ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ

ΚΡΗΤΗΣ

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ

Στεγανογραφία σε video. Υλοποίηση μεθόδου απόκρυψης πληροφορίας και player plug-in για την επανεμφάνιση της.

Σπουδάστρια: Σεβαστάκη Ιωάννα
Επιβλέπων καθηγητής : Χάρης Μανιφάρας

Ηράκλειο 2010

Περιεχόμενα

1.ΕΙΣΑΓΩΓΗ

Ζούμε σε μία εποχή όπου τα ψηφιακά μέσα μπορούν να αντιγραφούν εύκολα, με χαμηλό κόστος, χωρίς να προκαλέσουν την οποιαδήποτε απώλεια ή αλλοίωση της ποιότητας και της μορφής τους. Η ανάγκη για τη δημιουργία ασφαλών διαύλων ανεπαίσθητων, για το απόρρητο των επικοινωνιών και των παράνομων ψηφιακών δεδομένων δημιουργεί την ανάγκη αξιόπιστων και ασφαλών μεθόδων για την απόκρυψη δεδομένων, την προστασία και τον έλεγχο τους καθώς και τη διασφάλιση των δικαιωμάτων πνευματικής ιδιοκτησίας.

Η στεγανογραφία μπορεί να χρησιμοποιηθεί οποιαδήποτε στιγμή θέλουμε να αποκρύψουμε δεδομένα. Αυτό μπορεί να αναφέρεται είτε σε εταιρική κατασκοπεία για την αποστολή εμπορικών μυστικών ή σε μη εμπορικό τομέα για την απόκρυψη πληροφοριών που κάποιος θέλει να κρατήσει ιδιωτικές. Οι τρομοκράτες μπορούν επίσης να τη χρησιμοποιήσουν για να τηρούν το απόρρητο των επικοινωνιών και να συντονίζουν τις επιθέσεις τους.

Σε έναν ιδανικό κόσμο όλοι θα ήμασταν σε θέση να στέλναμε εύκολα κρυπτογραφημένο κείμενό μας μέσω e-mail ή αρχεία ο ένας στον άλλο χωρίς συνέπειες. Παρόλα αυτά υπάρχουν συχνά περιπτώσεις κατά τις οποίες αυτό δεν είναι δυνατό, είτε επειδή εργαζόμαστε σε μια επιχείρηση που δεν επιτρέπει την ανταλλαγή κρυπτογραφημένων μηνυμάτων μέσω e-mail, είτε γιατί η κυβέρνηση δεν εγκρίνει την χρήση της κρυπτογραφημένης επικοινωνίας.

Είναι μία αρχαία τεχνική η οποία έχει καταφέρει με τη βοήθεια του διαδικτύου και της τεχνολογίας να αναπτυχθεί και να διαδοθεί ευρέως. Πολλά δημόσια στεγανογραφικά λογισμικά, όπως η S-Εργαλεία, EZStego και Steganos εφαρμόζουν την τεχνική αυτή. Παράλληλα όμως γίνονται εντεταμένες προσπάθειες για προστασία της μυστικότητας μας, άλλα και της οποιαδήποτε επικοινωνίας. Βέβαια παρόλα τα μέτρα που έχουν ληφθεί δεν περιορίζεται η εγκληματική χρήση.

2.ΣΤΕΓΑΝΟΓΡΑΦΙΑ

2.1 ΟΡΙΣΜΟΣ

Στεγανογραφία είναι η τέχνη του να κρύβονται ιδιωτικές ή ευαίσθητες πληροφορίες εντός κάτι που τίποτα δεν φαίνεται να είναι έξω από το συνηθισμένο. Στη σύγχρονη έννοια του, ο όρος αναφέρεται σε πληροφορίες ή αρχεία που είναι κρυμμένα μέσα σε μία εικόνα, βίντεο αρχεία ή αρχεία ήχου.

Προέρχεται από τις ελληνικές λέξεις

- Στεγανό = καλυμμένο ή αλλιώς σκεπασμένο
- Γραφή = γραφή, κείμενο ή ζωγραφιά

Ένα στεγανογραφικό σύστημα περιλαμβάνει δύο μέρη: τον **αποστολέα**, ο οποίος ενσωματώνει το μυστικό μήνυμα στο αντικείμενο κάλυψη και τον **δέκτη**, ο οποίος το αποσπά. Το μέσο που χρησιμοποιείται για συγκεκαλυμμένη επικοινωνία είναι το ψηφιακό βίντεο. Ο αποστολέας λαμβάνει την ακολουθία βίντεο, η οποία εκπροσωπεί το **κάλυψη-βίντεο** και ενσωματώνει ένα μυστικό μήνυμα, χρησιμοποιώντας ένα κλειδί κ, δημιουργώντας έτσι το στέγο-βίντεο. Στη συνέχεια το στέγο-βίντεο κυκλοφορεί σε δημόσιο κανάλι με το δέκτη. Ο δέκτης το λαμβάνει και εξάγει το μυστικό μήνυμα με τη βοήθεια ενός κλειδιού κ.

- **μέσο μεταφοράς + μήνυμα + στέγο-κλειδί = στέγο-μέσο**

όπου :

- **μέσο μεταφοράς** μπορεί να είναι εικόνα, ήχος, κείμενο, video.
- **μήνυμα** είναι η πληροφορία που θέλουμε να κρύψουμε που μαζί με το μέσο μεταφοράς αποτελούν το στέγο-φορέα. (stego-carrier)
- **στέγο-κλειδί** μία επιπλέον πληροφορία ασφάλειας.

Ουσιαστικά εκμεταλλεύεται την ανθρώπινη αντίληψη, τις ανθρώπινες αισθήσεις οι οποίες δεν εκπαιδεύονται να ψάξουν για αρχεία που έχουν πληροφορίες που κρύβονται από αυτές. Ένα καλό στεγανογραφικό σύστημα πρέπει να εκπληρώνει τις προδιαγραφές που έθεσε η "Αρχή του Kerckhoff" στην κρυπτογραφία: "Η ασφάλεια ενός συστήματος πρέπει να βασίζεται στο δεδομένο ότι ο "εχθρός" έχει πλήρη γνώση των σχεδιαστικών λεπτομερειών και της υλοποίησης ενός στεγανογραφικού συστήματος". Η μόνη πληροφορία που λείπει από τον "εχθρό" και που πρέπει να κρατηθεί μυστική από αυτόν είναι ένας μικρός και εύκολα ανταλλάξιμος τυχαίος αριθμός, το μυστικό κλειδί, χωρίς το οποίο δεν μπορεί να γνωρίζει εάν στο κανάλι επικοινωνίας διενεργείται κρυφή επικοινωνία. Χωρίς γνώση αυτού του κλειδιού θα είναι δύσκολο για ένα τρίτο να το εξάγει ή και να ανιχνεύσει την ύπαρξη του.

3. ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ ΤΗΣ ΣΤΕΓΑΝΟΓΡΑΦΙΑΣ

Σε όλη τη διαδρομή της ιστορίας ο άνθρωπος συνεχώς ανακάλυπτε νέες μεθόδους που του επέτρεπαν να κρύψει κάποια πολύτιμη πληροφορία.

Τα πρώτα σημάδια εμφάνισης της προέρχονται από την εποχή του Ηροδότου, στην αρχαία Ελλάδα, όπου οι Έλληνες επικοινωνούσαν συχνά με το γράψιμο σε πίνακες καλυμμένους με κερί. Έκρυβαν τα μηνύματα τους σε ειδικούς ξύλινους πίνακες και έπειτα τα κάλυπταν με ένα νέο στρώμα κεριού. Στην επιθεώρηση οι πινακίδες φαίνονταν λευκές και αχρησιμοποίητες και με αυτό το τρόπο πέρναγαν κάθε έλεγχο εξασφαλίζοντας ότι το μήνυμα παρέμενε άγνωστο.

Στην αρχαία Ελλάδα, χρησιμοποιούσαν τους κλητήρες και το ξύρισμα του κεφαλιού τους προκειμένου να γράψουν τα διάφορα μηνύματα. Όταν η τρίχα του σκλάβου μεγάλωνε μετά από αρκετό καιρό για να κρύψει το μήνυμα, ο αγγελιοφόρος στελνόταν και ο παραλήπτης ξύριζε τον αγγελιοφόρο για να δεί το μυστικό μήνυμα.

Κατά την διάρκεια του Παγκόσμιου πολέμου η αόρατη μελάνη είχε χρησιμοποιηθεί για να γράψουν πληροφορίες σχετικά, σε κομμάτια χαρτιού, έτσι ώστε το χαρτί να φαινόταν από το μέσο άνθρωπο όπως ένα κενό χαρτί. Η προέλευση αυτών των μελανιών είναι το γάλα, διάφορα φρούτα, το ξίδι και τα ούρα. Τα υλικά αυτά όταν θερμαίνονται σκουραίνουν και τότε μόνο γίνονται ορατές από το ανθρώπινο μάτι. Με την ανάπτυξη της τεχνολογίας αναπτύχθηκαν νέα περιπλοκότερα μελάνια ώστε να αντιδρούν μόνο σε συγκεκριμένες χημικές ουσίες.

Άλλη μέθοδος είναι αυτή των "Null ciphers"□μη κρυπτογραφημένων μηνυμάτων. Υπήρχε τότε, όπως και σήμερα, η τεχνική της ανίχνευσης υπόπτων μηνυμάτων μέσω κάποιων ειδικών φίλτρων μιας αυτοματοποιημένης διαδικασίας. Ωστόσο, τα αθώα μηνύματα πέρναγαν ανενόχλητα. Το μόνο λοιπόν που είχε να κάνει κάποιος που ήθελε να στείλει κάποια κρυφή πληροφορία ήταν να την κάνει να φαίνεται αθώα. Έτσι έγραφε ένα τυχαίο κείμενο στο οποίο η πληροφορία βρισκόταν σε κάθε δεύτερο, για παράδειγμα, γράμμα των λέξεων του κειμένου. Ένα παράδειγμα ενός μηνύματος nullcipher τέτοιας μορφής είναι το εξής :
A**p**parently n**e**utral's p**r**otest i**s** t**h**oroughly d**i**scounted
And i**g**nored. I**s**man h**a**rd h**i**t. B**l**ockade i**s**sue a**f**fects
Pretext f**o**r e**m**bargo o**n** b**y**-products, e**j**ecting s**u**ets a**n**d v**e**getable
o**i**ls.
Διαβάζοντας το δεύτερο γράμμα από κάθε λέξη, το μήνυμα που προκύπτει είναι :
Pershing sails from NY June 1.

Καθώς, όμως η τεχνολογία συνέχισε να αναπτύσσεται, βρέθηκαν τρόποι διακίνησης μεγαλύτερου όγκου πληροφορίας με ακόμα πιο αόρατο τρόπο. Οι Γερμανοί ανέπτυξαν τη τεχνολογία των μικροτελειών (microdots). Οι μικροτελείες είναι φωτογραφίες υψηλής ανάλυσης και ασήμαντου μεγέθους τελείες. Αυτό το σύστημα χρησιμοποιήθηκε από Γερμανούς κατασκόπους κατά τον δεύτερο παγκόσμιο πόλεμο.

4.ΣΤΕΓΑΝΟΓΡΑΦΙΑ

4.1 ΣΤΕΓΑΝΟΓΡΑΦΙΑ – ΚΡΥΠΤΟΓΡΑΦΙΑ

Η στεγανογραφία συχνά συγχέεται με την κρυπτογραφία, διότι και οι δύο είναι παρόμοιες στον τρόπο που χρησιμοποιούνται για την προστασία των σημαντικών πληροφοριών. Η διαφορά μεταξύ τους είναι ότι η στενογραφία **αποκρύπτει την ύπαρξη του μηνύματος,** ενώ η κρυπτογραφία **μετασχηματίζει το μήνυμα ώστε να το καθιστά ακατανόητο σε οποιονδήποτε τρίτο.**

Εάν ένα άτομο ή άτομα δεν γνωρίζουν ότι υπάρχει κρυφή πληροφορία δεν θα επιχειρήσουν και να αποκρυπτογραφήσουν τις πληροφορίες. Εάν κάποιος ήθελε να εξετάσει ένα αρχείο με κρυμμένες πληροφορίες θα μπορούσε να τις βρει. Στη χειρότερη περίπτωση θα μπορούσε να καταλάβει ότι αυτές υπάρχουν έστω και αν δεν τις έβλεπε. Εάν οι κρυμμένες πληροφορίες είναι κρυπτογραφημένες τότε σίγουρα θα φτάσει μέχρι αυτό το σημείο και θα σταματήσει. Ωστόσο εάν δεν είναι κρυπτογραφημένες τότε θα είναι σε θέση να εξετάσει όλο το "κρυμμένο" μήνυμα. Για το λόγο αυτό δεν θα πρέπει να θεωρούμε τη στεγανογραφία σαν αντικαταστάτη της κρυπτογραφίας αλλά σαν συμπλήρωμά της.

Στο σύστημα της κρυπτογραφίας, οι πληροφορίες κωδικοποιούνται με ένα κλειδί και το πρόσωπο που έχει το κλειδί μπορεί να το αποκρυπτογραφήσει και να διαβάσει τις πληροφορίες, στις οποίες δεν θα έχει κανείς άλλος πρόσβαση. Η στεγανογραφία σε σχέση με την κρυπτογραφία προσθέτει ένα ακόμα επίπεδο ασφαλείας για τα ευαίσθητα αρχεία μας. Αν κάποιος τρίτος αποκτήσει πρόσβαση σε ένα αρχείο προστατευμένο με κωδικό, μπορεί με την χρήση των κατάλληλων εργαλείων (password cracker tools) να ανακαλύψει τον κωδικό προστασίας. (πάντως εάν χρησιμοποιείτε περίπλοκους κωδικούς η διαδικασία του cracking μπορεί να απαιτήσει πολλά χρόνια για να ολοκληρωθεί).

Αυτό που κάνει η στενογραφία ουσιαστικά είναι

- Ενσωμάτωση μυστικής πληροφορίας σε ένα αρχικό αντικείμενο(cover object)
- Στεγανογραφικό κλειδί (stego key)
- Στεγανογραφημένο αντικείμενο (stego object)
- Στεγανογραφική χωρητικότητα(cover capacity)

Οι λόγοι που οδήγησαν στην ανάπτυξη τέτοιων εφαρμογών είναι:

- Να μπορούν να μεταδίδουν πληροφορίες χωρίς να γίνονται αντιληπτοί από τρίτους.
- Αν γίνουν αντιληπτοί να μην υπάρχει η δυνατότητα να ερμηνευθεί το μήνυμα που μετέδωσαν.
- Αν τελικά υποκλαπεί το μήνυμα, ο παραβάτης να υφίστανται τις συνέπειες του νόμου .
- Αν αλλοιωθούν τα δεδομένα, να υπάρχει η δυνατότητα επαναφοράς στην αρχική τους μορφή.
- Αν διεκδικηθεί η ιδιοκτησία τους, να μπορούν να αποδείξουν την κυριότητα τους.

Η ανθεκτικότητα, η αντοχή και η ευρωστία είναι τρία χαρακτηριστικά που έχουν επιπτώσεις στη στεγανογραφία και τη χρησιμότητα της.

- Η ανθεκτικότητα αναφέρεται στην ικανότητα των ενσωματωμένων στοιχείων να παραμείνουν ανέπαφα, εάν υφίστανται μετασχηματισμός στη στέγο-εικόνα.
- Η ευρωστία είναι ζωτικής σημασίας για προστασία της πνευματικής ιδιοκτησίας, επειδή κάποιοι θα προσπαθήσουν να φιλτράρουν και να καταστρέψουν κάθε πληροφορία ενσωματωμένη σε εικόνες. Το μόνο μειονέκτημα που προσφέρει είναι μία υψηλή επιβάρυνση για μικρή πληροφορία και αν η μέθοδος αποκαλυφθεί δεν παρέχεται καμία προστασία.
- Πέρα από ευρωστία της καταστροφής, η παραποίηση αντοχής αναφέρεται στην δυσκολία για έναν εισβολέα να μεταβάλει ή να σφυρηλατήσει ένα μήνυμα τη στιγμή που θα έχει ενσωματωθεί σε μία στέγο –εικόνα. Όπως ένα πειρατικό αντικαθιστά ένα σήμα πνευματικής ιδιοκτησίας με μία διεκδίκηση της νόμιμης ιδιοκτησίας.

Ένα διάσημο πρότυπο για τη στεγανογραφία είναι το πρόβλημα των φυλακισμένων Simmons. Η Alice και ο Bob είναι κλειδωμένοι σε διαφορετικά κελιά αλλά έχουν την άδεια για να επικοινωνούν κάτω από το άγρυπνο μάτι της Eve, που είναι ο φύλακας των φυλακών.
Η Eve ελέγχει την επικοινωνία μεταξύ της Alice και Bob και είναι διατεθειμένη να διακόψει ορισμένες μορφές επικοινωνίας. Στην ιδανική περίπτωση, η Eve θα επιθεωρεί κάθε μήνυμα και θα αποφασίζει αν η επικοινωνία επιτρέπεται ή όχι. Έτσι, κρυπτογραφημένα δεδομένα δεν επιτρέπεται από την Eve να περάσουν γιατί δεν μπορεί να αποκωδικοποιήσει το περιεχόμενο τους.
Η Alice και ο Bob μοιράζονται ένα μυστικό κλειδί *K* το οποίο χρησιμοποιείται για την ενσωμάτωση και για τη λήψη του μηνύματος.
Σε ένα πραγματικό σενάριο, η Eve μπορεί να ανήκει σε μια εταιρεία η οποία προσπαθεί να διατηρήσει κάποιες μυστικές πληροφορίες και η Alice, που ανήκει σε αυτή την εταιρεία, προσπαθεί να μεταδώσει αυτό το μυστικό στο Bob, ο οποίος είναι εκτός.

Παραδείγματα

Αν παρατηρήσει κανείς με προσοχή θα καταλάβει ότι το παρακάτω είναι ένα κρυπτογραφημένο μήνυμα:

Ξλμθψκλφ λπιρ

Όταν θα μεταθέσω τα γράμματα κατά 8 θέσεις θα πάρω το μήνυμα μου. Έτσι αντί για Χ έγραψα Ξ, αντί για Τ έγραψα Λ κ.τ.λ. (Η φράση είναι "Χτυπήστε Τώρα").

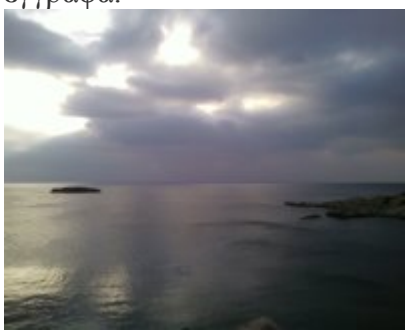
Η στενογραφία, είναι ένας τρόπος να κρύψεις ένα αρχείο, μέσα σε μια κανονική (ψηφιακή) εικόνα. Το ανθρώπινο μάτι δεν μπορεί να διακρίνει διαφορά. Αν γράψεις ένα κείμενο και το κρυπτογραφήσεις και το στείλεις σ' έναν φίλο σου μέσω ηλεκτρονικού ταχυδρομείου, μπορεί κάποιος που θα το δει, να θελήσει να μάθει ποίο είναι το μυστικό μήνυμα που έγραψες και να προσπαθήσει να το αποκρυπτογραφήσει.

Αν όμως γράψουμε ένα ηλεκτρονικό μήνυμα και το κρύψουμε μέσα σε φωτογραφίες, όταν τις στείλουμε και κάποιος τις δει ίσως και μην υποψιαστεί ποτέ ότι μέσα έχουμε κρύψει κάτι.

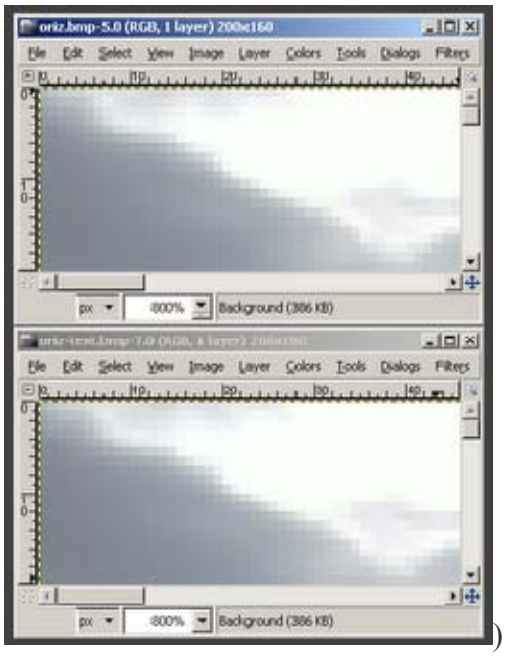
Έχουμε μια κανονική εικόνα

Παρακάτω βλέπετε την ίδια εικόνα, στην οποία κρύψαμε μέσα ένα αρχείο με το ακόλουθο κείμενο:

"Γεια σας. Αύριο συνάντηση στο γνωστό σημείο. Μην ξεχάσετε να φέρετε τα έγγραφα."



Για να συγκρίνουμε τις παραπάνω εικόνες θα τις μεγεθύνουμε κατά 800% προβάλλοντας καλύτερα τις διαφορές .



):

[Στην πραγματικότητα δεν θα είχατε την πρώτη εικόνα για να κάνετε τη σύγκριση...]

