DecryptStringAES(AEStext,
richTextBoxPassword.Text, defaultsalt, KeySize);
                                richTextBoxOutput.Text = "Drag the icon to save the
file !";
                                Cursor.Current = Cursors.Default;
                            }
                        }
                        catch
                        {
                            MessageBox.Show("Wrong Password or Method", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
                            Cursor.Current = Cursors.Default;
                        }
                    }
                }
            }
            catch
            {
                MessageBox.Show("Steganography Error", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                Cursor.Current = Cursors.Default;
            }
        }
        private void ButtonAEStext_Click(object sender, EventArgs e)
        {}
        private void textBoxSaveName_TextChanged(object sender, EventArgs e)
        {}
        private void pictureBoxSaveText_MouseDown(object sender, MouseEventArgs e)
        {}
        private void buttonBack_Click(object sender, EventArgs e)
        {}
        private void buttonGeneralOptions_Click(object sender, EventArgs e)
        {}

        /// Decryption Options
        private void comboBoxEncryptOptions_SelectedIndexChanged(object sender,
EventArgs e)
        {}
        // Mask ComboBox
        private void comboBoxMask_SelectedIndexChanged(object sender, EventArgs e)
        {}

        /// Output Type CheckBoxes
        private void checkBoxFileOutput_MouseUp(object sender, MouseEventArgs e)
        {}
        private void checkBoxTextOutput_MouseUp(object sender, MouseEventArgs e)
        {}

        /// Read AES options from temp folder
        private void ReadOptions()
        {}

        /// Withdrawing data from image (Steganography)
        private string WithdrawDataFromIMG(Bitmap bmp, int mask)
        {}

        /// Calculate Hidden Data
        /// This function scans the image and returns the number of steganographed
pixels.
        private int CalculateHiddenData()
        {}
    }
}
```

Στον παραπάνω κώδικα βασικό ρόλο κατέχει η συνάρτηση **ButtonDecrypt_Click** η οποία μαζεύει όλες τις επιλογές του χρήστη και εν συνεχεία καλεί δύο συναρτήσεις.

1. Αρχικά καλεί την **WithdrawDataFromIMG** η οποία παίρνει ως είσοδο την εικόνα και την μάσκα (θέση κρυμμένης πληροφορίας). Η συνάρτηση αυτή

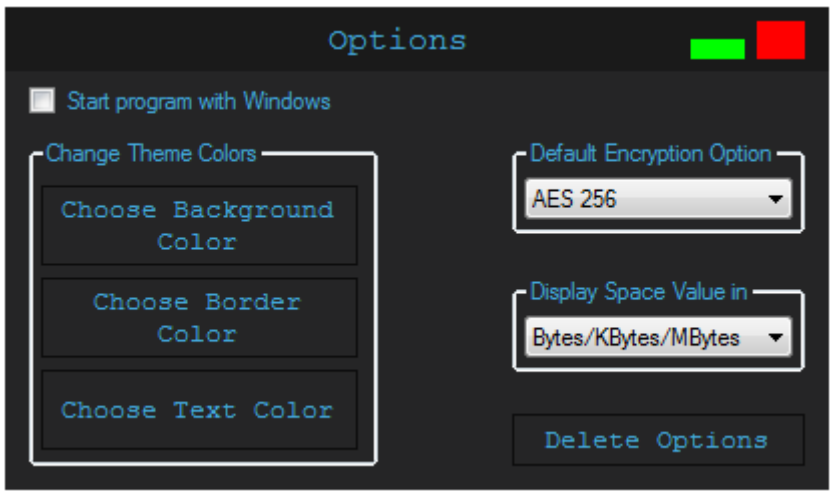
εφαρμόζοντας αποστεγανογράφηση στην εικόνα, ανασύρει την κρυμμένη πληροφορία η οποία μπορεί να είναι κρυπτογραφημένη (κλήση 2) ή ακρυπτογράφητη (εμφάνιση πληροφορίας και τερματισμός συνάρτησης).

2. Εάν ο χρήστης έχει επιλέξει αποκρυπτογράφηση τότε η **ButtonDecrypt_Click** καλεί ακόμα μια συνάρτηση την **DecryptStringAES** που βρίσκεται στην κλάση **AES.cs**. Η συνάρτηση αυτή παίρνει σαν είσοδο α) την κρυπτογραφημένη πληροφορία που είχε εξάγει από την εικόνα η προηγούμενη συνάρτηση (**WithdrawDataFromIMG**), β) το κλειδί κρυπτογράφησης που εισάγει ο χρήστης, γ) το μέγεθος κλειδιού κρυπτογράφησης, δ) την συμβολοσειρά **defaultsalt**. Τέλος η αποκρυπτογραφημένη πληροφορία στέλνεται πίσω στην **ButtonDecrypt_Click** η οποία την παρουσιάζει στο παράθυρο κειμένου **richTextBoxOutput**.

Στα πλαίσια λειτουργίας στης **WithdrawDataFromIMG** βοηθάει η συνάρτηση **CalculateHiddenData()** η οποία εφαρμόζει μια γρήγορη προσπέλαση στην εικόνα προκειμένου να μετρήσει τον όγκο της στεγανογραφημένης πληροφορίας. Ο όγκος αυτός είναι χρήσιμος για την ενημέρωση της μπάρας προόδου ώστε ο χρήστης να γνωρίζει ανά πάσα στιγμή την εξέλιξη της αποστεγανογράφησης.

4.4 Η φόρμα ρυθμίσεων (OptionsForm.cs)

Στην εικόνα 4.6 παρατηρούμε την μορφή της φόρμας OptionsForm.cs.



Εικόνα 4.6: Όψη φόρμας ρυθμίσεων (OptionsForm.cs).

Παρακάτω αναγράφονται όλες οι συναρτήσεις πλην αυτών που ανασύρουν δεδομένα επιλογών του χρήστη οι οποίες παραλείπονται.

```
using Microsoft.Win32;
using System;
using System.Drawing;
using System.IO;
using System.Windows.Forms;
```

```
namespace AES_Steganography_Chania
{
    public partial class OptionsForm : Form
    {
        string AESoptionFile = Path.GetTempPath() +
"\\AES_Steganography\\AESOptions.txt";
        string SpaceOptionFile = Path.GetTempPath() +
"\\AES_Steganography\\SpaceOptions.txt";
        string BackColorPath = Path.GetTempPath() +
"\\AES_Steganography\\BackgroundColor.txt";
        string BorderColorPath = Path.GetTempPath() +
"\\AES_Steganography\\BorderColor.txt";
        string TextColorPath = Path.GetTempPath() +
"\\AES_Steganography\\TextColor.txt";
        Point LastClick;
        // AES keysize
        string KeySize;
        // Free space option
        string SpaceValueInt;

        public OptionsForm()
        {
            InitializeComponent();
            //Read Options
            ReadOptions();
            // Read and Apply Theme Colors
            ChangeThemeColor();
        }

        // StartUp Checkbox
        private void checkBoxStartup_CheckStateChanged(object sender, EventArgs e)
        {
            if (checkBoxStartup.Checked==true)
            {
                try
                {
                    using (RegistryKey key =
Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",
true))
                    {
                        key.SetValue("AES Steganography", "\"" +
Application.ExecutablePath + "\"");
                    }
                }
                catch
                { }
            }
            else
            {
                using (RegistryKey key =
Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",
true))
                {
                    key.DeleteValue("AES Steganography", false);
                }
            }
        }
        // Encryption Options
        private void comboBoxEncryptOptions_SelectedIndexChanged(object sender,
EventArgs e)
        {
```