Μετά, όταν θελήσει η άλλη πλευρά να διαβάσει το κρυμμένο κείμενο, θα πρέπει πρώτα να δώσει το κλειδί (διότι το κείμενο δεν είναι μόνο κρυμμένο, αλλά και κρυπτογραφημένο)

πηγή πλαίσιο

πηγή πλαίσιο με κρυφό info

συμπιεσμένο πλαίσιο με κρυφό info - πληροφορίες που ήταν πλήρως!

Όσο αφορά το είδος του αρχείου, μπορεί να είναι το οτιδήποτε, π.χ. txt, pdf, κλπ. Μπορεί να είναι όμως και ένα αρχείο εικόνας (ναι, εικόνα κρυμμένη μέσα στην εικόνα). Μπορεί ακόμη και να είναι μέσα στην εικόνα κρυμμένο ένα βίντεο!

έχει εφαρμογή σε οτιδήποτε παρουσιάζει πλεονασμό :

Εικόνα
Ήχος
Video
Δικτυακά πρωτόκολλα
Κείμενα φυσικής γλώσσας

5.ΣΤΕΓΑΝΟΓΡΑΦΙΚΑ ΕΡΓΑΛΕΙΑ

MP3Stego
MP3Stego θα αποκρύψει πληροφορίες σε αρχεία MP3 συμπίεσης κατά τη διάρκεια της διαδικασίας. Τα δεδομένα είναι πρώτα συμπιεσμένα, μετά κρυπτογραφούνται και στη συνέχεια, κρυφά εκχωρούνται στον MP3.

JPHide και JPSeek
JPHIDE και JPSEEK είναι προγράμματα που μας επιτρέπουν να κρύψουμε ένα αρχείο σε μια εικόνα JPEG. Υπάρχουν πολλές εκδόσεις και παρόμοια προγράμματα διατίθενται μέσω του διαδικτύου. Ο σχεδιαστικός στόχος δεν ήταν απλώς να κρύψει ένα αρχείο, αλλά να το κάνει με τέτοιο τρόπο ώστε είναι αδύνατον να αποδείξει κάποιος ότι το αρχείο περιέχει ένα κρυφό μήνυμα. Φυσικά είναι πολύ καλύτερο όταν χρησιμοποιείται ένα πλήθος αρχείων, με πολλές λεπτομέρειες.

BlindSide Κρυπτογραφικός Tool
BlindSide είναι ένα παράδειγμα της τέχνης της στεγανογραφίας - το πέρασμα των μυστικών μηνυμάτων - σε μορφή τέτοια που ένας ύποπτος δεν θα αντιληφθεί το μήνυμα που πέρασε.
Blindside. Το βοηθητικό πρόγραμμα μπορεί να κρύψει ένα αρχείο (ή αρχεία) σε μια εικόνα bitmap των Windows (αρχείο BMP).

GIFShuffle
Το πρόγραμμα **gifshuffle** χρησιμοποιείται για να κρύψουν μηνύματα σε GIF εικόνες.
Αφήνει την εικόνα εμφανώς αμετάβλητη, λειτουργεί με όλες τις εικόνες GIF, συμπεριλαμβανομένων και εκείνων με διαφάνεια και κινούμενα σχέδια, ενώ παράλληλα παρέχει τη συμπίεση και την κρυπτογράφηση της.

WbStego
wbStego είναι ένα εργαλείο που κρύβει κάθε τύπο αρχείου σε bitmap εικόνες, αρχεία κειμένου, αρχεία HTML ή Adobe PDF αρχεία. Το αρχείο στο οποίο μπορείτε να αποκρύψετε τα στοιχεία δεν έχει οπτικά αλλάξει.

StegoVideo
MSU StegoVideo επιτρέπει να αποκρύψει οποιοδήποτε αρχείο βίντεο σε μια σειρά.
Μπορείτε να χρησιμοποιήσετε VirtualDub MSU StegoVideo ως φίλτρο ή ως standalone.exe του προγράμματος, ανεξάρτητα από το VirtualDub.

5.1 ΔΙΑΦΟΡΟΙ ΤΥΠΟΙ ΑΡΧΕΙΩΝ – ΣΤΕΓΑΝΟΓΡΑΦΙΚΑ ΠΡΟΓΡΑΜΜΑΤΑ

JPG: Μέχρι στιγμής το μόνο στεγανογραφικό πρόγραμμα που κρύβει δεδομένα σε κωδικοποίηση JPEG είναι το Jpeg-Jsteg.

GIF: Τα καλύτερα εργαλεία για στεγανογράφηση σε GIF μορφή είναι τα S-Tools4. Πρόκειται για ένα πρόγραμμα Windows95/NT το οποίο χρησιμοποιεί την τεχνική drag-and-drop.

BMP: Στη περίπτωση αυτή η δουλειά μπορεί να γίνει με συνδυασμό των S-Tools4 και Hide4PGP.

WAV: Ισχύει ότι και στη περίπτωση των αρχείων BMP.

VOC: Μόνο το Hide4PGP μπορεί να επεξεργαστεί αρχεία φωνής.

GZ: Ο τύπος αυτός αντιστοιχεί σε αρχεία που προκύπτουν από τον αλγόριθμο συμπίεσης του Linux και άλλων UNIX συστημάτων. Το GZ σημαίνει Gnu Zip ή Gzip. Στα PC τα αρχεία που συμπιέζονται με το GZ διατηρούν τα πρώτα δύο γράμματα της κατάληξής τους και το τρίτο αντικαθίσταται με το γράμμα "z". Για παράδειγμα το αρχείο README.TXT θα γινότανε README.TXZ. Τέλος, το πρόγραμμα που χρησιμοποιείται είναι το GZSteg.

TXT: Το "Texto" είναι ένα πρόγραμμα που παίρνει σαν είσοδο κρυπτογραφημένα με PGP (ASCII) αρχεία και παράγει ένα αρχείο αποτελούμενο από ακατανόητες φράσεις. Το "Snow" είναι ένα πρόγραμμα που κρύβει δεδομένα χρησιμοποιώντας tabs και κενά στο τέλος των γραμμών ενός αρχείου κειμένου.

6. ΣΤΕΓΑΝΟΓΡΑΦΙΑ ΣΕ ΒΙΝΤΕΟ

Το σήμα βίντεο μπορεί να μεταδοθεί σε διάφορα σχήματα. Μπορεί να κωδικοποιηθεί, να διαφοροποιηθεί, να διαβιβαστεί αποκωδικοποιημένο, μπορεί να demodulated και διαμορφωθεί εκ νέου σε επαφή, ή μπορεί να ψηφιοποιηθεί και τέλος να συμπιεστεί και να μεταδοθεί σε πακέτα.

Αρχεία βίντεο είναι γενικά μια συλλογή από εικόνες και ήχους, έτσι οι περισσότερες από τις τεχνικές που παρουσιάζονται για εικόνες και ήχο μπορούν να εφαρμοστούν και σε αρχεία βίντεο. Το μεγάλο πλεονέκτημα του βίντεο είναι ο μεγάλος όγκος των δεδομένων που μπορεί να κρύβει και το γεγονός ότι είναι κινούμενο ρεύμα εικόνων και ήχων. Επομένως, οποιαδήποτε αλλαγή ακόμα και μικρή, θα μπορούσε να περάσει απαρατήρητη από τον άνθρωπο, λόγω της συνεχούς ροής πληροφοριών. Η πραγματική διαδικασία να κρύβουμε ένα αρχείο μέσα σε άλλο είναι σχετικά απλή. Αλλά η προετοιμασία για την διαδικασία (η συρρίκνωση του βίντεο, η διεύρυνση των γραφικών, εύρεση του στεγανογραφικού προγράμματος) είναι αρκετά χρονοβόρα λόγω του μεγέθους των περιορισμών.

Αρχικά θα αναζητήσουμε ένα αρχείο αρκετά μεγάλο για να χρησιμοποιηθεί σαν μεταφορέας αρχείο, ενώ παράλληλα παρέχεται η δυνατότητα προσάρτησης ενός κωδικού πρόσβασης στο κρυφό αρχείο. Οι πληροφορίες προστατεύονται με το κωδικό αυτό, μια και είναι μοναδικός.

Το στέγο-βίντεο επιτρέπει να κρύβεται κάτι σε οποιοδήποτε αρχείο βίντεο. Όταν το πρόγραμμα δημιουργήθηκε ο αλγόριθμος που επιλέχτηκε προέβλεπε μικρή απώλεια δεδομένων μετά το βίντεο. Η διαβίβαση των δεδομένων κάθε φορά υπόκεινται σε κάποιο ποσοστό διαφθοράς λόγω σφαλμάτων, αλλά η μετάδοση βίντεο λόγω της φύσης του πραγματικού χρόνου ασχολείται με αυτά τα λάθη χωρίς την μετάδοση των κατεστραμμένων δεδομένων. Το MPEG-2 χρησιμοποιεί δεδομένα που κρύβονται για τη μετάδοση πληροφοριών και διορθώνει λάθη από πολλές τεχνικές απόκρυψης του αποκωδικοποιητή.

Έτσι σε 30 καρέ/δευτερόλεπτο βίντεο μπορούμε να αποθηκεύσουμε 30 φορές τις πληροφορίες μίας ενιαίας εικόνας. Επιπλέον, κάποιοι μπορεί να διαλέξουν πλαίσια για την αποθήκευση με τέτοιο τρόπο ώστε ακόμα και αν κάποιος μπορούσε να εντοπίσει την ύπαρξη τροποποίησης, δεν θα κατάφερνε να εξάγει την πληροφορία.

6.1 ΉΧΟΣ

Υπάρχει μεγάλο ενδιαφέρον για τη χρήση ψηφιακών πολυμέσων, όπως εικόνες, ήχο, βίντεο, με σκοπό την απόκρυψη στοιχείων. Πληροφορίες που κρύβονται στο ψηφιακό ήχο μπορεί να χρησιμοποιηθούν για ποικίλες εφαρμογές, όπως η απόδειξη της κυριότητας, της ταυτότητας του ελέγχου πρόσβασης, έλεγχος ακεραιότητας, μυστική επικοινωνία, λήψη δακτυλικών αποτυπωμάτων, παρακολούθηση και εκπομπή εκδήλωσης σχολιασμών. Οι περισσότερες από τις τεχνικές που χρησιμοποιούνται σε εικόνες, μπορούν επίσης να εφαρμοστούν και σε αρχεία ήχου.

Μερικές από αυτές παρουσιάζονται παρακάτω:

- **χαμηλά bit κωδικοποίησης** τα οποία είναι παρόμοια με την μέθοδο LSB που χρησιμοποιούνται γενικά στις εικόνες. Αντικαθιστά το λιγότερο σημαντικό Bit (LSB) ενός ηχητικού σήματος. Το ανθρώπινο ακουστικό σύστημα δεν λειτουργεί πέρα από ένα δυναμικό εύρος συχνοτήτων. Τα χαμηλά bit κωδικοποίησης δεν είναι συνήθως αισθητά στο ανθρώπινο αυτί, γιατί δεν δημιουργούν σημαντικές αλλαγές στην ακουστική του ήχου. Αυτό συνεπάγεται την εκμετάλλευση των ανθρώπινων περιορισμών, χρησιμοποιώντας συχνότητες που δεν ακούγονται στο ανθρώπινο αυτί. Δουλεύοντας με κάθε συχνότητα πάνω από 20,000 Hz, τα μηνύματα που μπορεί να κρύβονται δεν θα εντοπιστούν από τους ελέγχους του ανθρώπου.

Το ανθρώπινο ακουστικό σύστημα ενώ έχει μεγάλο δυναμικό εύρος συχνοτήτων, έχει επίσης έναν αρκετά μικρό διαφορικό εύρος. Κατά συνέπεια, οι δυνατοί ήχοι τείνουν να καλύψουν πιο ασθενές ήχους. Υπάρχουν και μερικές περιβαλλοντικές διαστρεβλώσεις οι οποίες αγνοούνται από τον ακροατή στις περισσότερες περιπτώσεις. Εκμεταλλευόμαστε πολλά από αυτά τα γνωρίσματα, καθώς έχουμε λάβει υπόψη προσεκτικά τις ευαισθησίες του ανθρώπινου ακουστικού συστήματος.

- **Spread Spectrum** είναι μια άλλη μέθοδος που χρησιμοποιείται για να συγκαλύψει πληροφορίες στο εσωτερικό ενός αρχείο ήχου. Αυτή η μέθοδος λειτουργεί με την προσθήκη τυχαία θορύβου στο σήμα της πληροφορίας που είναι να κρύψουν μέσα σε ένα μεταφορέα και σε όλη την φάσμα συχνοτήτων.

- **Echo δεδομένα** μπορούν να κρύβονται μέσα σε ένα αρχείο ήχου. Αυτή η μέθοδος χρησιμοποιεί την ηχώ σε αρχεία ήχου, προκειμένου να αποκρύψουμε πληροφορίες. Με την απλή προσθήκη επιπλέον ήχου μέσα σε ένα αρχείο ήχου. Το κομμάτι που καθιστά τη μέθοδο αυτή καλύτερη από άλλες μεθόδους είναι ότι μπορεί να βελτιώσει πραγματικά τον ήχο του ήχου μέσα στο αρχείο.

- **Masking**: Αυτή η τεχνική έχει τη μεγαλύτερη ικανότητα ενσωμάτωσης, αλλά και πάλι είναι λιγότερο ισχυρή. Αποθηκεύει δεδομένα σε ασήμαντες περιοχές του φάσματος.

Παράδειγμα: Εάν έχουμε ένα αρχείο ήχου σε wav μορφή με τα εξής χαρακτηριστικά : 44100 Hz 16-bit stereo του ενός λεπτού, τότε έχουμε διαστάσεις αρχείου = (16-bit x 44100 Hz x 60sec) x 2 (είναι δικάναλο)= 84672000 bit
Έχουμε συνεπώς μέγεθος για να κρύψουμε στο αρχείο (χρησιμοποιώντας τα 2τελευταία LSB) = 84672000 bit / 16 x 2 = 10584000 bit. Ποτέ δεν κρύβουμε μία πληροφορία σε ένα wav ή bmp αρχείο και μετά το συμπιέζουμε, γιατί υπάρχει μεγάλος κίνδυνος να χαθούν τα δεδομένα που είχαμε κρύψει.

Το πιο σημαντικό είναι η κάλυψη της ποσότητας των δεδομένων που μπορεί να αποθηκευτεί στο εσωτερικό της εικόνας ή του ήχου, παράλληλα με την αισθητή αλλαγή των ιδιοτήτων του αντικειμένου-κάλυψη. Όταν μια εικόνα είναι παραμορφωμένη ή ένα μουσικό κομμάτι ακούγεται διαφορετικά από το αρχικό, θα προκαλέσει υποψίες και είναι δυνατό να ελεγχθεί πιο διεξοδικά.

7.ΣΥΓΧΡΟΝΕΣ ΤΕΧΝΙΚΕΣ ΣΤΕΓΑΝΟΓΡΑΦΙΑΣ

Υπάρχουν πολλές μέθοδοι που χρησιμοποιούνται για την απόκρυψη πληροφοριών στο εσωτερικό της εικόνας, ήχου και βίντεο αρχείων. Οι τεχνικές αυτές μπορούν να χρησιμοποιηθούν με ποικίλους βαθμούς επιτυχίας για διαφορετικούς τύπους αρχείων εικόνας. Οι πιο κοινές μέθοδοι είναι **έγχυσης, LSB (το λιγότερο σημαντικό byte)** και τροποποιήσεις των διακριτά συνημίτονο μετασχηματισμών (**DCT**) .

7.1 Έγχυσης (injection)

Η έγχυσης είναι μια απλή μέθοδος που απλά συνεπάγεται απευθείας εισαγωγή των μυστικών πληροφοριών στο αρχείο του μεταφορέα. Το κύριο πρόβλημα αυτής της μεθόδου είναι ότι μπορεί να αυξήσει σημαντικά το μέγεθος του αρχείου μεταφορέα.

7.1.1 Αντικατάσταση - τροποποίηση της LSB

Η τροποποίηση-αντικατάσταση γίνεται συνήθως σε bytes του αρχείου που δεν είναι πραγματικά αναγκαία ή είναι λιγότερο σημαντικά. Αυτές οι περιοχές μπορεί να αντικατασταθούν από την πληροφορία που πρόκειται να κρύψουμε. Η αλλαγή αυτών των bits της εικόνας βέβαια προκαλεί ανεπαίσθητες αλλαγές στη μορφή της. Χωρίς απευθείας όμως σύγκριση με την αρχική εικόνα είναι πραγματικά αδύνατο να πει κανείς ότι κάτι άλλαξε. Η μέθοδος αυτή δεν αυξάνει το μέγεθος του αρχείου αλλά ανάλογα με το μέγεθος των πληροφοριών που πρέπει να κρύψουμε, το αρχείο μπορεί να παραμορφωθεί αισθητά.

Η μέθοδος LSB λειτουργεί καλύτερα σε αρχεία εικόνας που έχουν υψηλή ανάλυση και κάνουν χρήση πολλών διαφορετικών χρωμάτων καθώς και με τα αρχεία ήχου που έχουν πολλούς διαφορετικούς ήχους και που έχουν ένα υψηλό ρυθμό μετάδοσης bit. Όταν μία εικόνα είναι υψηλής ποιότητας είναι πολύ ευκολότερο να κρυφτούν και να καλυφθούν οι πληροφορίες στο εσωτερικό της. Αν και 24 Bit εικόνες είναι καλύτερες για απόκρυψη πληροφοριών, λόγω του σχετικά μεγάλου μέρους χώρου για απόκρυψη των μηνυμάτων κάποιοι μπορεί να επιλέξουν 8 bit BMP ως κάλυψη πηγή λόγω του μικρότερου διαστήματος και των διαφορετικών ιδιοτήτων ή ενδεχομένως μια άλλη μορφή εικόνας, όπως το GIF, μια και η απόσπαση μεγάλων εικόνων από το διαδίκτυο μπορεί να προκαλέσει υποψίες. Συνήθως θεωρείται ότι οι αλλαγές στο LSBs των χρωμάτων δεν μπορούν να ανιχνευθούν λόγω του θορύβου που είναι πάντα παρών στις ψηφιακές εικόνες.

Για έναν υπολογιστή ένα αρχείο εικόνας είναι απλά ένα αρχείο με διαφορετικά χρώματα και εντάσεις του φωτός σε διαφορετικούς τομείς μίας εικόνας. Ένα αρχείο εικόνων είναι, είτε δυαδικό αρχείο εικόνας, είτε αποτελείται από έναν πίνακα δεικτών χρώματος(παλέτα) και μία συλλογή των δεικτών που την δείχνουν. Ένα δυαδικό αρχείο εικόνας χρησιμοποιεί έναν δυαδικό αριθμό για να αντιπροσωπεύσει τη RGB αξία ενός εικονοκυττάρου.

Στην άλλη μέθοδο κάθε εικονοκύτταρο, είναι η αξία ενός δείκτη που δείχνει την παλέτα χρώματος όπου οι RGB τιμές των χρωμάτων αποθηκεύονται σε αυτήν. Η παλέτα είναι ένα μπλοκ bytes όπου έχουν καταγραφεί τα χρώματα διαθέσιμα για χρήση σε ένα συγκεκριμένο δείκτη-εικόνα χρώμα. Ένα χρώμα είναι όροι διαφορετικών εντάσεων (από 0 έως 255). Τα εικονοκύτταρα αντιπροσωπεύονται από τρία χρώματα:κόκκινο R, πράσινο G και μπλε B. Αυτά τα χρώματα διαμορφώνουν μαζί τις εικόνες ή το βίντεο.

Πολλοί είναι αυτοί που προτείνουν αλλαγή στο χρώμα της παλέτας της εικόνας αντί για τα δεδομένα της εικόνας. Η διαδικασία αυτή μπορεί να κρύψει το μήνυμα και δεν αλλάζει την εμφάνιση της εικόνας. Δεδομένου ότι ο αριθμός των χρωμάτων στην παλέτα είναι περιορισμένη, έτσι και η στεγανογραφική ικανότητα είναι μάλλον περιορισμένη.

Όταν χρησιμοποιούμε μία 24bit εικόνα, χρησιμοποιείται ένα bit για κάθε ένα από τα κόκκινο, πράσινο και μπλε χρώμα, έτσι σε κάθε pixel συνολικά 3 bits μπορεί να αποθηκευτούν. Κάθε χρώμα κάθε εικονοκυττάρου απαιτεί 8 bit.Δεδομένου ότι το πρώτο κομμάτι είναι το «πιο ελάχιστα σημαντικό» ή φέρνει το λιγότερο ποσοστό σπουδαιότητας αυτή η στεγανογραφική τεχνική επιλέγει να επικαλύψει το πρώτο κομμάτι των διαδοχικών ψηφιολέξεων έως ότου ενσωματωθεί ολόκληρο το μυστικό μήνυμα στο αρχικό αρχείο πηγής ή τα στοιχεία κάλυψης. Επειδή έχουμε τροποποιήσει μόνο τα λιγότερα σημαντικά bit μίας μερίδας του αρχείου πηγής, το ανθρώπινο μάτι δεν είναι σε θέση να ανιχνεύσει την υποβάθμιση στην εικόνα ή το βίντεο.
Για παράδειγμα: μία 24-bit εικόνα θα έχει 8 bit, που αντιπροσωπεύουν κάθε μία από τις τρεις - τιμές χρωμάτων (κόκκινο, πράσινο και μπλε) σε κάθε pixel.