```
            if (comboBoxEncryptOptions.SelectedIndex == 0)
            {
                KeySize = "0";
            }
            else if (comboBoxEncryptOptions.SelectedIndex == 1)
            {
                KeySize = "128";
            }
            else if (comboBoxEncryptOptions.SelectedIndex == 2)
            {
                KeySize = "192";
            }
            else if (comboBoxEncryptOptions.SelectedIndex == 3)
            {
                KeySize = "256";
            }
            SaveOptions(KeySize, AESoptionFile);
        }
        // Space Value Options
        private void comboBoxSpace_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (comboBoxSpace.SelectedIndex == 0)
            {
                SpaceValueInt = "0";
            }
            else if (comboBoxSpace.SelectedIndex == 1)
            {
                SpaceValueInt = "1";
            }
            else if (comboBoxSpace.SelectedIndex == 2)
            {
                SpaceValueInt = "2";
            }
            else if (comboBoxSpace.SelectedIndex == 3)
            {
                SpaceValueInt = "3";
            }
            SaveOptions(SpaceValueInt, SpaceOptionFile);
        }

        // Creating AES_Steganography Directory
        // Saves options to %temp%\AES_Steganography\OptionsFile.txt
        private void SaveOptions(string Input,string Filepath)
        {
            Directory.CreateDirectory(Path.GetTempPath() + "\\AES_Steganography");
            File.WriteAllText(Filepath,Input);
        }
        private void buttonDeleteOptions_Click(object sender, EventArgs e)
        {
            string OptionsDirectory = Path.GetTempPath() + "\\AES_Steganography";
            Directory.Delete(OptionsDirectory, true);
        }
        private void ReadOptions()
        {}

        /// Custom Theme Components
        private void CustomExit_Click(object sender, EventArgs e)
        {}
        private void CustomBorder_MouseDown(object sender, MouseEventArgs e)
        {}
        private void CustomBorder_MouseMove(object sender, MouseEventArgs e)
```

```
        {}
        private void CustomMinimize_Click(object sender, EventArgs e)
        {}
        private void CustomLabel_MouseDown(object sender, MouseEventArgs e)
        {}
        private void CustomLabel_MouseMove(object sender, MouseEventArgs e)
        {}

        /// Read and Apply Theme Colors
        private void ChangeThemeColor()
        {}
        private void ChangeBackColor()
        {}
        private void ChangeBorderColor()
        {}
        private void ChangeTextColor()
        {}

        /// Saving Theme Colors to Temp
        // Picking a color and saves it (hex) on %temp%\AES_Steganography\Color.txt
        // Creating AES_Steganography Directory
        private void BackgroundColor_Click(object sender, EventArgs e)
        {
            if (colorDialog1.ShowDialog() == DialogResult.OK)
            {
                string ColorHex = HexConverter(colorDialog1.Color);
                Directory.CreateDirectory(Path.GetTempPath() + "\\AES_Steganography");
                File.WriteAllText(BackColorPath, ColorHex);
                ChangeThemeColor();
            }
        }
        private void BorderColor_Click(object sender, EventArgs e)
        {
            if (colorDialog2.ShowDialog() == DialogResult.OK)
            {
                string ColorHex = HexConverter(colorDialog2.Color);
                Directory.CreateDirectory(Path.GetTempPath() + "\\AES_Steganography");
                File.WriteAllText(BorderColorPath, ColorHex);
                ChangeThemeColor();
            }
        }
        private void TextColor_Click(object sender, EventArgs e)
        {
            if (colorDialog3.ShowDialog() == DialogResult.OK)
            {
                string ColorHex = HexConverter(colorDialog3.Color);
                Directory.CreateDirectory(Path.GetTempPath() + "\\AES_Steganography");
                File.WriteAllText(TextColorPath, ColorHex);
                ChangeThemeColor();
            }
        }

        // Convert Color to HEX
        private static String HexConverter(Color c)
        {
            return "#" + c.R.ToString("X2") + c.G.ToString("X2") + c.B.ToString("X2");
        }
    }
}
```

Η φόρμα ρυθμίσεων δίνει την δυνατότητα στον χρήστη να μεταβάλει τις προεπιλεγμένες τιμές σε στοιχεία της εφαρμογής όπως:

1. Χρώματα των φορμών της εφαρμογής (**BackgroundColor_Click**, **BorderColor_Click**, **TextColor_Click**).
2. Αλλαγή της προεπιλεγμένης μονάδας μέτρησης (bits, bytes) διαθέσιμου χώρου για στεγανογραφία στην εικόνα (**comboBoxSpace_SelectedIndexChanged**).
3. Αλλαγή του προεπιλεγμένου μεγέθους κλειδιού κρυπτογράφησης (**comboBoxEncryptOptions_SelectedIndexChanged**).
4. Επιλογή εκκίνησης της εφαρμογής κατά της εκκίνηση του λειτουργικού συστήματος (**checkBoxStartup_CheckStateChanged**).

Όλες αυτές οι παραμετροποιήσεις αποθηκεύονται στον προσωρινό φάκελο του συστήματος (*C:\Users\Username\AppData\Local\Temp\AES_Steganography*) και μπορούν να διαγραφούν με το πάτημα του κουμπιού **Delete Options**.

4.5 Η κλάση κρυπτογραφίας (AES.cs)

Η κλάση αυτή αποτελείται από δυο βασικές συναρτήσεις, την 'EncryptStringAES' και την 'DecryptStringAES'. Οι συναρτήσεις αυτές παίρνουν τέσσερις εισόδους από το κυρίως πρόγραμμα.

1. Το κείμενο που θα κρυπτογραφήσουν ή θα αποκρυπτογραφήσουν.
2. Το κλειδί κρυπτογράφησης που ορίζεται από τον χρήστη.
3. Το επιπλέον κλειδί (salt) το οποίο αναφέρεται στο κεφάλαιο 3.8.
4. Το μέγεθος κλειδιού κρυπτογράφησης για τον AES (128, 192, 256 bits).

Ο κώδικας της κλάσης είναι ο παρακάτω:

```
using System;
using System.Text;
using System.Security.Cryptography;
using System.IO;
using System.Windows.Forms;

namespace AES_Steganography_Chania
{
    public class AES
    {
        public static string EncryptStringAES(string plainText, string userPassword,
string salt, int keySize)
        {
            byte[] userSalt = Encoding.ASCII.GetBytes(salt);
            // Encrypted string to return
            string OutputString = null;
            // RijndaelManaged object used to encrypt the data.
            RijndaelManaged AESrijndael = null;
            try
            {
                // Generate the key from the shared secret and the salt
                Rfc2898DeriveBytes key = new Rfc2898DeriveBytes(userPassword, userSalt);
                // Create a RijndaelManaged object
                AESrijndael = new RijndaelManaged();
```