Εάν εξετάσουμε το χρώμα μπλε θα καταλήξουμε σε 28 διαφορετικές τιμές. Η διαφορά μεταξύ 11111111 και 11111110 στην αξία για την μπλε ένταση δεν είναι ανιχνεύσιμη από το ανθρώπινο μάτι. **Μία 800X600 pixel εικόνα μπορεί να περιέχει το συνολικό ποσό των 1.440.000 bits(180,000 bytes) των μυστικών δεδομένων**.

Ας υποθέσουμε ότι έχουμε το ακόλουθο πλέγμα. Θεωρούμε ότι έχουμε 3 εικονοστοιχεία (9 bytes μνήμης) μιας εικόνας 24 bit με την ακόλουθη κωδικοποίηση RGB:
(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

Όταν ο χαρακτήρας 1, η οποία δυαδική τιμή ισούται με 10000001, προστίθεται το ακόλουθο πλέγμα έχει το αποτέλεσμα:
(00100111 1110100 ***0*** 11001000)
(0010011 ***0*** 11001000 1110100 ***0)***
(11001000 00100111 11101001)
Στην περίπτωση αυτή, μόνο τρία κομμάτια έπρεπε να αλλάξουν για να εισαχθεί ο χαρακτήρας επιτυχώς.

Κατά μέσο όρο, μόνο το ήμισυ των bits σε μια εικόνα θα πρέπει να τροποποιηθεί για να κρύψει ένα μυστικό μήνυμα χρησιμοποιώντας την μέγιστη κάλυψη μέγεθος.
Για να αποκρύψουμε περισσότερα δεδομένα, η κάλυψη εικόνα θα πρέπει να έχει αρκετά άκρα εικονοστοιχεία για απόκρυψη στοιχείων.
Βέβαια οι αλλαγές που έχουν γίνει στα λιγότερο σημαντικά bits είναι πολύ μικρές για να είναι αναγνωρισμένες από το ανθρώπινο μάτι, ώστε το μήνυμα να παραμείνει αποτελεσματικά κρυφό.
Παρόμοιες μέθοδοι μπορούν να εφαρμοστούν σε 8-bit εικόνες, αλλά οι αλλαγές, όπως ο αναγνώστης θα μπορούσε να φανταστεί, είναι πιο δραματικές.

Η επιλογή των εικονοστοιχείων στα οποία το μήνυμα θα είναι ενσωματωμένο είναι πολύ σημαντικό. Ένα ενιαίο τροποποιημένο pixel ξεχωρίζει από τα ομοιόμορφα γειτονικά pixels καθιστώντας έτσι την εικόνα ύποπτη. Μια πιθανή λύση για το πρόβλημα αυτό είναι να επιλέξετε την άκρη-pixels της εικόνας για να αποκρύψετε το μήνυμα. Σε μια φυσική εικόνα, υπάρχει συσχετισμός μεταξύ των τοπικών γειτονικών pixel της στάθμης του γκρίζου. Αυτό σημαίνει, ότι γειτονικά pixels πιθανότατα έχουν τις ίδιες αξίες.
Μία αλλαγή σε τρία δεκαδικά είναι ισοδύναμο της αλλαγής των δύο τουλάχιστον σημαντικών bits (LSB) -1 από «1» σε ένα «0 -0». Το αντίθετο θα ήταν η αύξηση χρώματος στοιχείου της αξίας δύο LSBs από'0 -0 'σε'1 -1'.
Σκεφτείτε ότι και πάλι το χρώμα του pixel είναι αυτό που αντιλαμβάνεται το μάτι μας όταν συγκεκριμένα ποσοστά κόκκινο, μπλε και το πράσινο είναι παρόντες.

Για την απόκρυψη σε μια εικόνα, χωρίς να μεταβάλλουμε τις ορατές ιδιότητες της, μπορούμε να μεταβάλλουμε **τις "θορυβώδες" περιοχές.** Όσο πιο πολλές παραλλαγές χρωμάτων έχουμε, τόσο μικρότερη προσοχή θα αποσπάσουν οι τυχόν τροποποιήσεις. Οι αλλαγές αυτές είναι δύσκολο να γίνουν αντιληπτές λόγω του θορύβου που είναι πάντα παρών στις ψηφιακές εικόνες.
Η πληροφορία δεν είναι κρυμμένη στο "θορυβώδες" επίπεδο, αλλά είναι μέσα στο ορατό τμήμα της εικόνας, η οποία καθιστά πιο κατάλληλη την LSB μέθοδο σε περίπτωση που έχουμε μία lossy συμπίεση αλγορίθμου όπως είναι JPEG.
Αυτός ο "ορατός θόρυβος" θα μεταφέρει μακριά την ύπαρξη των κρυφών πληροφοριών. Μόνο μετά από την σύγκριση πολλών αρχικών κάλυψη-εικόνων και stego-εικόνων ως προς το χρώμα, τη φωτεινότητα των pixel και τις ενώσεις σημείων να σημειώσουν αισθητές διαφορές.
Έτσι για να καταφέρουν να αποφευχθεί μια τέτοια επίθεση, προχώρησαν στην ενσωμάτωση του αρχικού βίντεο κάλυψη με τον θόρυβο, ώστε να αποκρύψει τις πληροφορίες.

Ένας άλλος σοβαρός παράγοντας είναι η σωστή επιλογή της κάλυψη-εικόνας. Μερικοί εμπειρογνώμονες προτείνουν τη χρήση εικόνων εκκεντρικού χρώματος που στη πλειονότητα τους είναι σε αποχρώσεις του γκρι όπου το ανθρώπινο μάτι δεν θα εντοπίσει τη διαφορά μεταξύ διαφορετικών τιμών γκρι τόσο εύκολα, όσο με τα διαφορετικά χρώματα .

- Οι εικόνες που αποθηκεύονται με τη μέθοδο JPEG είναι μία πολύ φτωχή επιλογή για τις εικόνες κάλυψης-εικόνας. Αυτό επειδή η κβαντοποιήση που εισάγεται από JPEG μπορεί να χρησιμεύσει ως ένα "watermark" και κάποιοι μπορούν να ανιχνεύσουν ακόμη και στις μικρές τροποποιήσεις της εικόνας κάλυψης.
- Ενώ η αρχική εικόνα και μία εικόνα με αντικατάσταση LSB μπορούν να είναι όμοιες στο ανθρώπινο μάτι. Κάποιος μπορεί να δεί τα οπτικά στοιχεία με σύγκριση των ιστιογραμμών των τιμών εικονοκυττάρου μεταξύ ενός αυθεντικού και μίας τροποποιημένης εικόνας.

Μέχρι τώρα αναφέραμε ότι για να κρύψουμε ένα μυστικό μήνυμα μέσα σε μια εικόνα, χρειαζόταν μια σωστή εικόνα κάλυψη. Επειδή αυτή η μέθοδος χρησιμοποιεί bits του κάθε pixel, είναι απαραίτητο να χρησιμοποιήσουμε μία lossless μορφή συμπίεσης, αλλιώς οι κρυφές πληροφορίες θα χαθούν στις μεταμορφώσεις ενός lossy αλγόριθμου συμπίεσης.

Σε γενικές γραμμές, ένα εξαιρετικά υψηλό ποσοστό συμπίεσης κάνει δύσκολη τη δουλειά της στεγανογραφίας. Ενώ τα λάθη συμπίεσης παρέχουν ένα καλό μέρος για να κρύψουμε τα στοιχεία, η υψηλή συμπίεση μειώνει τον όγκο των δεδομένων που διατίθενται για την απόκρυψη του ωφέλιμου φορτίου, αυξάνοντας την κωδικοποίηση και τη διευκόλυνση της εύκολης ανίχνευσης.

Οι ασυμπίεστες εικόνες ενώ έχουν περισσότερες περιττές πληροφορίες προσφέρουν μεγαλύτερη χωρητικότητα για την απόκρυψη μυστικού μηνύματος σε σύγκριση με συμπιεσμένες εικόνες, χωρίς να προκαλέσουν σοβαρές αλλαγές αντιληπτές από άλλους. Από τη άλλη όταν πρέπει να υποβάλλουμε μία εικόνα σε συμπίεση, τα πράγματα γίνονται πιο περίπλοκα όταν έχουμε να προσαρμόσουμε και την μέθοδο της ενσωμάτωσης.

Δύο από τα πιο δημοφιλή format είναι

- Graphic interchange format (gif)
- Bitmap (BMP)

Ωστόσο, λόγω της ύπαρξης εξελιγμένων τεχνικών συμπίεσης, η χρήση τους μειώνεται.

GIF (Graphics Interchange Format)είναι μια εικόνα bitmap μορφής που χρησιμοποιείται ευρέως στο World Wide Web, τόσο για ακίνητες εικόνες (παλέτα χρωμάτων raster εικόνες)όσο και για τα κινούμενα σχέδια.

Το GIF περιορίζεται σε 8-bit παλέτα, ή 256 χρώματα. Το γεγονός αυτό καθιστά το GIF μορφή κατάλληλη για την αποθήκευση των γραφικών με σχετικά λίγα χρώματα, όπως απλή διαγράμματα, σχήματα, τα λογότυπα και κινούμενων εικόνων στυλ. Η μορφή GIF υποστηρίζει η εμψύχωση και εξακολουθεί να χρησιμοποιείται ευρέως για την παροχή εικόνας εφέ. Επίσης, χρησιμοποιεί Lossless συμπίεση η οποία είναι πιο αποτελεσματική όταν μεγάλες περιοχές έχουν ένα μόνο χρώμα, και αναποτελεσματική για λεπτομερείς εικόνες.

Η μορφή αρχείου BMP είναι μία μορφή αρχείου εικόνας που χρησιμοποιείται για αποθήκευση bitmap ψηφιακών εικόνων. Σε ασυμπίεστα αρχεία και πολλές μορφές αρχείων Bitmap, τα εικονοστοιχεία αποθηκεύονται με βάθος χρώματος των 1, 4, 8, 16, 24,ή 32 bit ανά Pixel. Εικόνες των 8Bit και λιγότερα μπορεί να είναι είτε αποχρώσεις του γκρι ή δείκτη χρώμα. Ασυμπίεστα αρχεία είναι πολύ μεγαλύτερα από συμπιεσμένες μορφές αρχείων εικόνας για την ίδια εικόνα.

Το αρχείο BMP περιέχει πληροφορίες βίντεο. Μια εικόνα βίντεο αποτελείται από pixels, τα οποία εκπροσωπούνται από τρεις bytes σε ένα αρχείο BMP. Υπάρχει ένα byte για το μπλε χρώμα, άλλο ένα byte αντιπροσωπεύει το πράσινο και το άλλο για το κόκκινο. Η αξία των byte αντιπροσωπεύει την απόχρωση του χρώματος. Δεδομένου ότι ένα χρώμα εκπροσωπείται από 8 bits, υπάρχουν 2 αποχρώσεις του χρώματος, από το 0000 0000 στο 1111 1111.
Όλα LSBs όλων των bytes μιας εικόνας bmp θα μπορούσαν να αλλάξουν και η εικόνα να εξακολουθεί να έχει την ίδια μορφή από τη μεριά του παρατηρητή. Στη συνέχεια, ένα μήνυμα θα μπορούσε να ενσωματωθεί στο LSBs μιας εικόνας BMP, χωρίς να προκαλέσει υποψίες.

Η μέθοδος LSB συνήθως δεν αυξάνει το μέγεθος του αρχείου, αλλά ανάλογα με το μέγεθος της πληροφορίας που πρέπει να κρύβεται στο αρχείο, μπορεί να γίνει αισθητή στρέβλωση.

Μειονεκτήματα της χρήσης LSB, είναι κατά κύριο λόγο το γεγονός ότι απαιτεί αρκετά μεγάλη εικόνα κάλυψη για να δημιουργήσει έναν αξιόλογο χώρο για τις πληροφορίες. Ακόμη και ασυμπίεστες εικόνες των 800 x 600 pixel που δεν χρησιμοποιούνται συχνά στο Internet, μπορεί η χρήση τους να προκαλέσει υποψίες. Ένα άλλο θα προκύψει όταν συμπιέζεται μια εικόνα που κρύβει ένα μυστικό με τη χρήση lossy αλγόριθμο συμπίεσης. Το κρυμμένο μήνυμα δεν θα επιβιώσει αυτή τη λειτουργία και χάνεται μετά την μετατροπή.
Η LSB μπορεί να μην φαίνεται να έχει ουσιαστική σημασία, εφαρμόζοντας ένα φίλτρο που δείχνει μόνο τις λιγότερο σημαντικές bits, θα εξακολουθήσει να παράγει μια αναγνωρίσιμη εικόνα. Από αυτή την περίπτωση, μπορούμε να πούμε ότι η LSB δεν είναι καθόλου τυχαία, αλλά στην πραγματικότητα περιέχει πληροφορίες σχετικά με το σύνολο της εικόνας.

7.1.2 DISCRETE COSINE TRANSFORM

Ένας πιο σύνθετος τρόπος να κρύβεται κάτι μέσα σε βίντεο έρχεται με τη χρήση διακριτικής συνημίτονο μετασχηματισμού (DCT), μια τεχνική για τη μετατροπή ενός σήματος σε στοιχειώδη συνιστώσες συχνότητας. Χρησιμοποιείται ευρέως σε συμπίεση εικόνας.

Αλλάζει ελαφρώς η κάθε μία από τις εικόνες του βίντεο, μόνο τόσο πολύ ώστε να μην είναι αισθητή από το ανθρώπινο μάτι. Για την ακρίβεια μεταβάλλει τιμές ορισμένων τμημάτων της εικόνας, στρογγυλοποιώντας τες. Για παράδειγμα, αν ένα μέρος μίας εικόνας έχει την τιμή 6,667 θα είναι έως και 7. Τα pixels διατάσσονται, με οριζόντια και κάθετη συχνότητα, αυξανόμενη από αριστερά προς τα δεξιά και κάτω προς τα πάνω, αντίστοιχα. Το φωτεινότερο pixel βρίσκεται στην κάτω αριστερή γωνία και είναι γνωστός ως ο όρος DC, με συχνότητα (0, 0).

Η αλλαγή ενός μεγάλου αριθμού των συντελεστών δεν παράγει εμφανείς αλλαγές, αλλά θα έχει σημαντική επίδραση στο ρυθμό συμπίεσης και πιθανότατα να παράγει στατιστικές ανωμαλίες.

Στηρίζεται σε δύο τεχνικές**.** Η πρώτη είναι **quantization** των συντελεστών της εικόνας και η δεύτερη είναι η **εντροπία κωδικοποίησης**. Quantization είναι η διαδικασία μείωσης του αριθμού των πιθανών αξιών, μειώνοντας έτσι τον αριθμό των bit που απαιτούνται για να το εκπροσωπούν. Εντροπία κωδικοποίησης είναι μια τεχνική για την εκπροσώπηση των δεδομένων compactly δυνατό. Σε γενικές γραμμές, υψηλότερες συχνότητες είναι λιγότερο ορατές στο ανθρώπινο μάτι από χαμηλές συχνότητες.

- **Το πρώτο βήμα** είναι να επιλέξουμε το αρχείο που θέλουμε να αποκρύψουμε, γνωστό και ως αρχείο δεδομένων. Η διαδικασία της ψηφιοποίησης του βίντεο είναι σχετικά απλή -δημιουργία ενός μικρού βίντεο ώστε να κρύψει, είναι πολύ δύσκολο-.

- **Το δεύτερο βήμα** της διαδικασίας είναι να επιλέξουμε το αρχείο που θα χρησιμεύσει ως κρησφύγετο. Το αρχείο αυτό είναι γνωστό ως φορέας αρχείο ή το σκάφος. Ο μεταφορέας αρχείο πρέπει να είναι περίπου 8 φορές μεγαλύτερο από το αρχείο δεδομένων με σκοπό την κρυπτογράφηση για να εργαστούν.
  Έτσι με ένα αρχείο δεδομένων 710KB είχαμε ανάγκη από ένα γραφικό που είναι τουλάχιστον 5600KB.

- **Το τρίτο βήμα** είναι να επιλέξουμε ένα στεγανογραφικό πρόγραμμα, ικανό ώστε να κρύβονται σε αρχείο βίντεο.

- **Το επόμενο βήμα** είναι να στείλει το κρυφό αρχείο σε κάποιον που στη συνέχεια θα το αποκρυπτογραφήσει. Μόλις εξαχθεί το αρχείο βίντεο από το γραφικό βίντεο τότε θα μπορούμε να δούμε το βίντεο.

Ο παραλήπτης για να ανακτήσει το κρυμμένο αρχείο θα πρέπει να εγκαταστήσει το λογισμικό στεγανογραφίας που χρησιμοποιήσατε εσείς και να γνωρίζει και τον κωδικό προστασίας. Έτσι τυχόν αδιάκριτα μάτια (όσοι δεν γνωρίζουν τα δύο αυτά στοιχεία)το μόνο που θα μπορούν να δουν είναι το αρχείο καμουφλάζ.

8. ΣΤΕΓΑΝΑΛΥΣΗ

Στεγανάλυση είναι η τέχνη της ανίχνευσης της χρήσης στεγανογραφίας στο εσωτερικό ενός αρχείου. Δεν ασχολείται με το να προσπαθεί να αποκρυπτογραφήσει τις πληροφορίες στο εσωτερικό ενός αρχείου, μόλις τις ανακαλύψει, αλλά μπορεί να χρησιμεύσει ως ένας αποτελεσματικός τρόπος για να κριθεί η ασφάλεια των επιδόσεων των στεγανογραφικών τεχνικών.

Υπάρχουν πολλές μέθοδοι που μπορούν να χρησιμοποιηθούν για την ανίχνευση της στεγανογραφίας :

- Προβολή του φακέλου και σύγκριση με ένα άλλο αντίγραφο του φακέλου που βρέθηκε στο Διαδίκτυο (Φωτογραφία αρχείου). Υπάρχουν συνήθως πολλαπλά αντίγραφα εικόνων και βίντεο στο Διαδίκτυο, έτσι ώστε να μπορούν να ψάξουν για αρκετά από αυτά και να προσπαθήσουν, να συγκρίνουν το ύποπτο αρχείο με αυτά. Οι διαφορές (εάν υποτεθεί ότι ο μεταφορέας είναι ο ίδιος), θα συνθέτουν το ωφέλιμο φορτίο.

Εάν φορτώσουμε μία εικόνα JPEG και ο ύποπτος είναι επίσης ένα αρχείο JPEG, τότε και τα δύο είναι σχεδόν όμοια εκτός από το γεγονός ότι το ένα είναι μεγαλύτερο από το άλλο, συμπεραίνοντας έτσι ότι ο ύποπτος σας έχει κρυφό αρχείο πληροφοριών στο εσωτερικό του.

- Ακούγοντας το αρχείο. Αυτό είναι παρόμοιο με την παραπάνω μέθοδο που χρησιμοποιείται προσπαθώντας να ανιχνεύσει την ύπαρξη στεγανογραφίας σε αρχεία εικόνας.

Αν προσπαθούμε να εντοπίσουμε κρυμμένες πληροφορίες εντός ενός MP3 αρχείου ήχου θα πρέπει να βρεθεί ένα αρχείο ήχου ώστε να συγκριθεί με εκείνο που χρησιμοποιεί την ίδια συμπίεση (MP3.) Το ίδιο ισχύει και για την εξεύρεση κρυφών πληροφοριών εντός των αρχείων εικόνας.

Υπάρχουν δύο τύποι επιθέσεων κατά των στεγανογραφικά κρυμμένων μηνυμάτων, η ανίχνευση και η απόσπαση τους.

Κάθε εικόνα μπορεί να τροποποιηθεί με στόχο την καταστροφή της οποιαδήποτε κρυμμένης πληροφορίας που μπορεί να κρύβει. Η ανίχνευση της εξοικονομεί χρόνο από τη διαδικασία καταστροφής ή ανάκτησης της μια και γίνεται μόνο όταν η πληροφορία βρεθεί.

Όπως ακριβώς η κρυπτανάλυση εφαρμόζει διάφορες τεχνικές με σκοπό την αποκρυπτογράφηση της πληροφορίας, έτσι και η στεγανάλυση εφαρμόζοντας δικές της τεχνικές αποσκοπεί στην ανίχνευση της κρυμμένης πληροφορίας.

Ο στεγαναλυτής χρησιμοποιεί τεχνικές επίθεσης ανάλογα με το τι είδους πληροφορία έχει στα χέρια του.

- Η μία είναι η –**στέγο αποκλειστική**- όπου είναι διαθέσιμη για ανάλυση μόνο η κρυμμένη πληροφορία. Εάν τόσο η αρχική όσο και η κρυπτογραφημένη πληροφορία είναι διαθέσιμες τότε μιλάμε για επίθεση "γνωστού μέσου".

- Η εναλλακτική είναι η -**επιλεκτική στέγο-επίθεση**- όπου τόσο ο αλγόριθμος που χρησιμοποιείται άλλα και το στέγο –μέσο είναι γνωστά. Μια επίθεση επιλεγμένου μέσου είναι αυτή στην οποία ο στεγαναλυτής δημιουργεί το στέγο-μέσο από κάποιο στεγανογραφικό εργαλείο ή αλγόριθμο γνωστού μηνύματος. Ο στόχος είναι ο καθορισμός συγκεκριμένων ιδιοτήτων του στέγο-μέσου που συγκλίνουν στη χρήση κάποιου στεγανογραφικού εργαλείου ή αλγορίθμου.

Χωρίζεται σε δύο κύριες **κατηγορίες μεθόδου**:

- **Παθητικές -** ανιχνεύει την απουσία ή την παρουσία κρυφών δεδομένων σε ένα παρατηρηθέν μήνυμα.
- **Ενεργές –** εκθέτει μερικές ιδιότητες του μηνύματος ή του αλγόριθμου ενσωμάτωσης. Εξάγει μια έκδοση του μυστικού μηνύματος κατά προσέγγιση, από ένα μήνυμα στέγο.

Ο στόχος είναι να προσδιοριστούν τα πιθανά ρεύματα πληροφοριών, να καθορίσει εάν έχουν κρύψει ή όχι μηνύματα που κωδικοποιούνται σε αυτούς και αν είναι δυνατόν να ανακτήσουμε τις κρυμμένες πληροφορίες. Το ύποπτο ρεύμα πληροφοριών μπορεί ή δεν μπορεί να έχει κρύψει τα στοιχεία που κωδικοποιούνται σε αυτούς. Τα κρυμμένα αρχεία μπορεί να είχαν κρυπτογραφηθεί από πριν στο σήμα ή στο αρχείο. Εκτός αν είναι δυνατόν να ανακτηθούν πλήρως, να αποκρυπτογραφηθούν και να επιθεωρηθούν τα κρυμμένα στοιχεία.

Οι επιθέσεις και η ανάλυση στις κρυμμένες πληροφορίες μπορούν να λάβουν διάφορες μορφές αλλά και να θέσουν εκτός λειτουργίας ή και να καταστρέψουν τις κρυμμένες πληροφορίες. Ένα από τα κύρια αποτελέσματα μίας καλής στεγανάλυσης είναι η μείωση της ικανότητας ενσωμάτωσης και του μέγιστου αριθμού των bits που μπορούν να ενσωματωθούν.

9. ΣΤΕΓΑΝΑΛΥΣΗΣ ΕΡΓΑΛΕΙΑ

Stegdetect και **Xsteg**
Stegdetect είναι ένα αυτοματοποιημένο εργαλείο για την ανίχνευση στεγανογραφίας σε εικόνες. Περιέχει επίσης ένα βοηθητικό πρόγραμμα για επίθεση σε JSteg και JPHide, ενώ είναι ικανό να ανιχνεύει πολλές διαφορετικές μεθόδους σε εικόνες JPEG. Αυτό το βοηθητικό πρόγραμμα ονομάζεται Stegbreak.
Xsteg είναι το GUI (Graphical User Interface) για τα Stegdetect.
http://www.outguess.org/download.php

Steganography Analyzer Τεχνούργημα Scanner (StegAlyzerAS)
StegAlyzerAS έχει τη δυνατότητα να σαρώσει ολόκληρο το σύστημα αρχείων, ή μεμονωμένων καταλόγων, για την παρουσία στεγανογραφικών αντικειμένων.
http://www.sarc-wv.com/stegalyzeras.aspx

Steganography Analyzer Υπογραφή Scanner (StegAlyzerSS)
StegAlyzerSS μας δίνει τη δυνατότητα να σαρώνουμε κάθε αρχείο σχετικά με τα μέσα ενημέρωσης του υπόπτου για την παρουσία Steganography εφαρμογών στους φακέλους. Εάν μία γνωστή υπογραφή ανιχνεύεται, μπορεί να είναι δυνατή η εξαγωγή πληροφοριών με Steganography εφαρμογές που συνδέονται με την υπογραφή του.
http://www.sarc-wv.com/stegalyzerss.aspx

Ψηφιακή Αόρατη Μελάνη Toolkit
Το έργο αυτό παρέχει μια απλή Java που βασίζεται σε steganography εργαλεία που μπορεί να κρύβονται μέσα σε ένα μήνυμα 24-bit χρώμα εικόνας, έτσι ώστε να γνωρίζει πώς ήταν ενσωματωμένο, ή την εκτέλεση της στατιστικής ανάλυσης, για καταστεί ευκολότερο για να βρούμε το απέκρυψαν μήνυμα.
http://sourceforge.net/project/showfiles.php?group_id=139031

Συμπίεση αρχείων
Δοκιμάζουμε αυτή τη μέθοδο συμπιέζοντας το πρωτότυπο αρχείο bitmap και το ίδιο αρχείο που χρησιμοποιείται ως αρχείο μεταφορέας με βάση το WinZip .
(URL: http://www.winzip.com/ddchomea.htm)
Με τη σύγκριση των ιδιοτήτων των δύο αυτών αρχείων, παρατηρούμε ότι το αρχικό συμπιεσμένο αρχείο είναι καλύτερο από το αρχείο του μεταφορέα.

10. Εκτέλεση προγράμματος

➢ **Για την εκτέλεση του προγράμματος χρειαζόμαστε τα παρακάτω:**

1) **Βιβλιοθήκη μετατροπής και χειρισμού βίντεο ffmpeg.**
   Για να καταγράφουμε τη ροή ήχου και βίντεο σε πολλές μορφές. Θέτει σε εφαρμογή έναν αποκωδικοποιητή και στη συνέχεια ένα κωδικοποιητή, για να μετατρέπουμε αρχεία από τη μία μορφή στην άλλη.

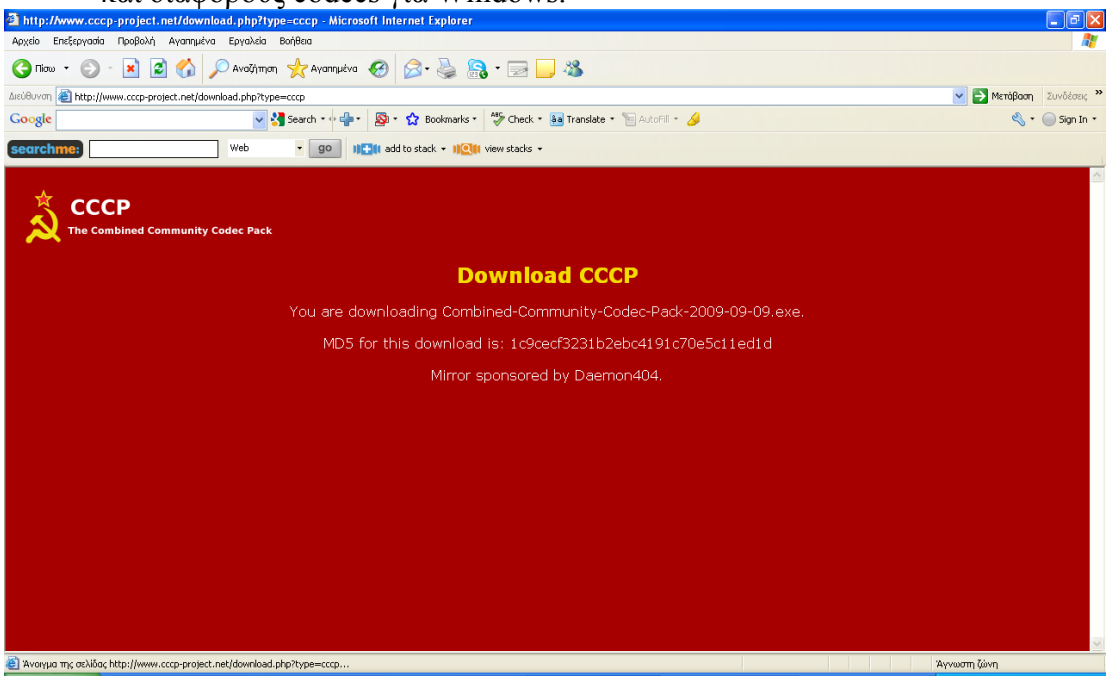
2) **Βιβλιοθήκη Xuggler, που είναι ένας wrapper του ffmpeg για Java.**
   Το χρησιμοποιούμε γιατί επιτρέπει την αποκωδικοποίηση, το χειρισμό, και την κωδικοποίηση (σχεδόν) οποιοδήποτε τύπο αρχείου βίντεο σε σχεδόν πραγματικό χρόνο.

3) **Το Java JDK, έκδοση 6.**
   Η γλώσσα που χρησιμοποιούμε για να γράψουμε το κώδικα του προγράμματος μας.

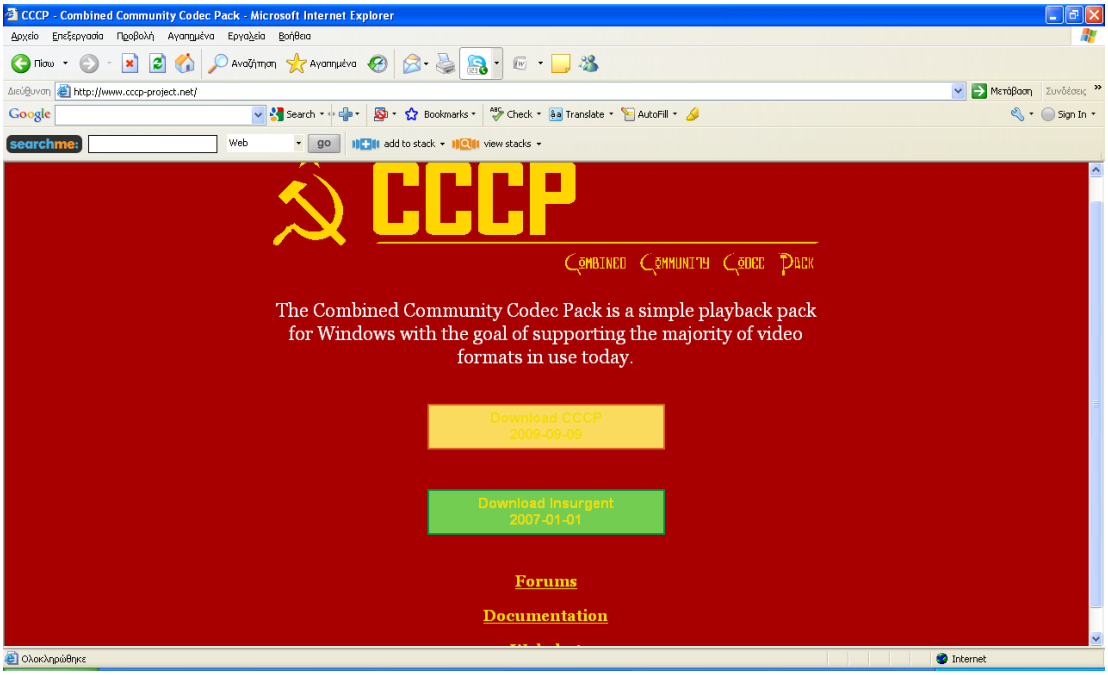
4) **Το Netbeans 6.5**
   Το πρόγραμμα στο οποίο έχουμε γράψει το κώδικα για την κωδικοποιήδη, αποκωδικοποιήση και την επεξεργασία του βίντεο καθώς και του μηνύματος που γράφουμε και θα αποθηκευτεί στα frames του βίντεο.

➢ **Εγκατάσταση** :

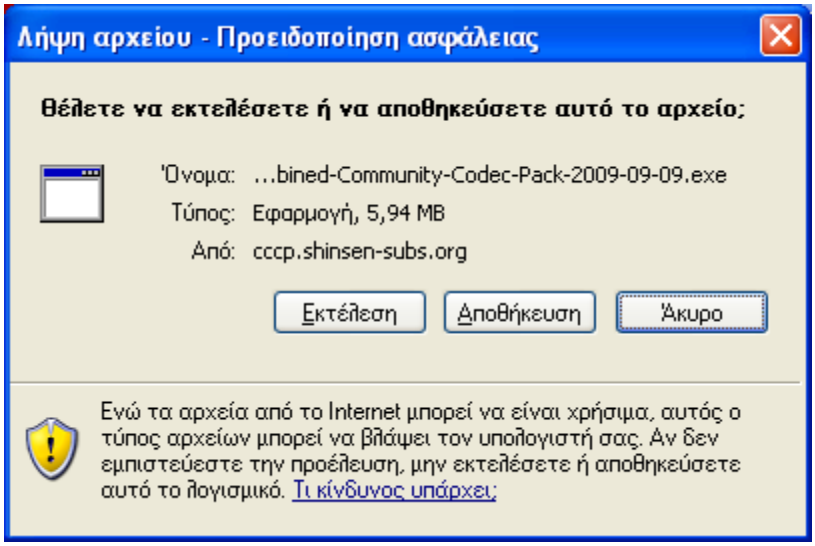
1) **Combined Community Codec Pack**
   http://www.cccp-project.net/download.php?type=cccp
   Πρόκειται για μία συλλογή codecs για Windows, που περιλαμβάνει το ffmpeg και διάφορους codecs για Windows.



Κάνετε κλίκ πάνω αριστερά στο cccp για να πάμε στο επόμενο βήμα της διαδικασίας κατεβάσματος.



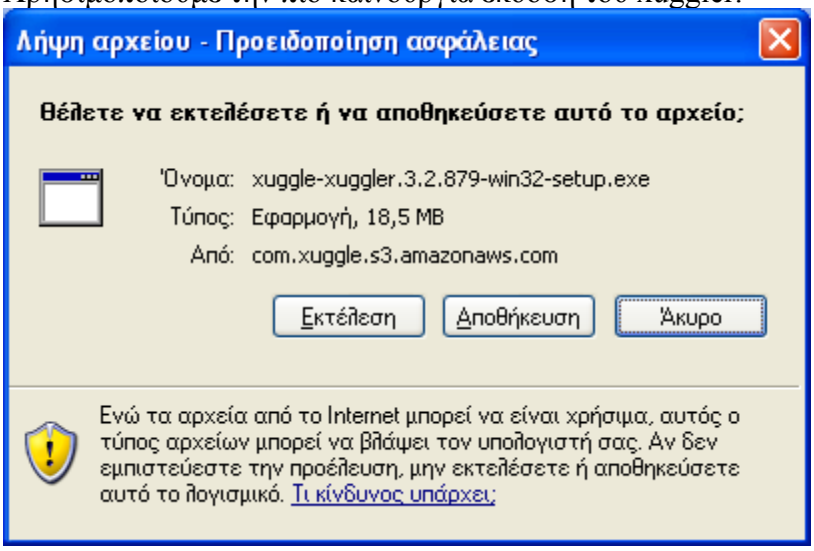
Κάνετε κλίκ πάνω στο πράσινο πλαίσιο download insurgent για να αρχίσει το κατέβασμα.



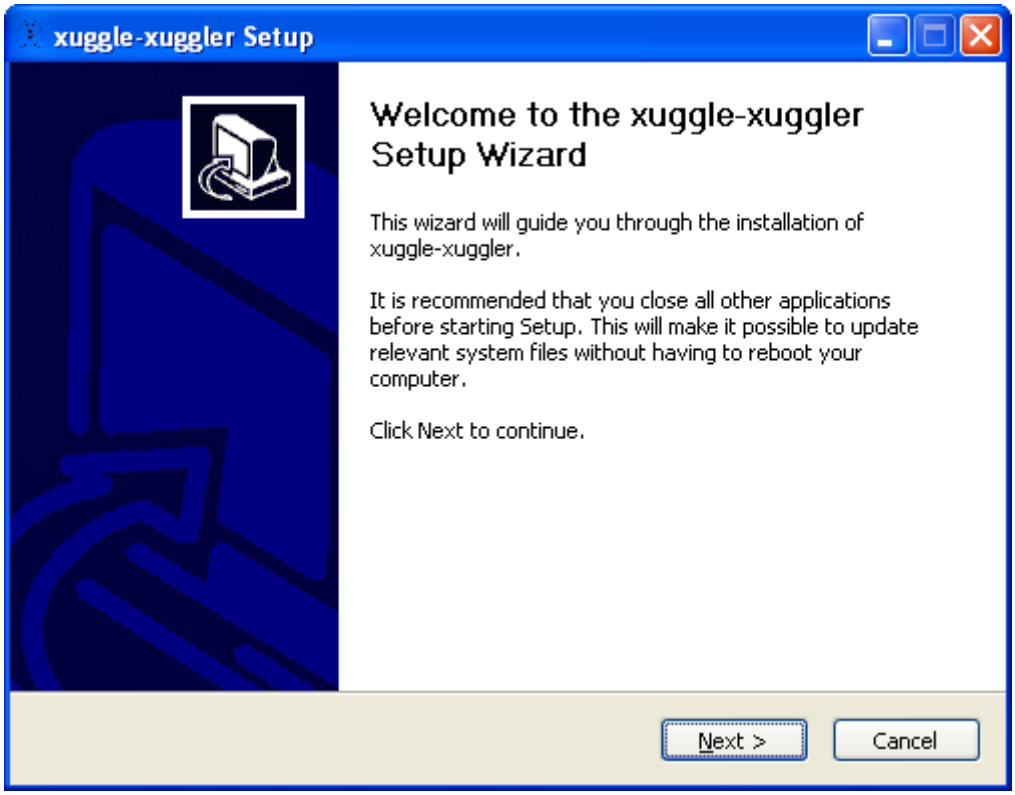
Αποθηκέυετε το αρχείο που κατεβάσατε εκεί που θέλετε μέσα στον υπολογιστή.

2) **Xuggler 3.2**
http://com.xuggle.s3.amazonaws.com/xuggler/xuggler-3.2.FINAL/xuggle-xuggler.3.2.879-win32-setup.exe
Απαιτείται reboot μετά για να λειτουργήσει.

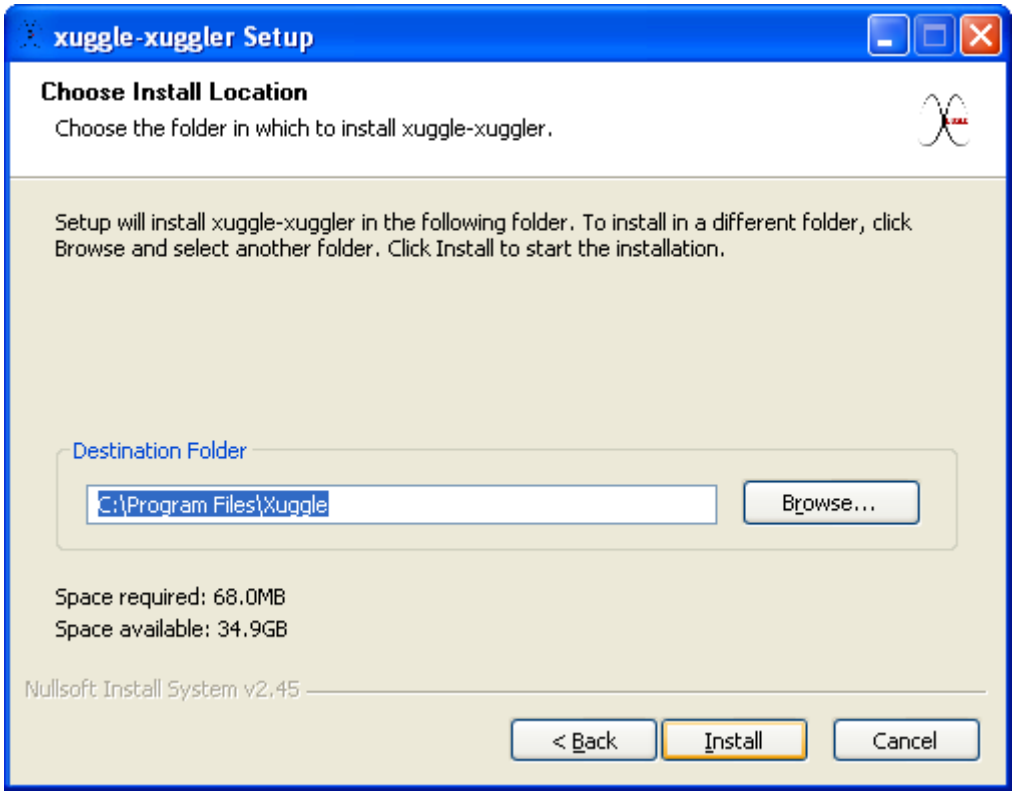
Χρησιμοποιούμε την πιο καινούργια έκδοση του xuggler.



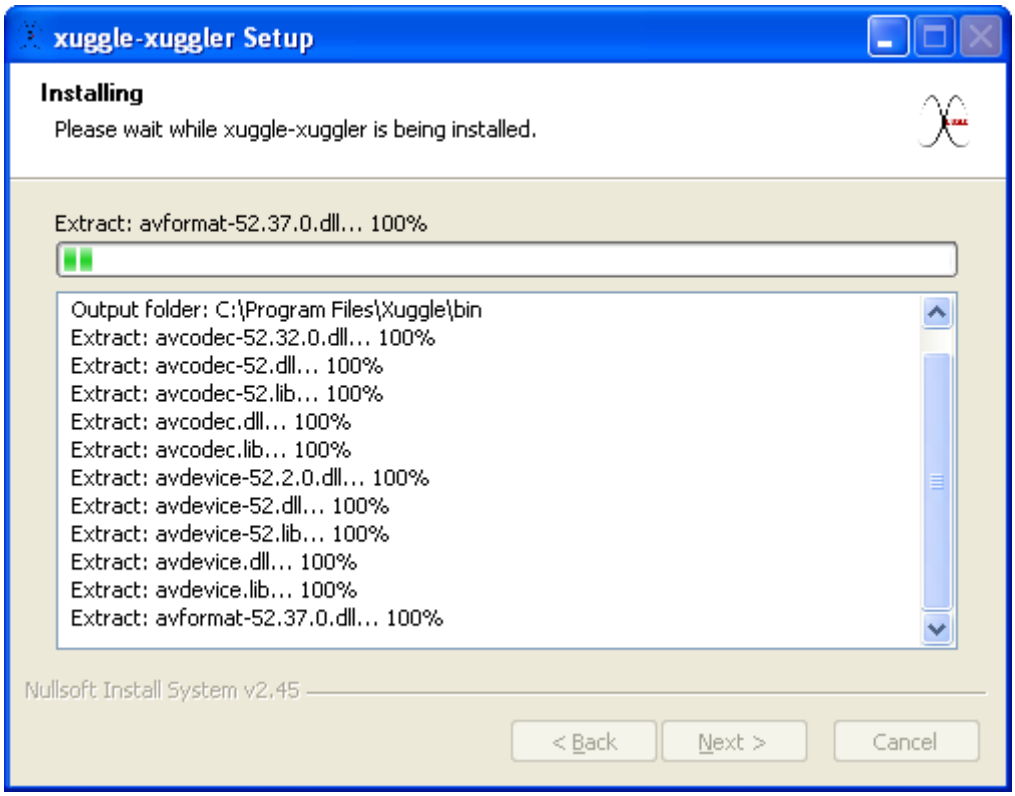
Κάνετε εκτέλεση και γίνεται εγκατάσταση του xuggle project.