```
                // Set the key size
                AESrijndael.KeySize = keySize;
                // Give the generated key to RijndaelManaged object
                AESrijndael.Key = key.GetBytes(AESrijndael.KeySize / 8);
                // Choose Cipher Block Chaining (CBC) Mode
                AESrijndael.Mode = CipherMode.CBC;
                //  Create an encryptor to perform the stream transform.
                ICryptoTransform encryptor = AESrijndael.CreateEncryptor(AESrijndael.Key,
AESrijndael.IV);
                // Create the streams used for encryption.
                using (MemoryStream MSencrypt = new MemoryStream())
                {
                    // Generating and Writing the IV (Initialization Vector)
                    MSencrypt.Write(BitConverter.GetBytes(AESrijndael.IV.Length), 0,
sizeof(int));
                    MSencrypt.Write(AESrijndael.IV, 0, AESrijndael.IV.Length);
                    using (CryptoStream CSencrypt = new CryptoStream(MSencrypt,
encryptor, CryptoStreamMode.Write))
                    {
                        using (StreamWriter SWencrypt = new StreamWriter(CSencrypt))
                        {
                            //Write all data to the stream.
                            SWencrypt.Write(plainText);
                        }
                    }
                    OutputString = Convert.ToBase64String(MSencrypt.ToArray());
                }
            }
            finally
            {
                // Clear the RijndaelManaged object.
                if (AESrijndael != null)
                    AESrijndael.Clear();
            }
            // Return the encrypted bytes from the memory stream.
            return OutputString;
        }

        public static string DecryptStringAES(string cipherText, string userPassword,
string salt, int keySize)
        {
            // Check for null inputs and catch errors
            if (salt==null)
            {
                MessageBox.Show("Error with decryption Salt", "Error");
            }
            if (cipherText==null)
            {
                MessageBox.Show("Error with encrypted Text", "Error");
            }
            if (userPassword==null)
            {
                MessageBox.Show("Error with Password", "Error");
            }
            // Save given salt
            byte[] userSalt = Encoding.ASCII.GetBytes(salt);
            // Declare the RijndaelManaged object used to decrypt the data.
            RijndaelManaged AESrijndael = null;
            // Declare the string used to hold the decrypted text.
            string plaintext = null;
            try
```

```
            {
                // Generate the key from the password and the salt
                Rfc2898DeriveBytes key = new Rfc2898DeriveBytes(userPassword, userSalt);
                // Create the streams used for decryption.
                byte[] bytes = Convert.FromBase64String(cipherText);
                using (MemoryStream MSdecrypt = new MemoryStream(bytes))
                {
                    // Create a RijndaelManaged object with the specified key and IV.
                    AESrijndael = new RijndaelManaged();
                    // Set the key size
                    AESrijndael.KeySize = keySize;
                    // Choose Cipher Block Chaining (CBC) Mode
                    AESrijndael.Mode = CipherMode.CBC;
                    // Give the generated key to RijndaelManaged object
                    AESrijndael.Key = key.GetBytes(AESrijndael.KeySize / 8);
                    // Get the Initialization Vector from the encrypted stream
                    AESrijndael.IV = ReadByteArray(MSdecrypt);
                    // Create a decrytor to perform the stream transform.
                    ICryptoTransform decryptor =
AESrijndael.CreateDecryptor(AESrijndael.Key, AESrijndael.IV);
                    using (CryptoStream CSdecrypt = new CryptoStream(MSdecrypt,
decryptor, CryptoStreamMode.Read))
                    {
                        using (StreamReader SRdecrypt = new StreamReader(CSdecrypt))
                            // Read the decrypted bytes from the decrypting stream and
place them in a string.
                            plaintext = SRdecrypt.ReadToEnd();
                    }
                }
            }
            finally
            {
                // Clear the RijndaelManaged object.
                if (AESrijndael != null)
                    AESrijndael.Clear();
            }
            return plaintext;
        }

        // Function for Stream to Byte convertion
        private static byte[] ReadByteArray(Stream s)
        {}
    }
}
```

Ο ρόλος της μεθόδου **Rfc2898DeriveBytes**, η οποία ευρίσκεται στην βιβλιοθήκη του Visual Studio που λέγεται `System.Security.Cryptography`, θεωρείται πολύ σημαντικός για την ασφάλεια της κρυπτογράφησης.

```
Rfc2898DeriveBytes key = new Rfc2898DeriveBytes(userPassword, userSalt);
```

Όπως βλέπουμε χρησιμοποιεί ως είσοδο το κλειδί που εισάγει ο χρήστης και την συμβολοσειρά userSalt (επιπλέον κλειδί) για να δημιουργήσει ένα νέο κλειδί (key) το οποίο αποτελεί συνδυασμό των άλλων δυο. Το νέο κλειδί θα έχει το μέγεθος (key size) που χρειάζεται ο αλγόριθμος κρυπτογράφησης για να λειτουργήσει. Εάν δεν υπήρχε η συνάρτηση αυτή θα είχαμε το παρακάτω αποτέλεσμα.

Ο χρήστης εισάγει κλειδί μήκους 40 δυαδικών ψηφίων (bits) πχ. «hallo».

Εάν υπάρχει επιπλέον κλειδί (salt) θα έχει συγκεκριμένο μέγεθος πχ. 100 bits.

Ο συνδυασμός αυτών των δυο κλειδιών θα δώσει ένα κλειδί μεγέθους 140 bits (εάν συνενωθούν).

Αποτέλεσμα ο αλγόριθμος κρυπτογράφησης δεν θα λειτουργήσει για καμία επιλογή μεγέθους κλειδιού (128, 192, 256 bits).

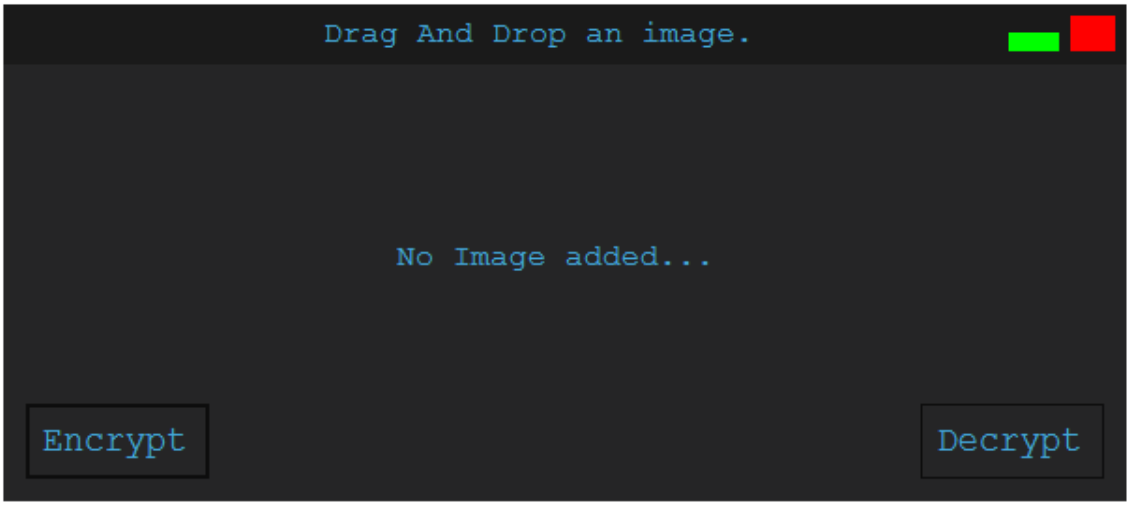
Η συνάρτηση **Rfc2898DeriveBytes** ούτε λίγο ούτε πολύ παράγει κλειδιά μεγέθους όσο το μέγεθος κλειδιού (128, 192, 256 bits) ανεξάρτητα από το μήκος των δυο κλειδιών που παίρνει ως είσοδο.

Για την ασφαλή υλοποίηση της παραπάνω κλάσης έπαιξε σημαντικό ρόλο ο ιστότοπος της Microsoft [17] και οι επεξηγήσεις σημαντικών κρυπτογραφικών συναρτήσεων.

5. Οδηγός χρήσης της εφαρμογής

5.1 Εισαγωγή εικόνας στην εφαρμογή

Κατά την υλοποίηση της εφαρμογής μεγάλη βαρύτητα δόθηκε στον σχεδιασμό ενός περιβάλλοντος φιλικό προς τον χρήστη. Για να εισάγουμε μια εικόνα της επιλογής μας στην εφαρμογή απλά τραβάμε το εικονίδιό της και το αφήνουμε πάνω στην εφαρμογή. Στην Εικόνα 5.1 παρατηρούμε την αρχική φόρμα χωρίς εικόνα ενώ η Εικόνα 5.2 εμφανίζει την ίδια φόρμα μετά την εισαγωγή μιας εικόνας.



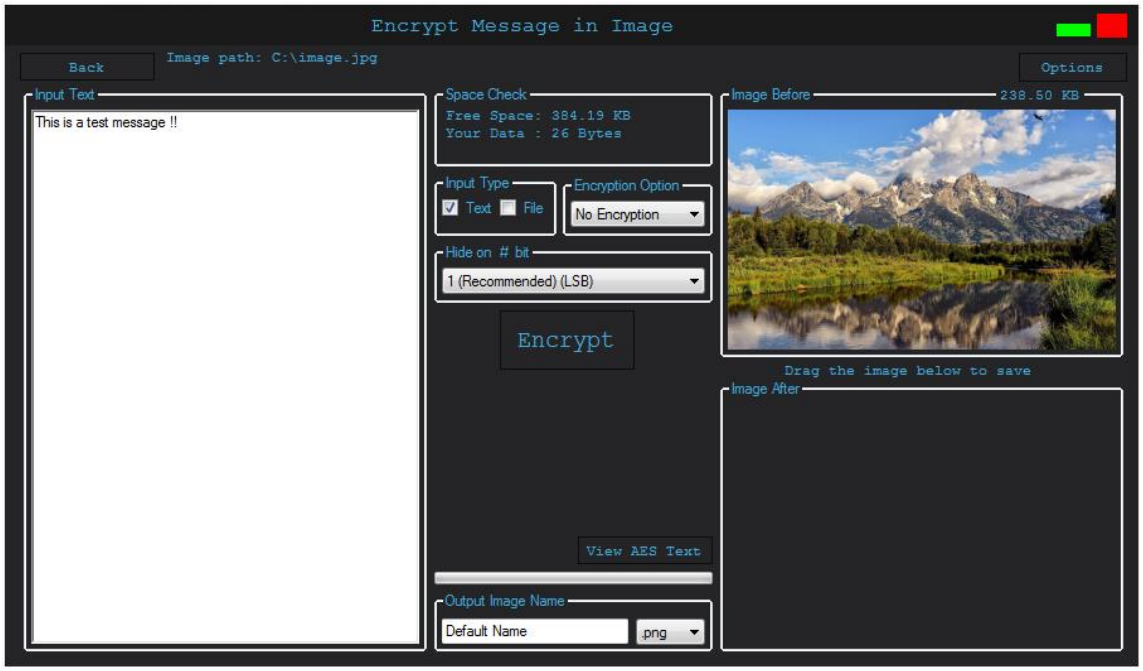
Εικόνα 5.1: Η αρχική φόρμα χωρίς εικόνα.

Εικόνα 5.2: Η αρχική φόρμα με εικόνα.

Στην συνέχεια μπορούμε να επιλέξουμε Encrypt για στεγανογράφηση (κεφ. 5.2) ή Decrypt για αποστεγανογράφηση (κεφ. 5.3).

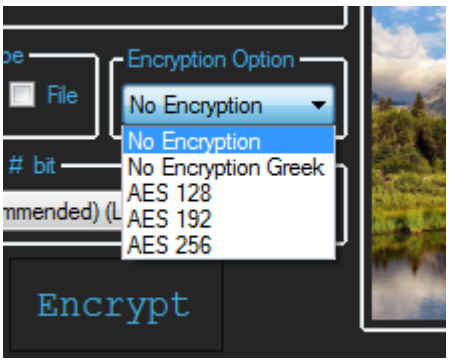
5.2 Στεγανογράφηση

Έχοντας επιλέξει μια εικόνα και πατήσει το κουμπί Encrypt, ανοίγει η παρακάτω φόρμα (Εικόνα 5.3).



Εικόνα 5.3: Φόρμα Encrypt με τις προεπιλεγμένες επιλογές.

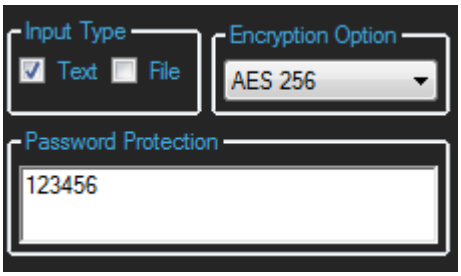
Για να εισάγουμε ένα κείμενο στην εικόνα, πρώτα εξασφαλίζουμε ότι στον τομέα **Input Type** είναι επιλεγμένο το κουτάκι **Text**. Στον τομέα **Input Text** πληκτρολογούμε το κείμενο της επιλογής μας (ή το επικολλούμε). Εάν θέλουμε να κρυπτογραφηθεί το μήνυμα που γράψαμε, επιλέγουμε κάποια κρυπτογράφηση AES στον τομέα **Encryption Option**. Οι διαθέσιμες επιλογές παρουσιάζονται στην εικόνα 5.4.



Εικόνα 5.4: Επιλογές κρυπτογράφησης κειμένου ή αρχείου.

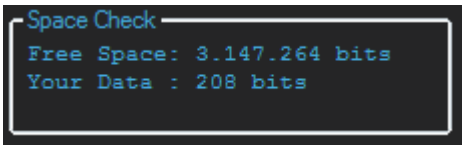
Η επιλογή **No Encryption** δεν εφαρμόζει κρυπτογράφηση και έχει συμβατότητα με μηνύματα που έχουν μόνο αγγλικούς χαρακτήρες. Αντιθέτως με την επιλογή **No Encryption Greek** υπάρχει υποστήριξη για όλους τους χαρακτήρες (ελληνικούς, κινέζικους, αραβικούς και άλλους) και φυσικά ούτε αυτή η επιλογή εφαρμόζει κρυπτογράφηση στο μήνυμα. Η επιλογή αυτή κωδικοποιεί το μήνυμα με Base64 για να κρατήσει αναλλοίωτο τον πχ ελληνικό χαρακτήρα, αυτό έχει ως αποτέλεσμα την αύξηση του χώρου που καταλαμβάνει το μήνυμα μέσα στην εικόνα. Οι τρεις επιλογές AES εφαρμόζουν την ανάλογη κρυπτογράφηση στο μήνυμα, με ασφαλέστερη εξ αυτών την **AES 256**. Μόλις επιλεγεί μια AES κρυπτογράφηση εμφανίζεται στην εφαρμογή ο τομέας **Password Protection** (εικόνα 5.5) με προεπιλεγμένο κλειδί