Πατάτε next για να προχωρήσει στα επόμενα βήματα εγκατάστασης.

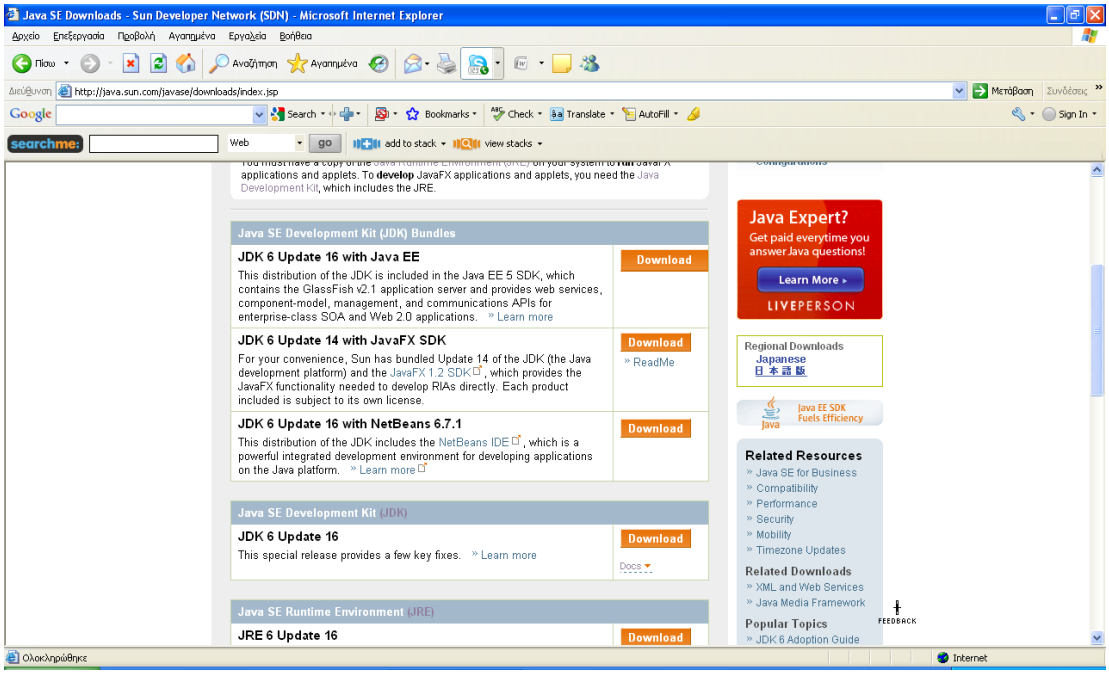


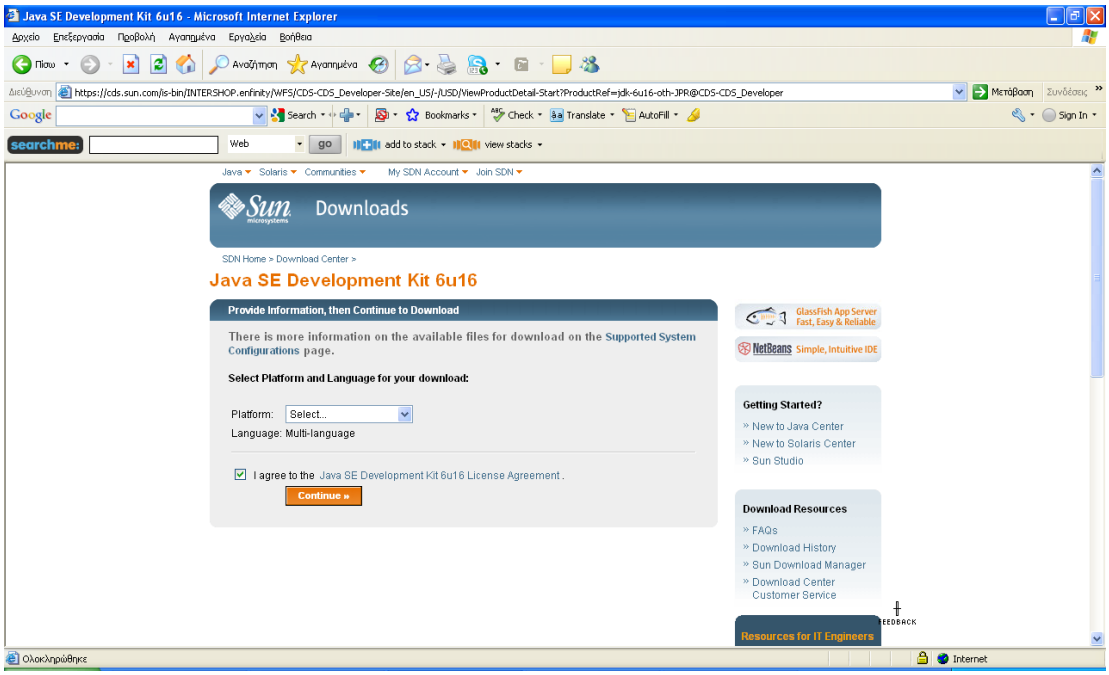
Πατάτε Install για να αρχίσει την εγκατάσταση του προγράμματος, δείχνοντας παράλληλα που θα εγκατασταθεί το παρόν πρόγραμμα.(εδώ μας δείχνει μέσα στο c στο φάκελο program files)



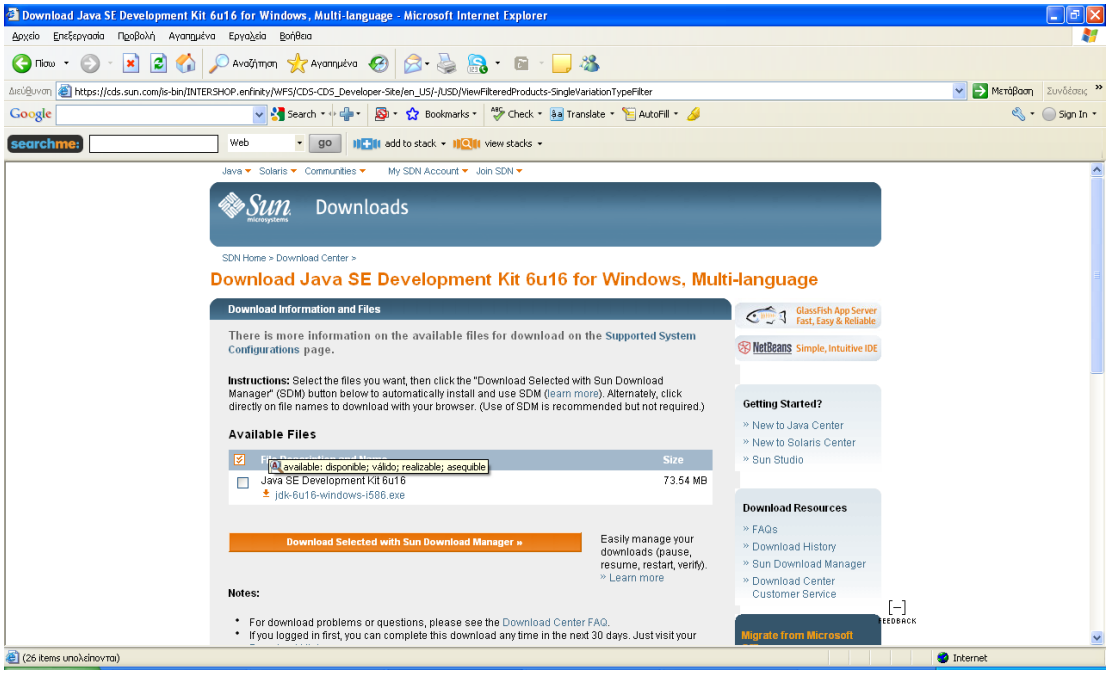
Κάνει εγκατάσταση του προγράμματος.

3) **Java JDK**
http://java.sun.com/javase/downloads/index.jsp
(Η 4η επιλογή)

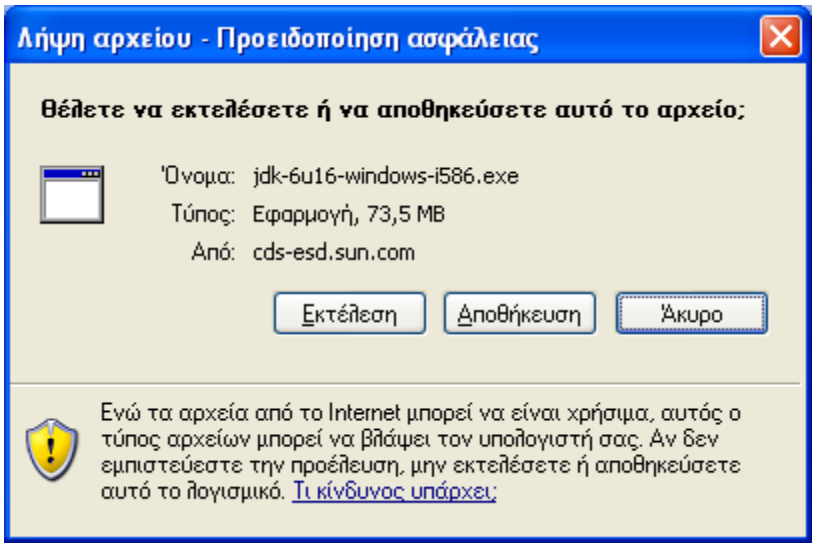




Κλικάρετε στην επιλογή I agree to the…… Πατάτε το κουμπί continue για να συνεχίσετε.



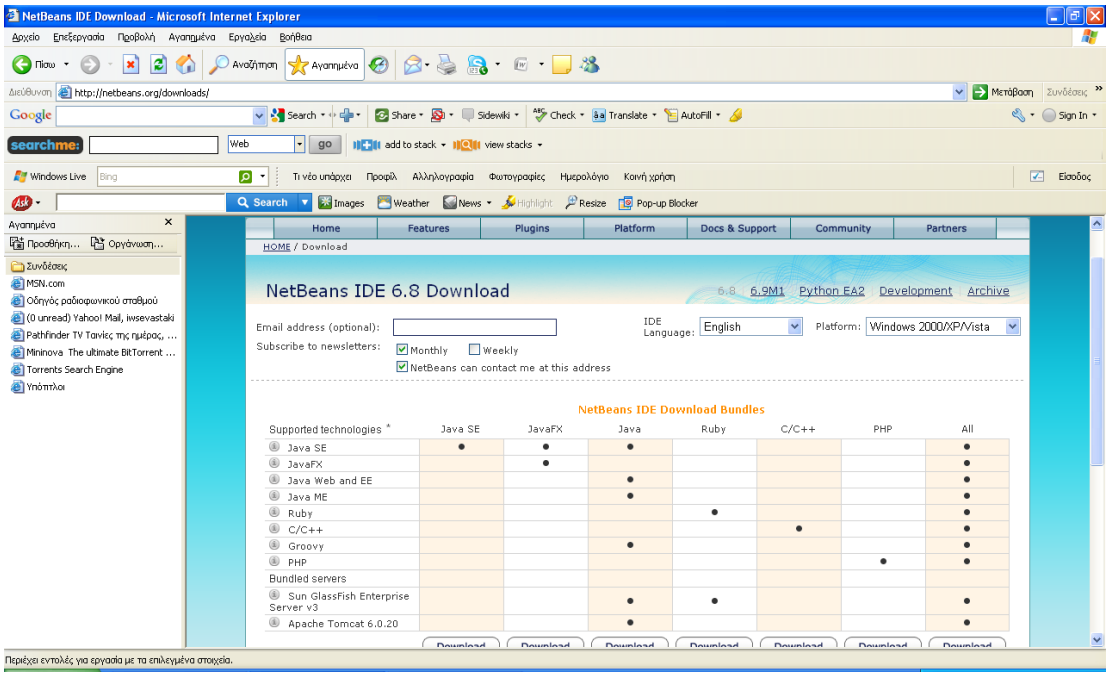
Κατεβάζει την έκδοση που διαλέξαμε στο προηγούμενο βήμα.



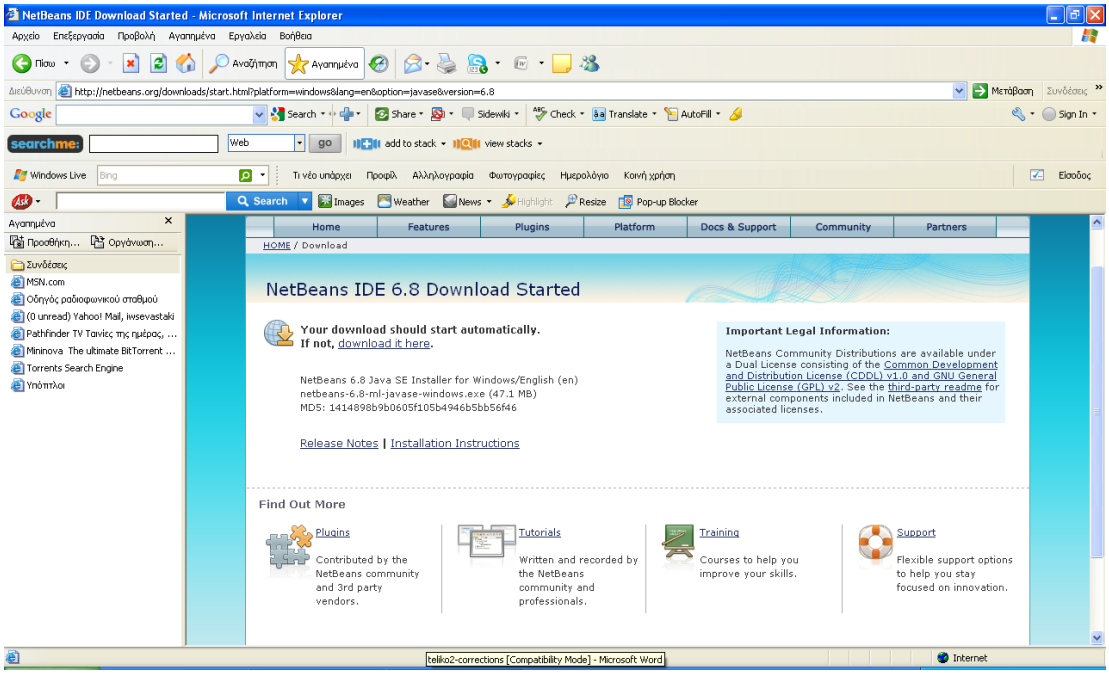
Πατάτε εκτέλεση για να τρέξει το πρόγραμμα.

3) **Netbeans**
http://www.netbeans.org/downloads/
(Η πρώτη επιλογή αρκεί)

Κατεβάζετε την καινούργια έκδοση του netbeans, αλλά το πρόγραμμα παίζει αντίστοιχα και με την προηγούμενη έκδοση 6.7.1

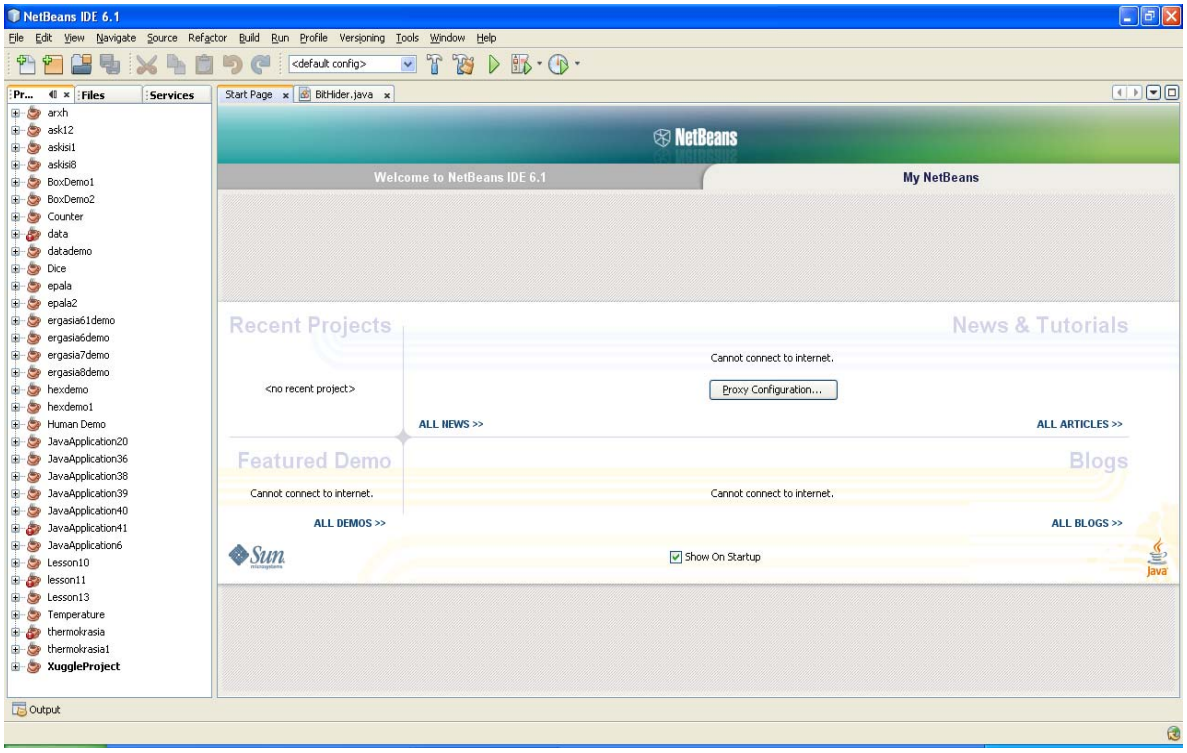


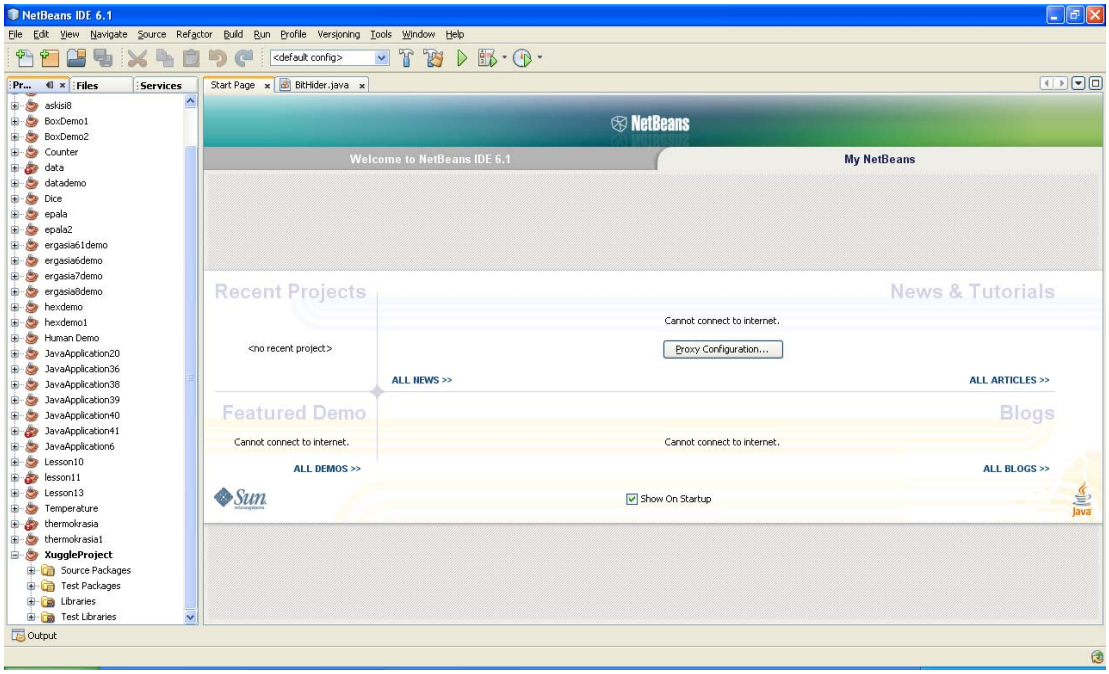
Εδώ φαίνονται όλες οι τεχνολογίες που υποστηρίζει η καθε μία έκδοση. Κάνουμε download την πρώτη επιλογή.



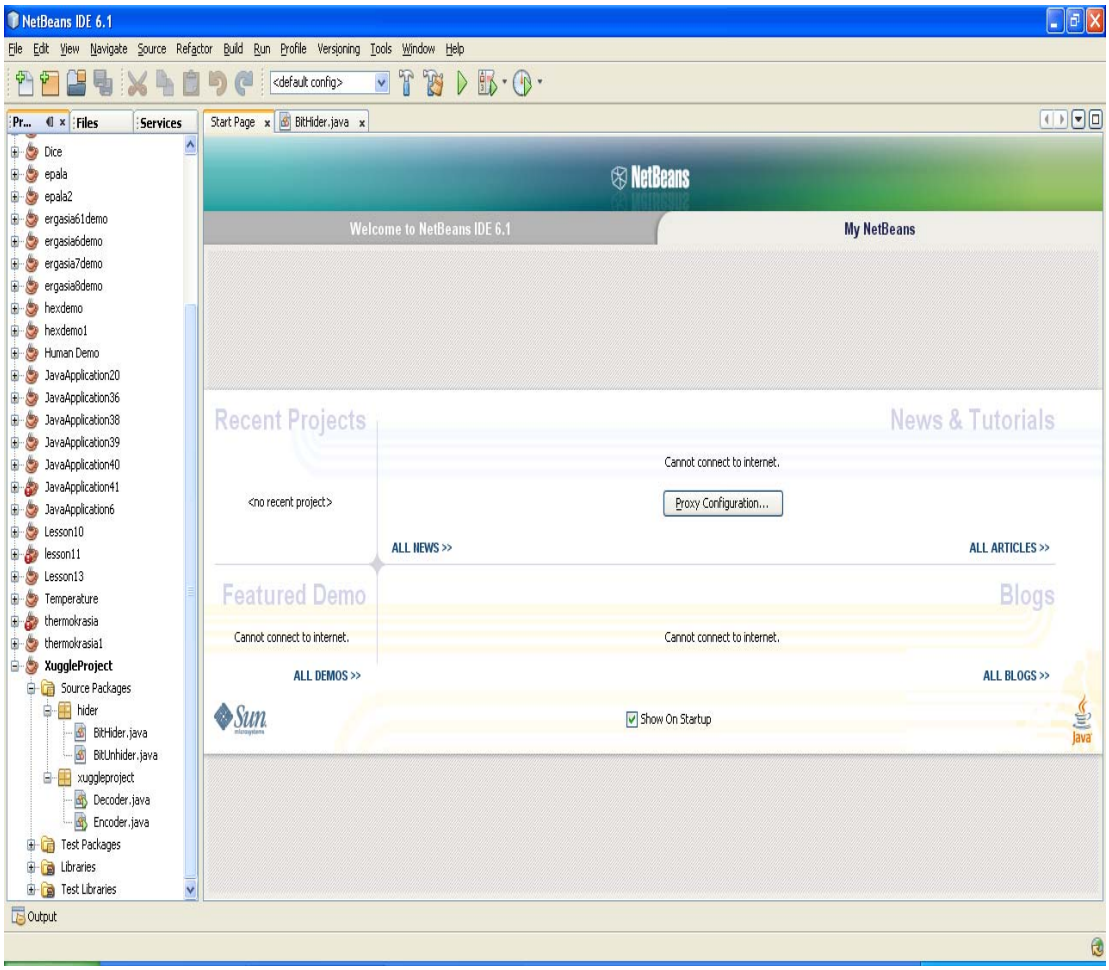
Δείχνει ότι έχει αρχίσει το κατέβασμα. Ύστερα κάνετε εκτέλεση και εγκατάσταση του προγράμματος. Αν δεν αρχίσει αυτόματα το κατέβασμα τότε κάνετε κλίκ πάνω στην ένδειξη download it here και θα αρχίσει το κατέβασμα.

Αφού εγκατασταθεί το netbeans, το ανοίγουμε και επιλέγουμε από τη μπαρά του μενού πάνω το file, έπειτα το open project, βρίσκουμε που είναι το xuggle project και το προσθέτουμε στο netbeans.





Το xuggle project περιλαμβάνει τα source packages, test packages, librairies, test libraries. Όσα χρειάζεται για να τρέξει το πρόγραμμα.



Ανοίγουμε το source package, το οποίο περιέχει το κώδικα της διαδικασίας του BitHider και του BitUnhider. Ενώ το xuggle project περιέχει τον αντίστοιχο κώδικα του Decoder και του Encoder αντίστοιχα.

Αυτό που γίνεται είναι το εξής:

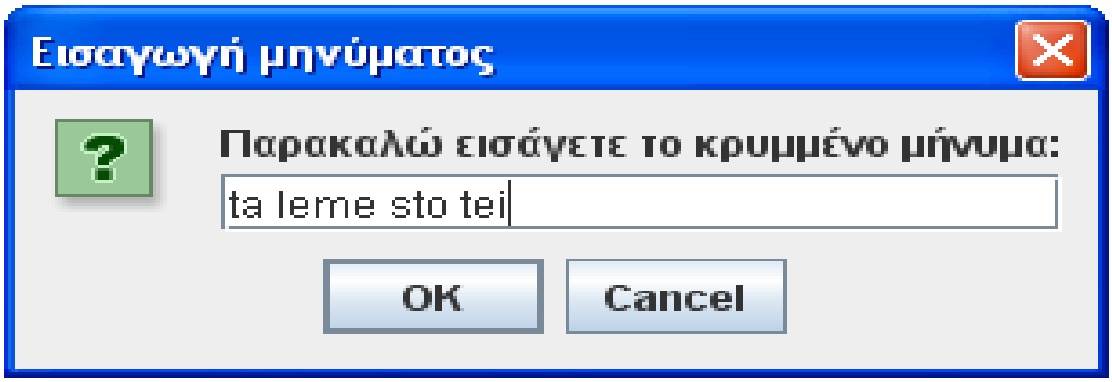
- Η κλάση Encoder που τρέχει κάνοντας διπλό κλικ στο Encoder, είναι υπεύθυνη για να κρύβει το μήνυμα.
- Κάνοντας δεξί κλίκ και επιλέγουμε το run file, μας βγάζει παράθυρο να επιλέξουμε το χώρο που βρίσκεται το βίντεο.
- Επιλέγουμε το βίντεο, το ανοίγουμε και μας ζητάει το μήνυμα που θέλουμε να πληκτρολογήσουμε. Το αποθηκεύει σε ένα νέο αρχείο που περιέχει και το μήνυμα. Αν ανοίξουμε το Wall-e.flv θα φτιάξει το Wall-e(1).avi που είναι πολύ μεγαλύτερο μιας και περιέχει το μήνυμα που γράψαμε.

- Παίζει σίγουρα με dvix, flv και media player.
  Το αρχείο avi που δημιουργείται είναι πολύ μεγάλο μια και περιέχει και το βίντεο αλλά και το κρυμμένο μήνυμα που έχουμε γράψει. Το μέγεθος του μηνύματος που θα γράψουμε εξαρτάται από το μέγεθος του βίντεο που έχουμε επιλέξει. Εδώ έχω επιλέξει μικρό μήνυμα και μικρό βίντεο για να μην είναι χρονοβόρα η διαδικασία της ενσωμάτωσης του μηνύματος στο βίντεο.

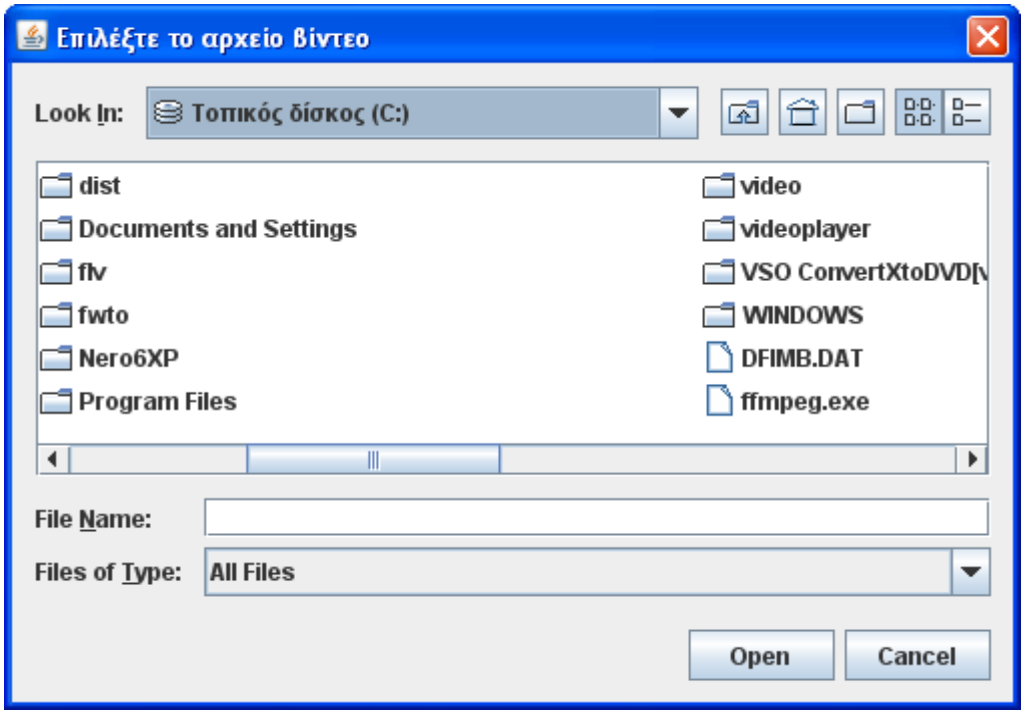
- Η κλάση Encoder ανοίγει το βίντεο και το επανασυμπιέζει. Χρησιμοποιεί το BitHider για να κρύψει μέσα στα pixels το μήνυμα.
- Η κλάση Decoder παίζει το βίντεο και βρίσκει σε αυτό το μήνυμα αν υπάρχει. Χρησιμοποιεί το BitUnhider για να βγάλει από τα pixels του βίντεο το μήνυμα.

Επειδή το μήνυμα έχει γραφτεί από τον Encoder στο Wall-e(1).avi χρειάζεται να ανοίξουμε αυτό το αρχείο από τον Decoder.
Η κλάση Decoder ανοίγει το βίντεο και το δείχνει στην οθόνη. Χρησιμοποιεί την κλάση BitUnhider για να βγάλει το μήνυμα από τα pixels του βίντεο.

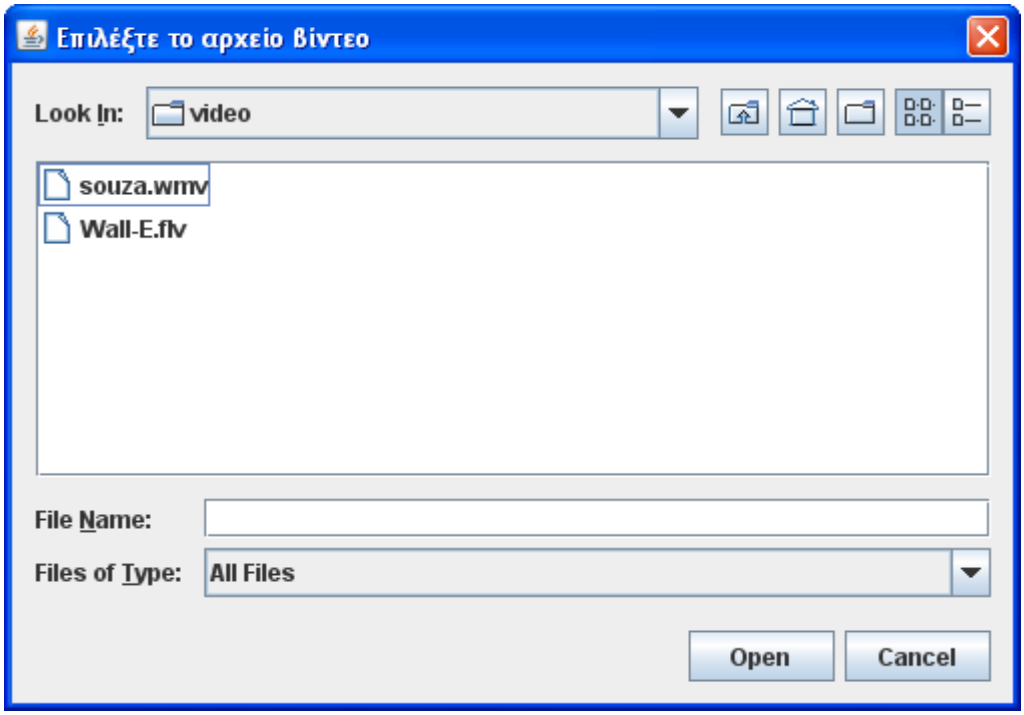
Εμφανίζει το παραθύρο που ζητάει από τον χρήστη να εισάγει το κρυμμένο μήνυμα που αυτός επιθυμεί. Δημιουργεί ένα αντικειμένο reader που θα διαβάζει το μήνυμα.
Αν επιλέξουμε cancel οδηγούμαστε στην έξοδο, αλλιώς συνεχίζει.



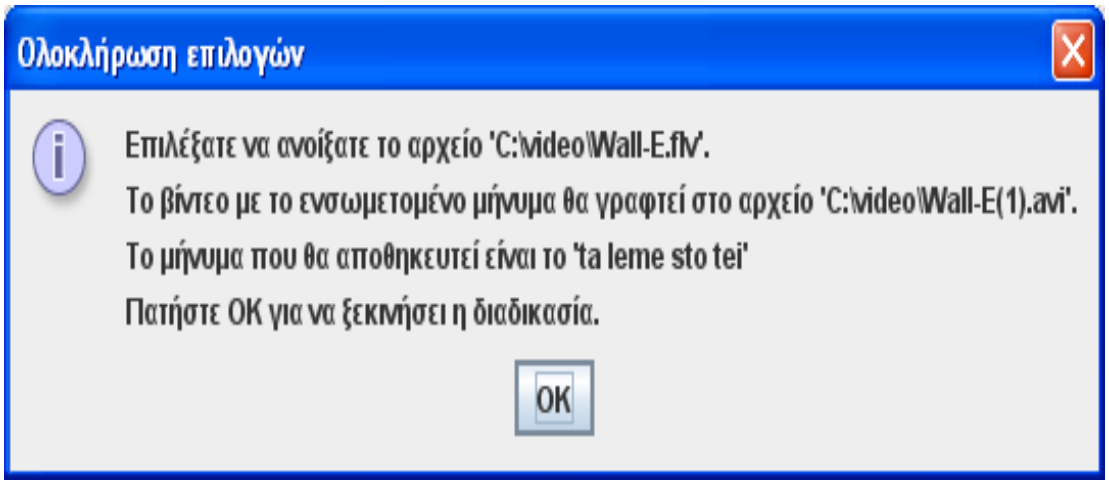
Αφού εισάγουμε το μήνυμα μας πατάμε είτε οκ για να συνεχίσει και να κάνει την διαδικασία ενσωμάτωσης του μηνύματος, είτε πατάμε το cancel για να κλείσουμε το παράθυρο και να μην προβούμε στην διαδικασία.



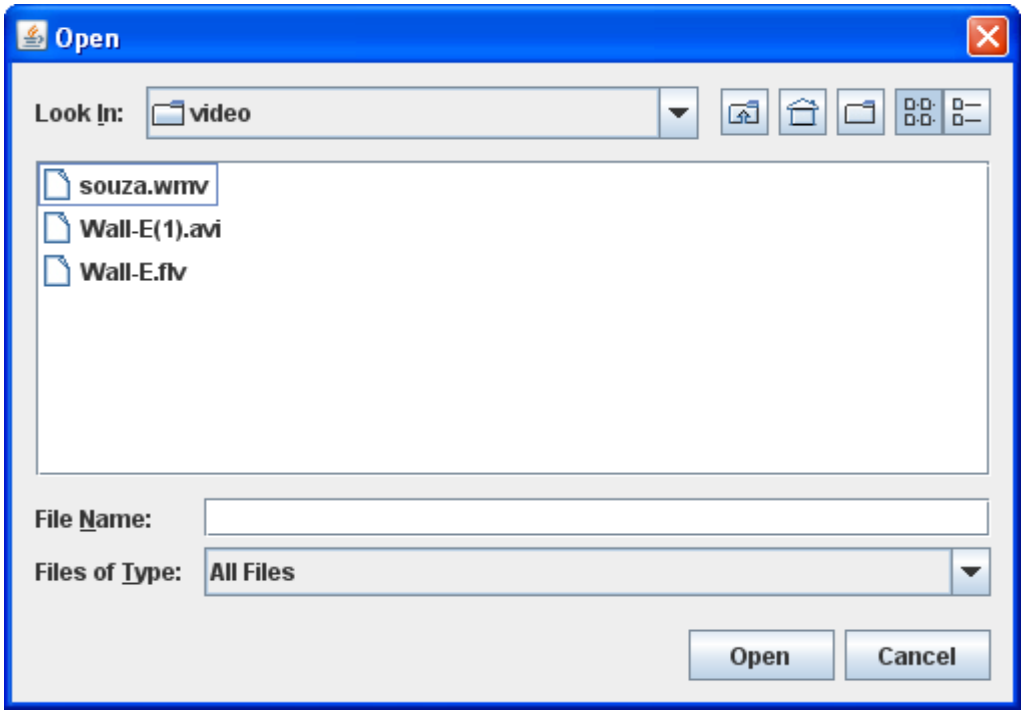
Εδώ πάμε να βρούμε το βίντεο που θέλουμε. Επιλέγουμε να το τοποθετήσουμε στο τοπικό δίσκο c για να μην ψάχνει το πρόγραμμα που βρίσκεται το βίντεο. Έπειτα ανοίγουμε το αρχείο video όπου επιλέγουμε το αρχείο βίντεο που θέλουμε να κρύψουμε εκεί το μήνυμα. Επιλέγω το wall-e .flv.



Εδώ με μήνυμα μας ενημερώνει ποίο αρχείο έχουμε διαλέξει και που θα είναι ενσωνατωμένο το μήνυμα που γράψαμε μόλις παραπάνω. Πατώντας το κουμπί οκ συμφωνούμε να αρχίσει η διαδικασία.

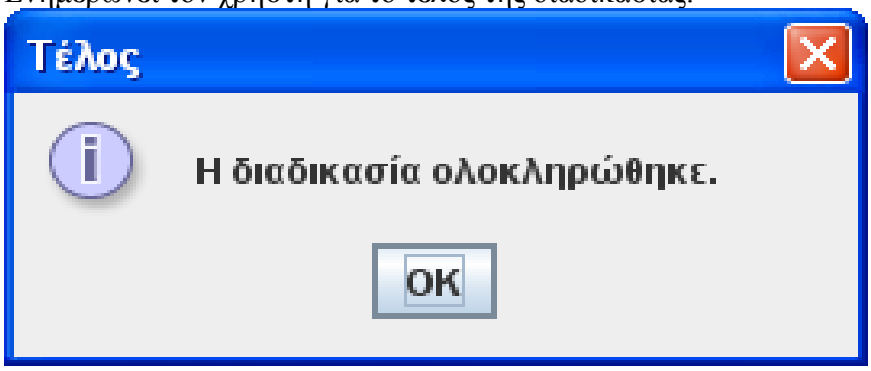


Διατρέχει κάθε pixel της εικόνας μέχρι ο hider να κρύψει όλο το μήνυμα.
Έπειτα καλούμε την μέθοδο hide του hider με παράμετρο το τρέχον pixel, ώστε να κρύψει σε αυτό το μήνυμα.
Δημιουργεί έτσι το αντικειμένο writer που θα γράψει το νέο βίντεο αποτελέσματος.
Αποθηκεύουμε την τροποιημένη εικόνα.
Συσχετίζουμε τον writer με τον reader, ώστε ο writer να γράφει ότι διαβάζει ο reader.
Λέμε στον reader να διαβάσει όλο αρχείο.



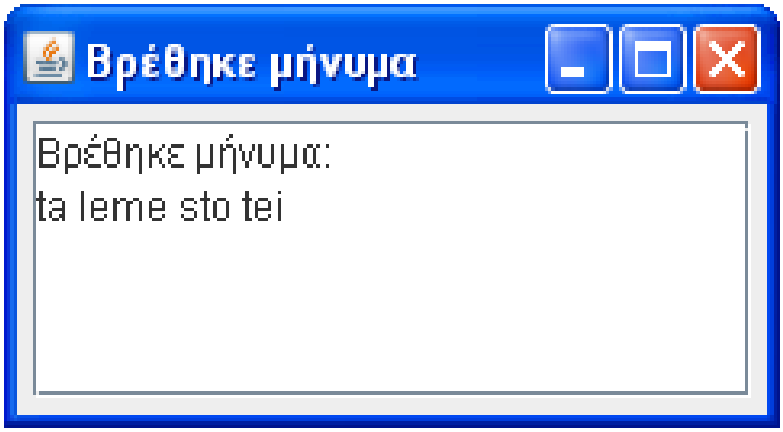
Όπως παρατηρούμε δημιουργείτε στο αρχείο video και νέο βίντεο με το ενσωματωμένο μήνυμα το οποίο είναι το wall-e(1).avi.

Ενημερώνει τον χρήστη για το τέλος της διαδικασίας.