κρυπτογράφησης το «123456». Προτείνεται η αλλαγή του κλειδιού αυτού για επιπλέον ασφάλεια.



Εικόνα 5.5: Τομέας Password Protection, εμφανίζεται όταν επιλεγεί κρυπτογράφηση AES.

Ανά πάσα στιγμή στον τομέα **Space Check** (εικόνα 5.6) βλέπουμε πόσο χώρο καταλαμβάνει το μήνυμά μας (**Your Data**) αλλά και τον διαθέσιμο χώρο στην εικόνα (**Free Space**). Η προεπιλεγμένη μονάδα μέτρησης είναι τα δυαδικά ψηφία (bits) και όπως θα δούμε στο κεφάλαιο 5.4 υπάρχει η δυνατότητα επιλογής και άλλων μονάδων μέτρησης.



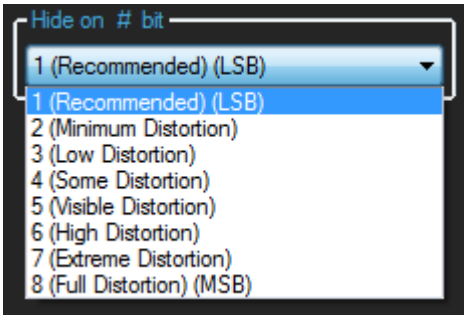
Εικόνα 5.6: Τομέας Space Check για τον έλεγχο μεγέθους του μηνύματος στην εικόνα.

Στον τομέα **Hide on # bit** (εικόνα 5.7) εμφανίζονται οι επιλογές της θέσης του μηνύματος μέσα στην εικόνα. Ανάλογα με την θέση που θα επιλέξουμε θα δημιουργήσουμε και την ανάλογη παραμόρφωση στην εικόνα. Προτεινόμενη επιλογή είναι η θέση 1 με μηδενική παραμόρφωση.

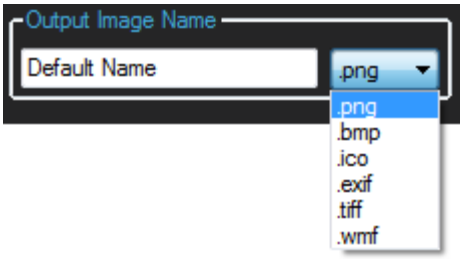
Παραδείγματα παραμόρφωσης εικόνας στο κεφάλαιο 5.5.

Στον τομέα **Output Image Name** (εικόνα 5.8) εισάγουμε το όνομα που θα έχει η στεγανογραφημένη εικόνα.

Στην συνέχεια πατάμε το κουμπί Encrypt για να ξεκινήσει η στεγανογράφηση της οποίας η πρόοδος φαίνεται από μια μπάρα προόδου. Μόλις τελειώσει η διαδικασία αυτή, θα εμφανιστεί η στεγανογραφημένη εικόνα στον τομέα **Image After** (εικόνα 5.9).

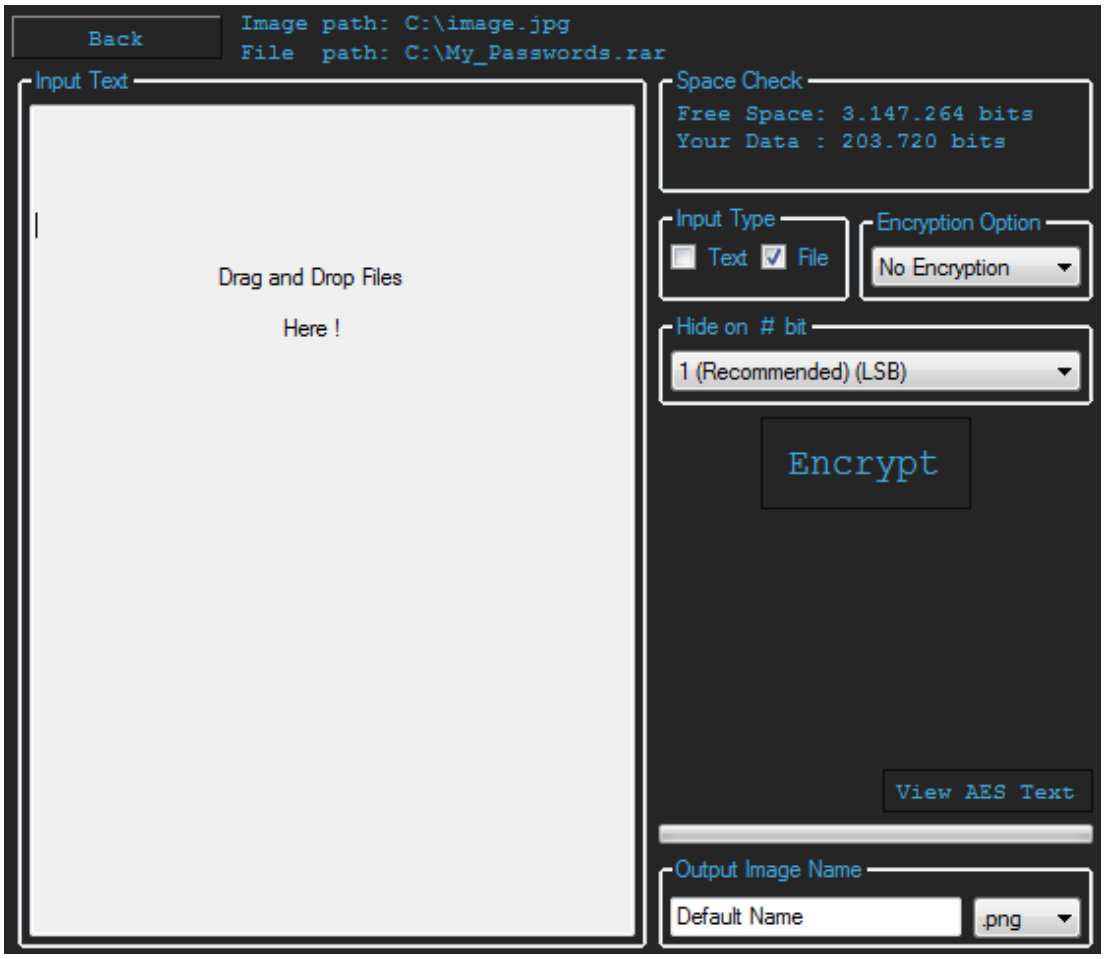


Εικόνα 5.7: Τομέας Hide on # bit, επιλογές παραμόρφωσης εικόνας.



Εικόνα 5.8: Επιλογή ονόματος στεγανογραφημένης εικόνας.

Εικόνα 5.9: Τομέας Image After. Στεγανογραφημένη εικόνα με πληροφορία στην θέση 8 (πλήρης παραμόρφωση)



Εικόνα 5.10: Στεγανογράφηση του αρχείου My_Passwords.rar στην εικόνα image.jpg.

Για να αποθηκεύσουμε την τελική εικόνα απλά την τραβάμε από τον τομέα **Image After** και την αφήνουμε στον επιθυμητό φάκελο.

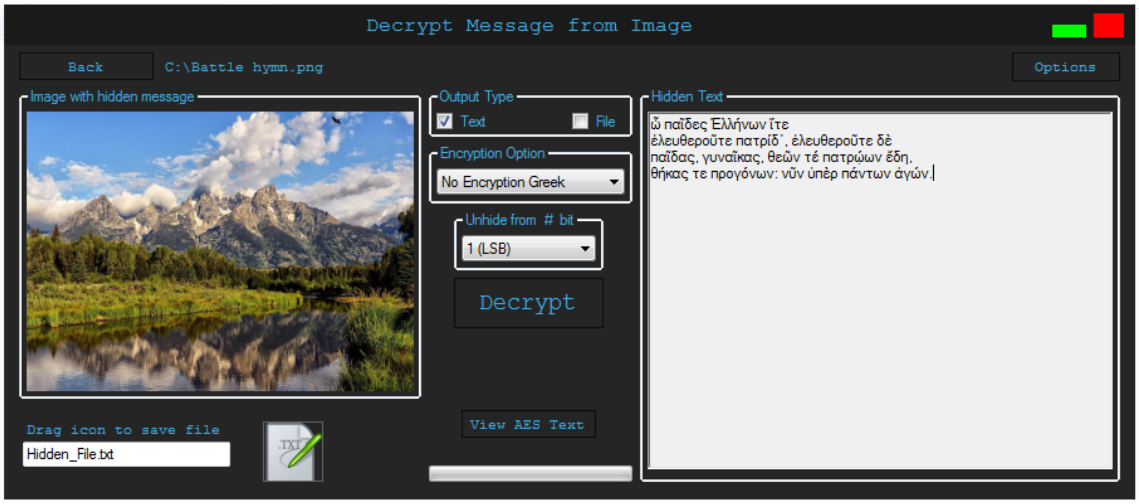
Για να στεγανογραφήσουμε ένα αρχείο (πχ .RAR, .mp3) σε μια εικόνα θα πρέπει να επιλέξουμε **File** στον τομέα **Input Type** και να τραβήξουμε το επιθυμητό αρχείο μέσα στον τομέα **Input Text** ο οποίος θα έχει αλλάξει και θα έχει την μορφή της εικόνας 5.10 .

Όταν επιλέγουμε αρχεία πρέπει να δώσουμε προσοχή στον χώρο που καταλαμβάνει το αρχείο και τον διαθέσιμο χώρο της εικόνας.

5.3 Αποστεγανογράφηση

Για να ανασύρουμε κρυμμένη πληροφορία από μια στεγανογραφημένη εικόνα, αρχικά την εισάγουμε στην αρχική φόρμα της εφαρμογής και επιλέγουμε **Decrypt**. Στην εικόνα 5.11 παρουσιάζεται η μορφή της φόρμας αποστεγανογράφησης.

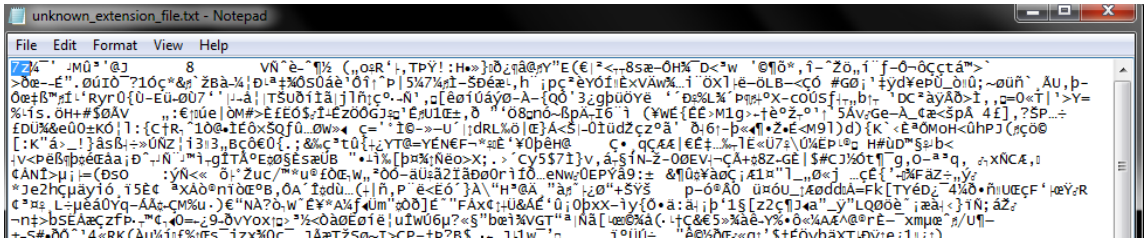
Για να είναι επιτυχημένη η αποστεγανογράφηση πρέπει να γνωρίζουμε εάν χρησιμοποιήθηκε κάποια κρυπτογράφηση και, αν ναι, ποιο είναι το κλειδί της. Αν δεν γνωρίζουμε σε ποια θέση της εικόνας είναι κρυμμένη η πληροφορία μπορούμε απλά να δοκιμάσουμε αποστεγανογράφηση και στις 8 θέσεις (τομέας **Unhide from # bit**). Για να ξεκινήσει η αποστεγανογράφηση πατάμε το κουμπί Decrypt και παρακολουθούμε την πρόοδο της διαδικασίας στην μπάρα προόδου. Μόλις η διαδικασία τελειώσει το κρυμμένο μήνυμα θα εμφανιστεί στον τομέα **Hidden Text**. Υπάρχει η επιλογή για αποθήκευση του μηνύματος σε μορφή κειμένου (.txt) όπως φαίνεται στα κάτω αριστερά της εικόνας 5.11. Μόλις εισάγουμε το όνομα της επιθυμίας μας και την κατάληξη του αρχείου κειμένου, τραβάμε το **εικονίδιο .TXT** (πράσινο μολύβι) στον φάκελο που θέλουμε.