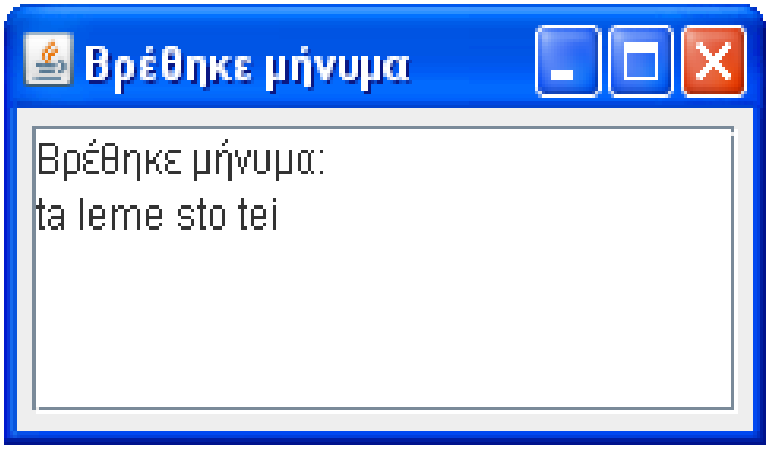


Αφού ολοκληρωθεί η διαδικασία αναπαράγουμε το βίντεο. Το βίντεο παίζει και παράλληλα εμφανίζεται το παράθυρο με το κρυμμένο μήνυμα που γράψαμε στα προηγούμενα στάδια.

Θα μπορούσε να παίζει το βίντεο χωρίς να εμφανίζεται το μήνυμα, αρκεί να το καθορίζαμε εμείς. Εξάλλου πρώτα γράφουμε το κώδικα για να παίζει το βίντεο και ύστερα τα αντίστοιχα για να δημιουργήσουμε το παράθυρο και να γράψουμε μέσα το μήνυμα.

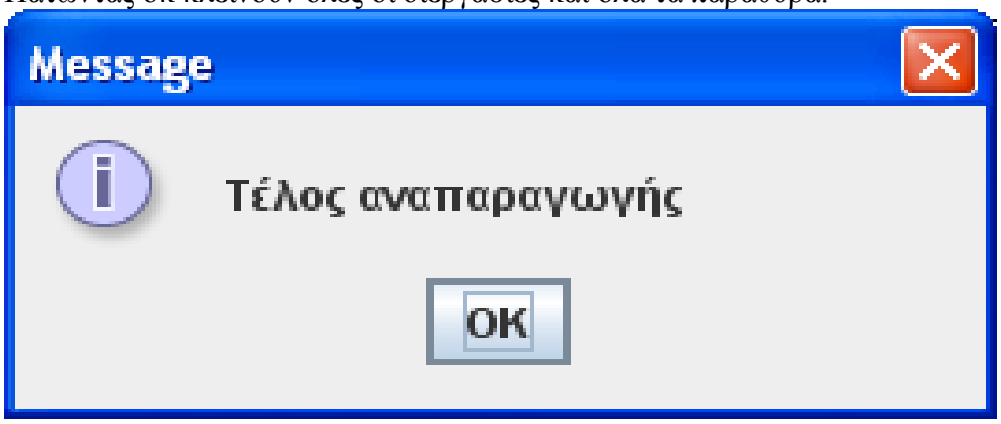


Το βίντεο συνεχίζει να παίζει, ενώ βλέπουμε ότι το παράθυρο με το κρυμμένο μήνυμα παραμένει.



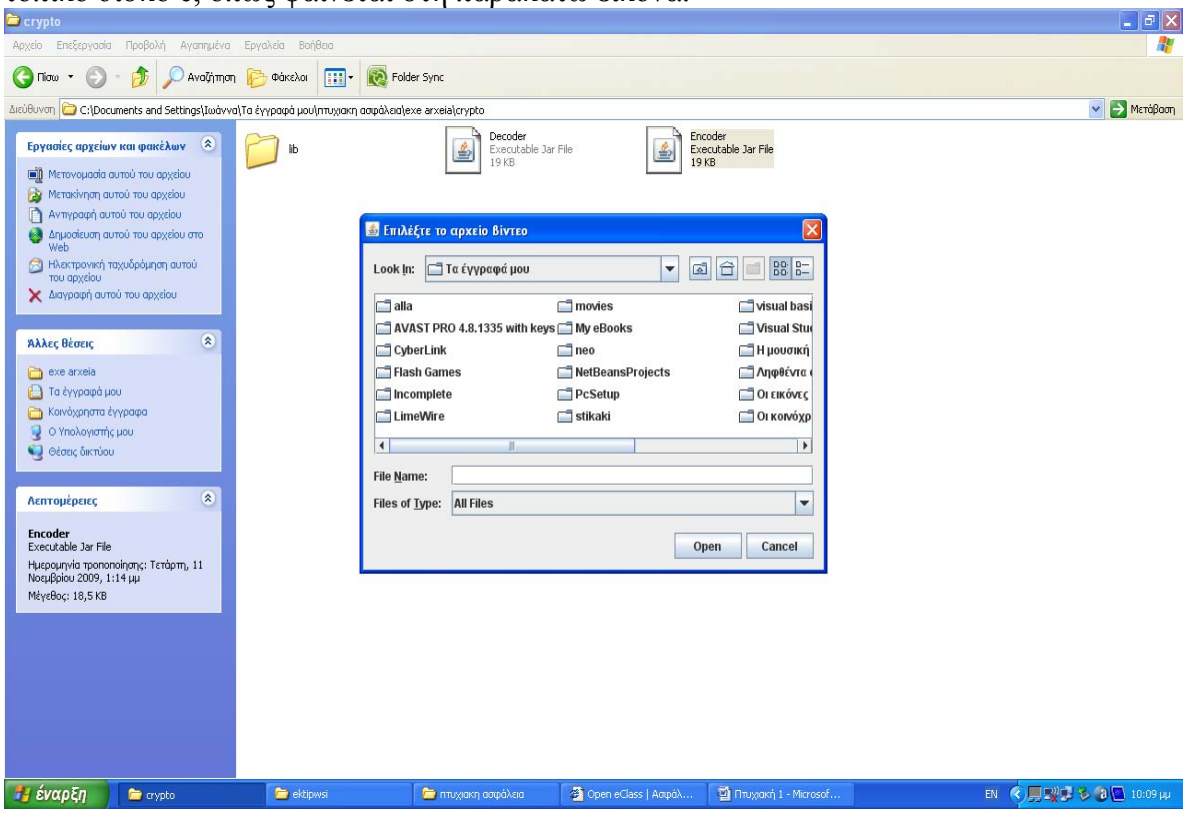
Το παράθυρο με το μήνυμα παραμένει, ενώ το βίντεο συνεχίζει να παίζει.

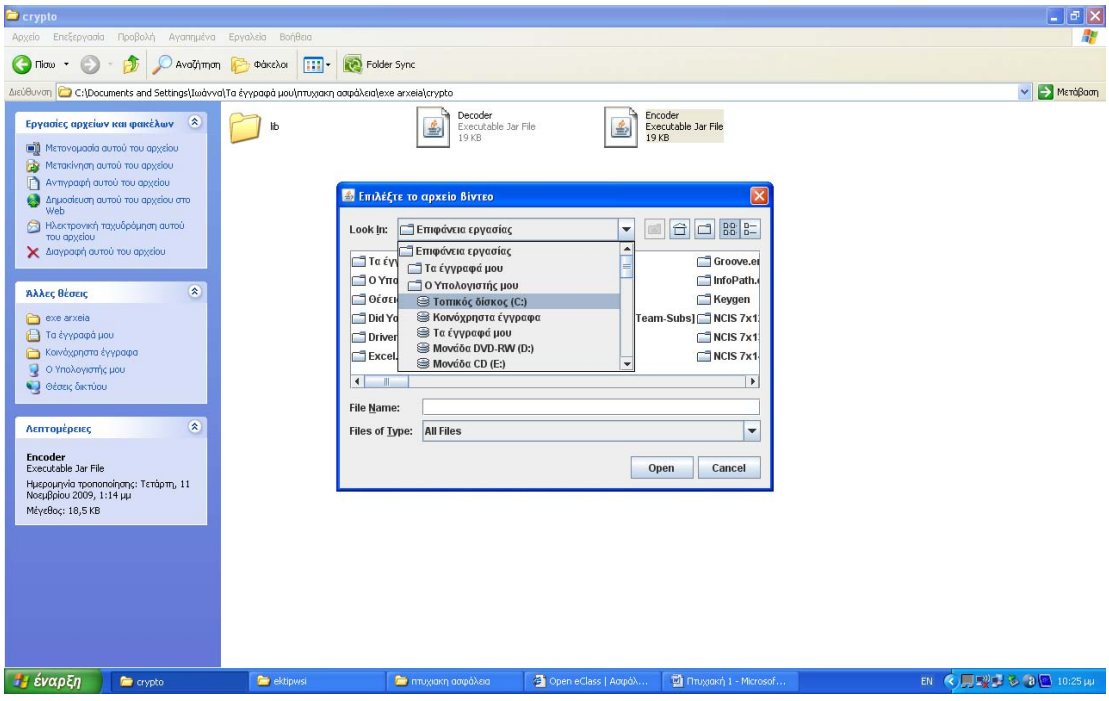
Όταν τελειώσει η αναπαραγωγή του βίντεο, μας ενημερώνει με νέο μήνυμα. Πατώντας οκ κλείνουν όλες οι διεργασίες και όλα τα παράθυρα.



Η ίδια διαδικασία μπορεί να πραγματοποιηθεί όταν στο decoder ή στο encoder κάνουμε δεξί κλίκ και επιλέξουμε run file.

Αν κάποιος είναι στο φάκελο με τα executable αρχεία και κάνει διπλό κλίκ στο encoder του εμφανίζεται το παράθυρο όπου στο βήμα αυτό θα πρέπει να διαλέξει τοπικό δίσκο c, όπως φαίνεται στη παρακάτω εικόνα.





Αν κάποιος επιλέξει το κουμπί cancel όπως δείχνει η παραπάνω εικόνα, τότε θα βγάλει το παρακάτω μήνυμα λάθους, ότι δεν έχει επιλεχθεί αρχείο βίντεο και το πρόγραμμα θα τερματιστεί.

EXECUTABLE ΑΡΧΕΙΑ

Έχω δημιουργήσει και executable αρχεία για να μπορεί κάποιος να τα τρέξει κατευθείαν χωρίς να χρειάζεται να έχει ανοίξει το netbeans και το τρέξει από εκεί ανοίγοντας κάθε κλάσση ξεχωριστά. Τα executable αρχεία συνοδεύονται και από το παρακάτω κείμενο προκειμένου να καταφέρετε να το τρέξετε με επιτυχία.

1.Λοιπόν το πρώτο που κάνετε είναι να αποσυμπιέσετε το αρχείο crypto. Πάτε στο τοπικό δίσκο c και δημιουργείτε εκεί ένα folder πχ video στο οποίο βάζετε μέσα το βίντεο που θέλετε να χρησιμοποιήσετε.(έγω έχω το wall-e.flv)

Το αρχείο crypto έχει μέσα δύο executable αρχεία το decoder και το encoder.

2. Κάνετε διπλό κλίκ στο encoder το οποίο ανοίγει ένα παράθυρο ,επιλέγεται το τοπικό δίσκο C και από εκεί επιλέγεται το αρχείο video όπου μέσα βρίσκεται το βίντεο που έχετε. Κάνοντας κλίκ πάνω του σας βγάζει το παράθυρο και σας ζητάει να εισάγεται το μήνυμα που θέλετε να βάλετε.

Γράφετε ένα μήνυμα και μετά οκ. Διαβάζετε το μήνυμα και κοιτάτε που θα σας αποθηκεύσει το νέο αρχείο, πατάτε οκ και περιμένετε μέχρι να σας βγάλει μήνυμα ότι ολοκληρώθηκε η διαδικασία .

3.Έπειτα πάτε στο decoder το ανοίγετε, διαλέγετε τοπικό δίσκο c, το φάκελο video αλλά τώρα το αρχειο .avi που θα πρέπει να έχει δημιουργήσει με βάση τα προγράμματα που έχετε εγκαταστήσει στον υπολογιστή για να κάνουν τη μετατροπή.

Όταν επιλέξετε το avi σας βγάζει το βίντεο αλλά και το παράθυρο που έχει το μήνυμα που είχατε γράψει στο προηγούμενο βήμα .Όταν τελειώσει το βίντεο κλείνουν και όλα τα παράθυρα.

ΚΩΔΙΚΟΠΟΙΗΣΗ ΚΑΙ ΑΠΟΚΩΔΔΙΚΟΠΟΙΗΣΗ ΒΙΝΤΕΟ

(λίγα λόγια για το πως λειτουργούν)

ΚΩΔΙΚΟΠΟΙΗΣΗ ΒΙΝΤΕΟ

Οι όροι κωδικοποίηση και αποκωδικοποίηση που αναφέρονται στο παρακάτω κείμενο δεν αφορούν στην κρυπτογραφία αλλά στην διαδικασία ανάγνωσης δεδομένων από ένα αρχείο βίντεο και κατάλληλης μετατροπής τους σε εικόνες για την προβολή τους.

Οι περισσότεροι τύποι αρχείων(file format) που χρησιμοποιούνται για την αποθήκευση βίντεο δεν περιέχουν άμεσα ένα βίντεο, αλλά είναι απλά containers, δηλαδή μπορούν να περιέχουν μέσα τους πολλά διαφορετικά ήδη δεδομένων.

Αυτά τα δεδομένα χωρίζονται σε ροές (streams) με κάθε ροή να μπορεί να περιέχει μόνο βίντεο ή μόνο ήχο(πχ τα αρχεία avi) ενώ κάποιες από τις πιο πρόσφατες μορφές τέτοιων αρχείων επιτρέπουν και την ύπαρξη άλλου τύπου ροών (πχ τα αρχεία mkv επιτρέπουν και ροές υπότιτλων).

Με αυτόν τον τρόπο είναι δυνατή η ενσωμάτωση στο ίδιο αρχείο δύο ή και περισσότερων διαφορετικών βίντεο, κάθε ένα από τα οποία μπορεί για παράδειγμα να είναι το ίδιο πλάνο από διαφορετική κάμερα κοκ. Επιπλέον είναι εφικτή η ενσωμάτωση πολλών διαφορετικών ροών ήχου που μπορεί να είναι πχ οι διάλογοι μεταγλωττισμένοι σε διάφορες γλώσσες κοκ.

Γενικά, αν ένα αρχείο βίντεο διαρκεί ένα συγκεκριμένο χρονικό διάστημα, κάθε ροή που αυτό περιέχει διαρκεί και αυτή από μία ώρα. Οι ροές δεν είναι συνεχόμενες αλλά ταυτόχρονες, δεν έχει νόημα να προβάλλεται η μία μετά από την άλλη. Για κάθε συγκεκριμένη χρονική στιγμή, πχ το $15^{ο}$ λεπτό, ο χρήστης μπορεί να επιλέξει τόσο την ροή βίντεο που θα δει (πχ λήψη από την πρώτη ή την δεύτερη κάμερα, επιλέγοντας ανάλογα την $1^{η}$ ή την $2^{η}$ ροή) όσο και την ροή ήχου που επιθυμεί.

Για αυτό τον λόγο η πλειοψηφία των προγραμματιστικών βιβλιοθηκών που ασχολούνται με την επεξεργασία βίντεο θεωρούν ότι κάθε αρχείο έχει πολλές ροές, άσχετα από την μορφή του. Ακόμα και για το απλούστερο βίντεο, δηλαδή αυτό που περιέχει μία ροή βίντεο και μία ήχου, είναι απαραίτητο να επιλεχθεί μία συγκεκριμένη ροή την οποία θα επεξεργαστεί στην πορεία. Αυτή την τακτική ακολουθεί η βιβλιοθήκη Xuggler αλλά και το ffmpeg το οποίο χρησιμοποιείται εσωτερικά από το Xuggler.

ΑΝΑΠΑΡΑΓΩΓΗ ΒΙΝΤΕΟ

Μια τυπική εφαρμογή αναπαραγωγής βίντεο είναι απαραίτητο να ακολουθήσει κάποια συγκεκριμένα βήματα για να μπορέσει να παίξει κάποιο αρχείο, τα οποία αναφέρονται παρακάτω. Η εφαρμογή που αναπτύξαμε υλοποιεί και αυτή την συγκεκριμένη διαδικασία.

Αρχικά η εφαρμογή ανοίγει το αρχείο βίντεο και επιλέγει τις ροές περιεχομένου που επιθυμεί να χρησιμοποιήσει. Αυτό χρειάζεται ακόμα και αν υπάρχει μόνο μία ροή, περίπτωση στην οποία επιλέγεται η πρώτη ροη (που συνήθως είναι η ροή 0).

Έπειτα ελέγχει ποια είναι η κωδικοποίηση του βίντεο, δηλαδή ο τρόπος αποθήκευσης και συμπίεσης των εικόνων και αν στο σύστημα είναι εγκατεστημένος ο αντίστοιχος αποκωδικοποιητής βίντεο(video decoder). Σε περίπτωση που υπάρχει ξεκινάει η αναπαραγωγή.

Με την διαδικασία της αναπαραγωγής η εφαρμογή ζητάει από τον decoder κάθε φορά το επόμενο καρέ(frame), το οποίο αυτός αναλαμβάνει να επιστρέψει στην εφαρμογή σαν απλή εικόνα, δηλαδή σαν σύνολο pixel άσχετα από τον τρόπο με τον οποίο είναι εσωτερικά αποθηκευμένο στο αρχείο.

Έχοντας το επόμενο καρέ σαν εικόνα η εφαρμογή το προβάλει στην οθόνη. Το πρόβλημα είναι πλέον ο χρόνος προβολής. Σε κάθε αρχείο περιέχεται πληροφορία για τον αριθμό των καρέ ανά δευτερόλεπτο . Πρόκειται για ένα σταθερό νούμερο που συνήθως έχει τιμές 23,96 ή 25. Για να προβληθεί το βίντεο σε κανονική ταχύτητα και όχι πιο γρήγορα ή πιο αργά από το φυσιολογικό η εφαρμογή χρειάζεται να φροντίσει να προβάλλει τα καρέ σε κατάλληλα χρονικά διαστήματα ώστε ο αριθμός καρέ που προβάλλει ανά λεπτό να συμπίπτει με τον παραπάνω.

11. Η γλώσσα προγραμματισμού java

Για τη δημιουργία του προγράμματος χρησιμοποίησα γλώσσα προγραμματισμού java, το πρόγραμμα xuggler και ffmpeg.

Xuggler είναι μια **δωρεάν και ανοικτής πηγής** βιβλιοθήκη για Java ή C + + προγραμματιστές που επιτρέπει την αποκωδικοποίηση, το χειρισμό, και την κωδικοποίηση (σχεδόν) οποιοδήποτε τύπο αρχείου βίντεο σε σχεδόν πραγματικό χρόνο. Είναι για τους προγραμματιστές που θέλουν να προσθέσουν βίντεο στήριξη για τη μεταποίηση των προϊόντων τους. Xuggler έρχεται επίσης με μια βιβλιοθήκη ολοκλήρωση Red5 το οποίο επιτρέπει στους προγραμματιστές Java την τροποποίηση και την εκ νέου μετάδοση βίντεο χρησιμοποιώντας Red5. Αυτό σημαίνει ότι μπορούμε να γράψουμε εφαρμογές που τροποποιούν το βίντεο.

Ενσωματώνει FFMPEG, αλλά προσθέτει ένα απλούστερο περιβάλλον εργασίας που είναι ασφαλές για χρήση από Java γλώσσες. Είναι μία πηγή Ffmpeg που επιτρέπει κωδικοποίηση με ασφάλεια και αποκωδικοποίηση σε αρχεία πολυμέσων Java ή C + + προγράμματα. Λειτουργεί με ασφάλεια μέσα σε μια διαδικασία Java.

Το ffmpeg είναι ένα πρόγραμμα με το οποίο μπορούμε να καταγράφουμε τη μετατροπή και τη ροή ήχου και βίντεο σε πολλές μορφές. Είναι ένα command line εργαλείο που αποτελείται από μία συλλογή δωρεάν λογισμικού και ανοιχτού κώδικα βιβλιοθήκες. Είναι ένα εργαλείο το οποίο, στην απλούστερη μορφή του, θέτει σε εφαρμογή έναν αποκωδικοποιητή και στη συνέχεια ένα κωδικοποιητή, δίνοντας έτσι τη δυνατότητα στο χρήστη να μετατρέψετε αρχεία από τη μία μορφή στην άλλη. Με το ffmpeg μπορούμε επίσης να κάνουμε κάποιους βασικούς χειρισμούς σχετικά με τα δεδομένα ήχου και βίντεο. Οι χειρισμοί περιλαμβάνουν την αλλαγή της συχνότητας δειγματοληψίας του ήχου και την προώθηση ή την καθυστέρηση σε σχέση με αυτό το βίντεο.

11.1 Η γλώσσα προγραμματισμού java :

Λίγα λόγια για την αντικειμενοστραφή γλώσσα πραγραμματισμού java.
Η Java είναι δημιούργημα της SUN MICROSYSTEMS.
http://java.sun.com
Πρωτοαναπτύχθηκε από τον James Gosling το 1990, για να χρησιμοποιηθεί σε "έξυπνες" συσκευές.
Οι ιδιότητες της γλώσσας σύντομα την έκαναν κατάλληλη για χρήση στο
World Wide Web, ενώ το όνομά της έγινε Java.
Τελευταίες εκδόσεις Java 2 SE 5.0
Όταν το World Wide Web εμφανίστηκε στο διαδίκτυο το 1993,
παρουσιάστηκε η ανάγκη μιας γλώσσας η οποία να είναι ανεξάρτητη πλατφόρμας.
Η Java έχει δημιουργηθεί να λειτουργεί σε πολλαπλά συστήματα.