Εικόνα 5.11: Αποστεγανογράφηση εικόνας με περιεχόμενο ελληνικών χαρακτήρων.

Στην περίπτωση που μέσα στην στεγανογραφημένη εικόνα βρίσκεται ένα αρχείο, επιλέγουμε **File** στον τομέα **Output Type** και ονομάζουμε το εξαγόμενο αρχείο με το αντίστοιχο όνομα (πχ music.mp3, My_Password.rar). Εάν δεν γνωρίζουμε τι κατάληξη

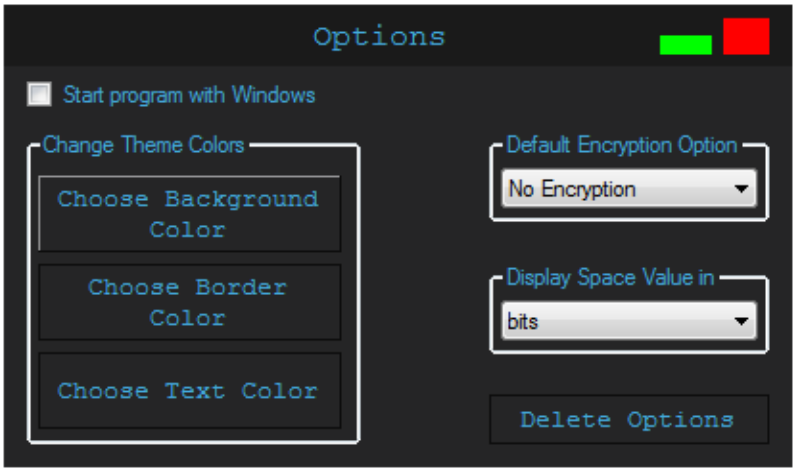
έχει το κρυμμένο αρχείο το αποθηκεύουμε σαν .TXT και το ανοίγουμε με έναν κειμενογράφο. Στην αρχή του κειμένου που εμφανίζεται, αναγράφεται η κατάληξη του αρχείου όπως φαίνεται στην εικόνα 5.12. Στο παράδειγμα αυτό, η κατάληξη του αρχείου είναι .7z όπως υποδηλώνουν οι δύο πρώτοι χαρακτήρες.



Εικόνα 5.12: Εύρεση κατάληξης κρυμμένου αρχείου. Το αρχείο είναι .7z.

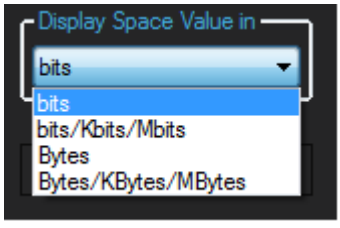
5.4 Παραμετροποίηση επιλογών

Στις φόρμες στεγανογράφησης και αποστεγανογράφησης υπάρχει ένα κουμπί με όνομα **Options** το οποίο ανοίγει την φόρμα παραμετροποίησης των επιλογών. Στην εικόνα 5.13 φαίνεται η μορφή της φόρμας αυτής.



Εικόνα 5.13: Φόρμα παραμετροποίησης επιλογών.

Πάνω αριστερά παρατηρούμε το κουτί **Start program with Windows** το οποίο όταν είναι επιλεγμένο, η εφαρμογή θα ξεκινάει αυτόματα με την εκκίνηση του συστήματος. Στον τομέα **Default Encryption Option** επιλέγουμε ποια θα είναι η προεπιλεγμένη κρυπτογράφηση με την οποία θα ξεκινάει το πρόγραμμα. Στον τομέα **Display Space Value in** επιλέγουμε την μονάδα στην οποία θα εμφανίζεται ο διαθέσιμος χώρος εικόνας και ο χώρος που καταλαμβάνουν οι πληροφορίες μας. Στην εικόνα 5.14 εμφανίζονται οι επιλογές του τομέα αυτού.



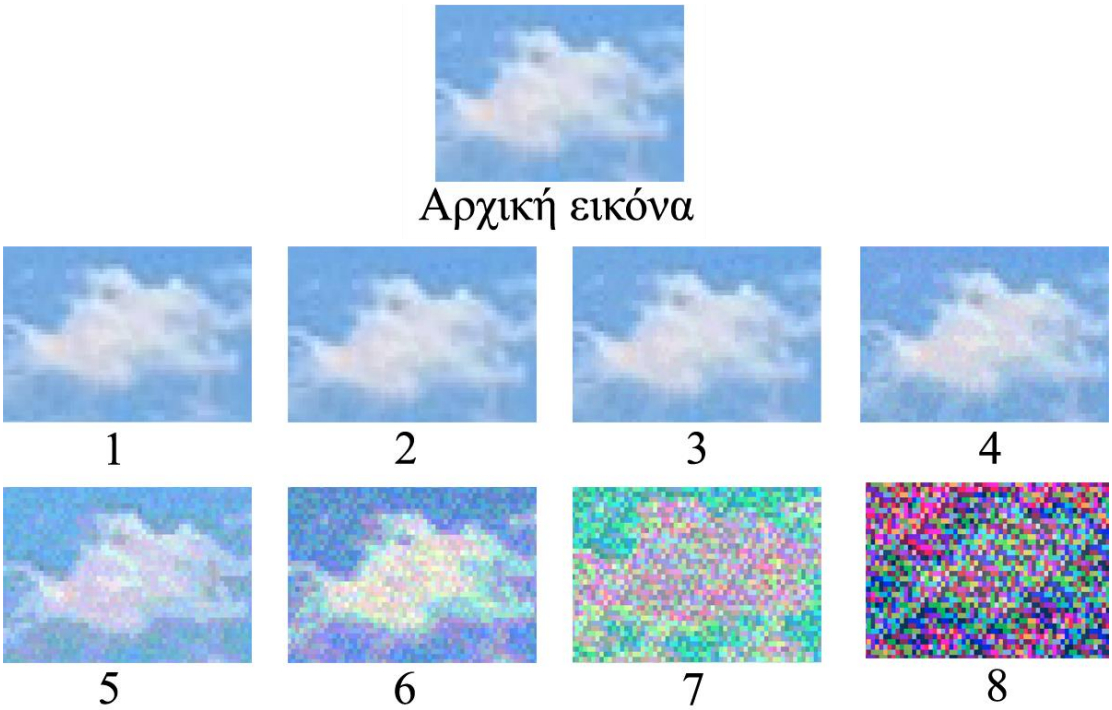
Εικόνα 5.14: Οι διαθέσιμες επιλογές μονάδος μέτρησης.

Στον τομέα **Change Theme Colors** ο χρήστης έχει την δυνατότητα να αλλάζει τα χρώματα των φορμών με τα χρώματα της αρεσκείας του. Κάτω δεξιά στην φόρμα υπάρχει το κουμπί **Delete Options** το οποίο όταν πατηθεί θα διαγράψει τυχών παραμετροποιήσεις που έχουμε κάνει.

5.5 Παραμόρφωση εικόνας

Μεγεθύνουμε κατά 1500% ένα σημείο της αρχικής εικόνας που χρησιμοποιήθηκε για στεγανογραφία (εικόνα 5.15). Το σημείο είναι το σύννεφο μέσα στο κόκκινο τετράγωνο.

Εικόνα 5.15: Σημείο μεγέθυνσης, αρχική εικόνα στεγανογράφησης.



Εικόνα 5.16: Παράδειγμα αλλοίωσης εικόνας ανάλογα με την θέση στεγανογραφίας.

Στην εικόνα 5.16 παρουσιάζονται οι διαφορές στην αλλοίωση της εικόνας ανάλογα με την θέση στην οποία έχει στεγανογραφηθεί η πληροφορία.

Όπως παρατηρούμε η παραμόρφωση είναι σταδιακή με κάποιες από τις πρώτες θέσεις να μην έχουν αισθητές παραμορφώσεις συγκριτικά με την αρχική εικόνα.

Στο κεφάλαιο 2.3 πληροφορηθήκαμε για την μέθοδο στεγανογράφησης στο λιγότερο σημαντικό ψηφίο (LSB) της εικόνας. Στο παράδειγμα της εικόνας 5.16 η θέση που κατέχει το λιγότερο σημαντικό ψηφίο είναι η $1^{η}$. Η $8^{η}$ θέση αποτελεί το περισσότερο σημαντικό ψηφίο (MSB) της εικόνας. Όσο σημαντικότερο είναι το ψηφίο στο οποίο στεγανογραφούμε πληροφορία τόσο μεγαλύτερη είναι η παραμόρφωση που θα υποστεί η τελική εικόνα.

6. Συμπεράσματα και μελλοντικές επεκτάσεις.

6.1 Συμπεράσματα

Διαβάζοντας όλα τα παραπάνω αντιλαμβανόμαστε ότι όπως υπάρχουν ανάγκες έτσι υπάρχουν και λύσεις που τις ικανοποιούν. Η ανάγκη για μια ασφαλή και παράλληλα κρυφή επικοινωνία μπορεί να ικανοποιηθεί από την παραπάνω εφαρμογή. Εάν ο αποστολέας-χρήστης ενεργοποιήσει την κρυπτογράφηση της εφαρμογής και ο παραλήπτης εφαρμόσει αποκρυπτογράφηση τότε η επικοινωνία ικανοποιεί τα κριτήρια ώστε να χαρακτηριστεί από άκρη σε άκρη κρυπτογραφημένη (End to End Encryption) [22]. Αυτός ο χαρακτηρισμός λείπει από πολλές εφαρμογές επικοινωνίας αυτή την εποχή. Ένας από του κυριότερους λόγους όπου εταιρείες αποφασίζουν να μην ασφαλίσουν τις επικοινωνίες των χρηστών τους με κρυπτογράφηση από άκρη σε άκρη, είναι η επιβεβλημένη ανάγκη για την καταγραφή των πληροφοριών που ανταλλάσσουν οι χρήστες. Η καταγραφή των συνομιλιών των χρηστών έχει διαφορετική έννοια ανάλογα με το ποιος την υποκινεί. Εάν την υποκινεί μια εταιρία ονομάζεται παρακολούθηση [23], ενώ εάν υποκινείται από κυβερνήσεις ονομάζεται κατασκοπεία [24]. Όταν το κανάλι επικοινωνίας δεν είναι κρυπτογραφημένο από άκρο σε άκρο δεν σημαίνει απαραίτητα ότι και η επικοινωνία θα είναι έρμαια στα χέρια κάποιου κακόβουλου εισβολέα (black hat hacker). Συνήθως οι εταιρίες κρυπτογραφούν το κανάλι επικοινωνίας από τον αποστολέα στον διακομιστή (server) τους. Μόλις τα δεδομένα του αποστολέα φτάσουν στον διακομιστή αποκρυπτογραφούνται και τα αποθηκεύουν για εταιρική ή κρατική χρήση. Στην συνέχεια τα δεδομένα αυτά κρυπτογραφούνται εκ νέου και φεύγουν από τον διακομιστή προς τον παραλήπτη που είχε επιλέξει ο αποστολέας εξ αρχής. Με αυτό τον τρόπο εάν ο κακόβουλος εισβολέας θέλει να αποκτήσει πρόσβαση στην πληροφορία που ανταλλάχτηκε θα πρέπει να πάρει τον έλεγχο μολύνοντας είτε τον υπολογιστή του αποστολέα είτε του παραλήπτη είτε τον διακομιστή της εταιρίας. Η κατάσταση θα ήταν χειρότερη εάν η πληροφορία που ανταλλάχτηκε ήταν ακρυπτογράφητη από άκρο σε άκρο καθώς πρόσβαση στην πληροφορία αποκτά οποιοσδήποτε βρίσκεται στο ίδιο δίκτυο με τον αποστολέα ή τον παραλήπτη. Κλείνοντας αξίζει να σημειωθεί ότι ούτε η από άκρη σε άκρη κρυπτογράφηση δεν μπορεί να εγγυηθεί την απόλυτη ασφάλεια των δεδομένων καθώς υπάρχει πάντα ο περιορισμός της εφαρμογής που εφαρμόζει την κρυπτογράφηση αυτή. Στην περίπτωση που η εφαρμογή είναι υλοποιημένη με την βέλτιστη ασφάλεια, τα δεδομένα και η ασφάλειά τους εξαρτώνται από το κατά πόσο είναι ευάλωτος ο χρήστης σε επιθέσεις τύπου κοινωνικής μηχανικής (Social Engineering Attacks) [25].

6.2 Μελλοντικές επεκτάσεις

Η χρήση συμμετρικής κρυπτογράφησης (AES) σε αυτή την εφαρμογή μπορεί να είναι αποδοτική από άποψη ασφαλείας αλλά είναι δύσχρηστη η ανταλλαγή του κλειδιού αποκρυπτογράφησης. Ως μελλοντική βελτίωση της εφαρμογής θα μπορούσε να θεωρηθεί ένας μηχανισμός ανταλλαγής του κλειδιού κρυπτογράφησης. Για αυτό τον σκοπό μπορεί να χρησιμοποιηθεί κάποιος ασύμμετρος αλγόριθμος κρυπτογραφίας (πχ RSA) ο οποίος θα κρυπτογραφεί μόνο το κλειδί του συμμετρικού αλγορίθμου και όχι τα δεδομένα. Η αντικατάσταση του συμμετρικού αλγορίθμου από τον ασύμμετρο θα

επέφερε μείωση στην απόδοση της εφαρμογής καθώς τα κρυπτογραφημένα δεδομένα έχουν πολύ μεγαλύτερο όγκο και παράλληλα η επεξεργαστική ισχύς που απαιτείται είναι πολύ μεγαλύτερη. Όσο αφορά τον τομέα της στεγανογραφίας θα μπορούσε να χρησιμοποιηθεί μια μέθοδος διασποράς της πληροφορίας, που θα στεγανογραφεί τα δεδομένα σε τυχαίες θέσεις εικονοστοιχείων και να κρυπτογραφεί τις θέσεις αυτές σε ένα σημείο της εικόνας με το κλειδί που εισάγει ο χρήστης. Επίσης στην παρούσα εφαρμογή όπως είδαμε ο χρήστης έχει την δυνατότητα να επιλέξει σε ποια θέση εικονοστοιχείου θα στενογραφήσει τα δεδομένα του. Βελτίωση θα μπορούσε να θεωρηθεί η δυνατότητα επιλογής στεγανογράφησης σε περισσότερες θέσεις εικονοστοιχείων ταυτόχρονα (πχ θέση από 2 έως 4). Οι θέσεις στις οποίες στεγανογραφήθηκε η πληροφορία με κάποιο τρόπο θα πρέπει να είναι γνωστές για να μπορεί να εφαρμοστεί αποστεγανογράφηση και αποκρυπτογράφηση. Οι θέσεις αυτές θα μπορούσαν να αποστέλλονται παράλληλα με την εικόνα ή να καταχωρούνται σε έναν τομέα μέσα στην εικόνα.

Βιβλιογραφία

Βιβλία

[1] Πομπόρτσης Α., Παπαδημητρίου Γ., "Ασφάλεια Δικτύων Υπολογιστών", εκδόσεις Τζιόλα, 2003.

[2] Bruce Schneier, "Applied Cryptography", Wiley, 1996.

[3] William Stallings, "Cryptography and Network Security", 4th edition, Pearson Prentice Hall, 2006.

Άρθρα

[4] Abbas Cheddad, Joan Condell, Kevin Curran, Paul Mc Kevitt, "Digital image steganography: Survey and analysis of current methods", Signal Processing 2010, pp. 727–752.

[5] Krzysztof Szczypiorski, "Steganography in TCP/IP Networks", Warsaw University of Technology, 2003.

[6] Jayaram P., Ranganatha H. R., Anupama H. S., "INFORMATION HIDING USING AUDIO STEGANOGRAPHY – A SURVEY", August 2011, R V College of Engineering.

[7] Deepali Ghanwat, "SPREAD SPECTRUM BASED AUDIO STEGANOGRAPHY IN TRANSFORMATION DOMAIN", Global Journal of Advanced Engineering Technologies, Vol2, Issue4-2013.

[8] Daniel Gruhl, Anthony Lu, Walter Bender, "Echo Hiding", Massachusetts Institute of Technology Media Laboratory, 1996.

[9] K. Gopalan, S. Wenndt, "Audio steganography for covert data transmission by imperceptible tone insertion", Department of Engineering, Purdue University Calumet.

[10] Monika Agarwal, "Text steganographic approaches: A comparison", International Journal of Network Security & Its Applications (IJNSA), Vol. 5, No.1, January 2013.

[11] Prem Singh, Rajat Chaudhary, Ambika Agarwal, "A Novel Approach of Text Steganography based on null spaces", IOSR Journal of Computer Engineering (IOSRJCE) ISSN: 2278-0661 Vol. 3, Issue 4 (July-Aug. 2012), PP 11-17.

[12] Md. Khairullah, "A Novel Text Steganography System in Cricket Match Scorecard", International Journal of Computer Applications (0975 –8887) Vol. 21– No.9, May 2011.

Ιστότοποι

[13] Wikipedia https://el.wikipedia.org/wiki/Στεγανογραφία

[14] An Overview of Steganography for the Computer Forensics Examiner (Gary C. Kessler 2004) http://www.garykessler.net/library/fsc_stego.html#intro και Steganography: Hiding Data Within Data (Gary C. Kessler 2001) http://www.garykessler.net/library/steganography.html

[15] Online Text to Text Steganography http://www.spammimic.com

[16] Wikipedia https://en.wikipedia.org/wiki/Parity_bit

[17] Εφαρμογή στεγανογραφίας κενού διαστήματος Snow. (Matthew Kwan) http://www.darkside.com.au/snow/

[18] Wikipedia https://el.wikipedia.org/wiki/Κρυπτογραφία

[19] Haystack https://www.grc.com/haystack.htm

[20] Microsoft AES cryptography https://msdn.microsoft.com/en-us/library/system.security.cryptography.aesmanaged.aspx

[21] Webopedia http://www.webopedia.com/TERM/S/steganography.html

[22] Wikipedia E2EE https://en.wikipedia.org/wiki/End-to-end_encryption

[23] Wikipedia Παρακολούθηση https://en.wikipedia.org/wiki/Surveillance

[24] Wikipedia Κατασκοπεία https://en.wikipedia.org/wiki/Espionage

[25] Wikipedia Κοινωνική μηχανική https://el.wikipedia.org/wiki/Κοινωνική_μηχανική