Εισαγωγή στα βασικά της JAVA :

Η μεταβλητή είναι μια οντότητα η οποία μπορεί να παίρνει διαφορετικές τιμές. Επίσης δύναται να αλλάζει τιμές στη ροή του προγράμματος.
Για παράδειγμα, double a=1.35; int b=32; a = 7.5; b = 10;
Η σταθερά είναι μια οντότητα της οποίας η τιμή δεν αλλάζει ποτέ. Κάθε σταθερά ορίζεται μία φορά στο πρόγραμμα. Οι σταθερές γράφονται με κεφαλαία γράμματα. Int DAY=7; double GRAVITY=9.81;

Μία παράσταση είναι ένας συνδυασμός σταθερών, μεταβλητών, τελεστών ή και συναρτήσεων. Οι παραστάσεις χρησιμοποιούνται για να δηλώσουν **υπολογισμούς**. Για παράδειγμα, σκεφτείτε τις παρακάτω γραμμές:
Int a = 10; int b = 5; int c = (a+5)*b;
Ο όρος (a+5)*b αποτελεί μια παράσταση. Η μεταβλητή c λαμβάνει την τιμή75.
Μία **εντολή** είναι μια οδηγία η οποία τελειώνει με ένα **ερωτηματικό**. Στο προηγούμενο παράδειγμα η γραμμή c = (a+5)*b; αποτελεί μια εντολή. Η τιμή 75 η οποία υπολογίζεται από την παράσταση(a+5)*b προσδίδεται στην μεταβλητή c.

Οι βασικοί τύποι δεδομένων στη Java είναι
boolean false ή true
char 16 Όλοι οι Unicode χαρακτήρες
±4.94065645841246x10-324 έως
±1.79769313486231x10308
Double 64
Float 32 ±1.401298x10-45 έως ±3.402823x1038
Long 64 ± 9223372036854775807
Int 32 -2146473648 έως 2147483647
Short 16 -32768 έως 32767
byte 8 -128 έως 127

Οι αριθμητικοί τελεστές είναι οι ακόλουθοι:
% Υπόλοιπο / Διαίρεση * Πολλαπλασιασμός - Αφαίρεση + Πρόσθεση
<= Μικρότερο ή ίσο , < Μικρότερο , >= Μεγαλύτερο ή ίσο ,> Μεγαλύτερο
!= Άνισο με ,== Ίσο με

Τους συσχετιστικούς τελεστές, τους τελεστές ισότητας και τους λογικούς τελεστές τους συναντάμε κυρίως στις εντολές **if**, **for**, **while**, **do**. Οι παραπάνω τελεστές χρησιμοποιούνται για συγκρίσεις μεταξύ αριθμών, μεταβλητών και παραστάσεων.
Εάν η σύγκριση είναι **αληθής** τότε το αποτέλεσμα είναι 1 διαφορετικά εάν είναι **ψευδής** τότε το αποτέλεσμα είναι μηδέν.
! Λογικός τελεστής NEGATION
|| Λογικός τελεστής OR
&& Λογικός τελεστής AND

Ο τελεστής αύξησης ++ και ο τελεστής μείωσης -- χρησιμοποιούνται όταν θέλουμε να προσθέσουμε ή να αφαιρέσουμε το 1 από μία μεταβλητή.
Έτσι το ++a; ισοδυναμεί με το a=a+1;
ενώ το --a; ισοδυναμεί στο a=a-1;
οι τελεστές ++ και -- μπορούν να χρησιμοποιηθούν είτε ως **προθεματικοί** τελεστές (δηλ. πριν την μεταβλητή, όπως ++a ή--a) είτε ως **μεταθεματικοί**(δηλ. μετά την μεταβλητή, όπως a++ ή a--).

Στην παράσταση ++a η τιμή του a αυξάνει πριν χρησιμοποιηθεί η τιμή της.
Στην παράσταση a++ η τιμή του a αυξάνει αφού χρησιμοποιηθεί η τιμή της.
Παράδειγμα: Έτσι έστω ότι το a ισούται με 5 τότε η a = 5; τότε η παράσταση b = a++; δίνει στο b την τιμή 5 ενώ η παράσταση b=++a; την τιμή 6. Το a και στις δύο περιπτώσεις γίνεται 6.

Οι τελεστές αντικατάστασης είναι οι ακόλουθοι
Το a += b; ισοδυναμεί με το a = a+b;
Το a -= b; ισοδυναμεί με το a = a-b;
Το a *= b; ισοδυναμεί με το a = a*b;
Το a /= b; ισοδυναμεί με το a = a/b;
Το a %= b; ισοδυναμεί με το a = a %b;
Τελεστής υπολοίπου **%=** και αντιστοίχησης
/= Τελεστής διαίρεσης και αντιστοίχησης
*= Τελεστής πολ/μου και αντιστοίχησης
-= Τελεστής αφαίρεσης και αντιστοίχησης
+= Τελεστής πρόσθεσης και αντιστοίχησης

Η κλάση Math περιέχει μεθόδους με τις οποίες μπορούμε να κάνουμε βασικές πράξεις με εκθετικά, λογαρίθμους, τετραγωνικές ρίζες και τριγωνομετρικές συναρτήσεις. Παραδείγματα:
double c=Math.pow(a,b); αντιστοιχεί σε c=ab
Double b=Math.cos (a); αντιστοιχεί σε b=cos (a)
Double b=Math.abs (a); αντιστοιχεί σε b=|a|

Χαρακτήρες char c='A';
Οι συμβολοσειρές (Strings) είναι ακολουθίες χαρακτήρων
String uoi="University of Ioannina";
Μια συμβολοσειρά στην Java είναι ένα αντικείμενο της κλάσης String.
Εμφάνιση συμβολοσειρών. Μέθοδος println() με αλλαγή γραμμής στην
Εκτύπωση. Εμφάνιση συμβολοσειρών.
Μέθοδος print() χωρίς αλλαγή γραμμής στην εκτύπωση
System.out.println("University");
System.out.println(" of ");
System.out.println("Ioannina");
Συνένωση συμβολοσειρών
String uoipd = uoi + " Physics Department";

Έστω οι συμβολοσειρές s1 και s2.
Μετατροπή συμβολοσειράς s2=s1.toLowerCase() s1 σε πεζά
s1.charAt(4) Επιστροφή του χαρακτήρα της θέσης πχ. 4
s1.replace('B','C') Αντικατάσταση του 'B' με το 'C'
Αναζήτηση συμβολοσειράς s2 στην s1
a η θέση της s2 στην s1
int a=s2.indexOf(s1);
s2.equals(s1) Σύγκριση συμβολοσειρών s1 και s2
s2=s1.toUpperCase() Μετατροπή συμβολοσειράς s1 σε κεφαλαία
s1.length() Προσδιορισμός μήκους συμβολοσειράς s1

Τα αντικείμενα της κλάσης StringBuffer είναι συμβολοσειρές οι οποίες δύνανται να μεταβάλλονται.
StringBuffer ss = new StringBuffer(20); // Μήκος 20 χαρακτήρες
Μερικές μέθοδοι της κλάσης StringBuffer
ss.length() //Προσδιορισμός μήκους
ss.capacity() //Προσδιορισμός χωρητικότητας
ss.setLength() //Επαναπροσδιορισμός μήκους
ss.reverse() //Αντιστροφή συμβολοσειράς
ss.append() //Επέκταση της συμβολοσειράς
ss.setCharAt() //Αλλαγή χαρακτήρα σε μια θέση
Για την εισαγωγή δεδομένων από τη γραμμή εντολών εκμεταλλευόμαστε τον πίνακα συμβολοσειράς String[] arguments ο οποίος αποτελεί το όρισμα της μεθόδου main(). Οι συμβολοσειρές τις οποίες εισάγουμε από την γραμμή εντολών αποθηκεύονται με την σειρά στα στοιχεία του πίνακα arguments (δηλαδή στο arguments[0], στο arguments[1] στο arguments[2] κτλ.).

Κώδικας Steganography πίσω από τη java :
Οι δύο κύριες λειτουργίας στην εφαρμογή μας είναι κωδικοποίηση και αποκωδικοποίηση. Για χάρη της υλοποίησής του συστήματος αυτές τις δύο διαδικασίες τις έχουμε ονομάσει hider και unhider αντίστοιχα.
Αυτό που γίνεται τώρα είναι το εξής:
Ο Encoder, δηλαδή η κλάσση Encoder που τρέχει κάνοντας διπλό κλικ στο Encoder είναι υπεύθυνος για να κρύβει το μήνυμα. Για την ακρίβεια ανοίγει το βίντεο, σου ζητάει το μήνυμα και το αποθηκεύει σε ένα νέο αρχείο που περιέχει και το μήνυμα.

Η κλάσση Encoder χοντρικά ανοίγει το βίντεο και το επανασυμπιέζει. Χρησιμοποιεί το BitHider για να κρύψει μέσα στα pixels το μήνυμα.

Ο Decoder, δηλαδή η κλάσση Decoder παίζει το βίντεο και βρίσκει σε αυτό το μήνυμα αν υπάρχει. Χρησιμοποιεί το BitUnhider για να βγάλει από τα pixels του βίντεο το μήνυμα. Επειδή το μήνυμα έχει γραφτεί από τον Encoder στο Wall-e(1).avi χρειάζεται να ανοίξουμε αυτό το αρχείο από τον Decoder.
Η κλάσση Decoder ανοίγει το βίντεο και το δείχνει στην οθόνη. Χρησιμοποιεί την κλάσση BitUnhider για να βγάλει το μήνυμα από τα pixels του βίντεο.

```
package hider;
```

* Ένα αντικείμενο που δημιουργείται με παράμετρο ένα String, το οποίο είναι το μήνυμα που θέλουμε να κρύψουμε.

* Αφού το δημιουργήσουμε καλούμε την μέθοδο hide με παράμετρο έναν ακέραιο. Η συνάρτηση κρύβει δύο bit μέσα στον ακέραιο.

* Τα pixels του βίντεο που παίρνουμε από το xuggle είναι πράκτικα ένας ακέραιος, δηλαδή 32 bit. Τα πρώτα 8 bit δεν χρησιμοποιούνται, ενώ οι υπόλοιπες 3 οκτάδες είναι τα χρώματα RGB που περιέχει το pixel.

* Η συνάρτηση αποθηκεύει για κάθε ακέραιο (pixel) την κρυμμένη πληροφορία στο LSB (less significant bit, πρακτικά στο αριστερό) κάθε οκτάδας.

* Έτσι τα bits κάθε pixel είναι της μορφής:

XXXXXXXX RRRRRRRR GGGGGGGG BBBBBBBB

Όπου:

* X δεν χρησιμοποιείται

* R Τα bit για το κόκκινο

* G Τα bit για το πράσινο

* B Τα bit για το μπλε

* Μετά την μετατροπή γίνονται:

XXXXXXXX RRRRRRRR GGGGGGGI BBBBBBBI

Όπου:

* I Τα bits που κρύβω

* Αν και θα μπορούσαμε να αποθηκεύουμε 3 bit σε κάθε ακέραιο, κάτι τέτοιο θα έκανε πολύ ποιο δύσκολη την εξαγωγή του μηνύματος.

```
public class BitHider {
```

*Αποθηκεύουμε το μήνυμα σαν πίνακα bytes γιατί είναι απλούστερο στον χειρισμό του.

```
private byte[] message;
```

* Η θέση του bit που θέλουμε να αποθηκεύσουμε.

```
private int index = 0;

public BitHider(String message) {
```

*Για να ξέρω από αρχίζει και που τελειώνει ένα μήνυμα όταν το αποκωδικοποιώ προσθέτω στην αρχή και στο τέλος του τα #.

```
message = message + "#END#";

this.message = message.getBytes();

System.out.println(this.message.length);   }


public int hide(int target){
```

*Παίρνω τα δύο επόμενα bit που θα βάλω στον target

```
int nextBit1 = getNextBit();  // Θα μπει στην δεξιότερη θέση, στο blue

int nextBit2 = getNextBit();  //  Θα μπει στο green
```

*Μηδενίζω τις θέσεις που θα κάνω τις εισαγωγές.

Το 0xFFFFFEFE είναι 11111111 11111111 11111110 11111110

* Το AND με το target θα αφήσει άθικτα όλα τα ψηφία του target εκτός από τις θέσεις που είναι 0, οι οποίες θα μηδενιστουν.

```
target = target & 0xFFFFFEFE;
```

Βάζω το 1ο bit.

Πχ αν το nextBit1 είναι 1, για ένα τυχαίο target:

11111111 11011011 11010110 11101000

| 00000000 00000000 00000000 00000001

11111111 11011011 11010110 11101001

```
target = target | nextBit1;
```

*Πάω την τιμή του nextBit2 8 θέσεις αριστερά.

Πχ αν είναι 1:

00000000 00000000 00000000 00000001 << 8 θα δώσει

00000000 00000000 00000001 00000000

```
nextBit2 = nextBit2 << 8;
```

Βάζω και το 2ο bit στο target.

```
target = target | nextBit2;

return target;

}
```

*Βοηθητική συνάρτηση που επιστρέφει το επόμενο bit. Αν και επιστρέφει int, αφού στην java δεν υπάρχει τύπος bit, η τιμή επιστροφής είναι πάντα 0 ή 1.

```
@return


public int getNextBit(){
```

*Υπολογίζω την θέση που βρίσκεται το bit που θέλω στον πίνακα.

* Πχ

αν θέλω το 11ο bit αυτό βρίσκεται στο στοιχείο 1 του πίνακα,δηλαδή arIndex = 11/8 = 1 (Ακέραια διαίρεση), ενώ είναι το 3ο bit μέσα σε αυτό το byte, δηλαδή bitIndex = 11 % 8 = 3;

*Αν το bitIndex είναι 3 σημαίνει ότι θέλω το 4ο bit ΑΠΟ ΤΑ ΔΕΞΙΑ.

```
int arIndex = index / 8;

int bitIndex = index % 8;

int result = message[arIndex];

System.out.println("i: " + index + " arI: " + arIndex + "
bI: " + bitIndex);
```

* Μετακυλάω αυτό το bit αριστερά, ώστε να σβήσω τυχόν bit που υπάρχουν αριστερά του.

Πχ αν θέλω το 4ο bit (bitIndex = 3) τότε το byte θα είναι της μορφής:

XXXX1XXX, όπου X αδιάφορα για μένα bits.

* Θα το πάω 4 θέσεις προς τα αριστερά. Το 4 βγαίνει σαν 7 - bitIndex = 7-3 = 4, οπότε θα γίνει: 1XXX0000, αφού η μετακίνηση βάζει στην θέση των κενών χαρακτήρων 0.

```
result = (result << 7 - bitIndex);
```

```
System.out.print("==> " + Integer.toBinaryString(result)
+ " - ");
```

* Τώρα πρέπει να καθαρίσω τους αδιάφορους χαρακτήρες που υπάρχουν και στα δεξιά του, οπότε το μετακυλάω τέρμα δεξιά, 7 θέσεις, οπότε θα γίνει 00000001

```
result = (result >> 7);
```

* Οι ακέραιοι στην java αναπαριστόνται εσωτερκά σαν συμπλήρωμα του 2.

* Αυτό σημαίνει ότι πχ το 0 αντί να αποθηκεύεται σαν 0000...01 αποθηκεύεται σαν 11111111 11111111 11111111 11111110, ενώ το 1 αποθηκεύεται σαν 11111111 11111111 11111111 11111111.

Επειδή θέλω μόνο το τελευταίο bit κάνω λογικό AND με το 1, δηλαδή:

result 11111111 11111111 11111111 11111110

AND 00000000 00000000 00000000 00000001

00000000 00000000 00000000 00000000

* Ομοίως αν

result 11111111 11111111 11111111 11111111

AND 00000000 00000000 00000000 00000001

00000000 00000000 00000000 00000001

```
result = (result & 1);
index++;
if(index%8==0){
System.out.print(" ");
}
return result;
}
```

* Επιστρέφει true όσο έχω ακόμη bits να αποθηκεύσω.

```
* @return
public boolean hasMore(){
if(index/8 >= message.length){
return false;
}else{
return true;
}
}
}
```

```
package hider;

import java.util.ArrayList;
```

*Αντικείμενο το οποίο ανακτά τα κρυμμένα bits.Εδώ δημιουργείται μία άλλη κλάσση η bit unhider.

```
 public class BitUnhider {
```

* Μία λίστα που κρατάει τα bits που έχουν ανακτηθεί μέχρι τώρα.

* Τα bits αποθηκεύονται σαν ακέραιοι αφού είναι 0 ή 1.

```
private ArrayList<Integer> bitList = new
ArrayList<Integer>();

public BitUnhider(){

}
```

*Συνάρτηση που παίρνει για παράμετρο έναν ακέραιο, βγάζει το 1ο και το 9ο bit και τα αποθηκεύει στην λίστα.

```
public void unhide(int target){
```

* Κάνω 0 όλα τα bits του target εκτός από το τελευταίο

Πχ

XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX

& 00000000 00000000 00000001 00000000

(Το 0x100 στο δυαδικό)

00000000 00000000 0000000X 00000000

```
int b1 = target & 0x100;
```

* Πάω το παραπάνω bit 8 θέσεις προς τα δεξιά.

Πχ

00000000 00000000 0000000X 00000000 >> 8 =

00000000 00000000 00000000 0000000X

```
b1 = b1 >> 8;
```

* Κάνω 0 όλα τα bits του target εκτός από το τελευταίο

Πχ

XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX

& 00000000 00000000 00000000 00000001

(Το 1 στο δυαδικό)

00000000 00000000 00000000 0000000X

```
int b2 = target & 1;
```

Τα αποθηκεύω στην αρχή της λίστας.

```
bitList.add(0, b2);

bitList.add(0, b1);

}

public boolean hasMessageFinished(){

if(bitList.size() >= 16*8)

return true;

if(bitList.size() != 0 && bitList.size() %8 == 0 &&
getMessage().endsWith("#END#")){

return true;

}

return false;

}
```

```
public String getMessage(){
ArrayList<Byte> byteList = new ArrayList<Byte>();
String tempByte = "";
for(int i=0; i<bitList.size();i++){
tempByte = tempByte + bitList.get(i);
if((i+1)%8==0){
int temp = Integer.parseInt(tempByte);
String ts = Integer.toBinaryString(temp);
System.out.println(ts + " ");
byteList.add(0, (byte)Integer.parseInt(tempByte, 2) );
tempByte = "";
}
}

byte[] byteArray = new byte[byteList.size()];
for(int i=0; i<byteArray.length; i++){
byteArray[i] = byteList.get(i);
}
String result = new String(byteArray);
return result;
}
}
```

```
Package xuggleproject;

import java.awt.image.BufferedImage;
import com.xuggle.mediatool.IMediaReader;
import com.xuggle.mediatool.IMediaWriter;
import com.xuggle.mediatool.MediaListenerAdapter;
import com.xuggle.mediatool.ToolFactory;
import com.xuggle.mediatool.event.IAddStreamEvent;
import com.xuggle.mediatool.event.IVideoPictureEvent;
import com.xuggle.mediatool.event.VideoPictureEvent;
import com.xuggle.xuggler.IStreamCoder;
import com.xuggle.xuggler.ICodec.ID;
import com.xuggle.xuggler.ICodec.Type;
import com.xuggle.xuggler.IPixelFormat;
import hider.BitHider;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

public class Encoder {

public static void main(String args[]){
String inputFile = "";
String outputFile = "";
String info  = "";
```

*Δημιουργία και εμφάνιση παραθύρου επιλογής αρχείου. Αν επιλεγεί αρχείο έξοδος, αλλιώς συνέχεια.

```
JFileChooser chooser = new JFileChooser();

chooser.setDialogTitle("Επιλέξτε το αρχείο βίντεο");

int returnVal = chooser.showOpenDialog(null);

if (returnVal == JFileChooser.APPROVE_OPTION) {

String path =
chooser.getCurrentDirectory().getAbsolutePath() + "\\";

inputFile = path + chooser.getSelectedFile().getName();

outputFile=inputFile.substring(0,inputFile.lastIndexOf(".
")) + "(1).avi";

info = "Επιλέξατε να ανοίξατε το αρχείο '" + inputFile +

"'.\nΤο βίντεο με το ενσωματομένο μήνυμα θα γραφτεί στο
αρχείο '" + outputFile + "'.";

}else{

JOptionPane.showMessageDialog(null,"Δενεπιλέχθηκε αρχείο.
Το πρόγραμμα θα τερματιστεί.", "Λάθος"
,JOptionPane.ERROR_MESSAGE);

return;

}
```

*Εμφάνιση παραθύρου που ζητάει από τον χρήστη να εισάγει το κρυμμένο μήνυμα.

```
String message = JOptionPane.showInputDialog(null,
"Παρακαλώ εισάγετε το κρυμμένο μήνυμα:", "Εισαγωγή
μηνύματος", JOptionPane.QUESTION_MESSAGE);

if(message.length() == 0){

JOptionPane.showMessageDialog(null, "Δεν εισάγατε μήνυμα.
Το πρόγραμμα θα τερματιστεί.", "Κενό μήνυμα",
JOptionPane.ERROR_MESSAGE);

return;

}else{

info += "\nΤο μήνυμα που θα αποθηκευτεί είναι το '" +
message + "'\nΠατήστε OK για να ξεκινήσει η διαδικασία.";

}
```

*Εμφάνιση των τελικών επιλογών.

```
JOptionPane.showMessageDialog(null, info, "Ολοκλήρωση επιλογών", JOptionPane.INFORMATION_MESSAGE);
```

*Δημιουργία του BitHider που θα κρύψει το μήνυμα.

```
final BitHider hider = new BitHider(message);
```

*Δημιουργία του αντικειμένου reader που θα διαβάζει το μήνυμα και άνοιγμα του αρχείου.

```
final IMediaReader reader = ToolFactory.makeReader(inputFile);
```

*Ρύθμιση του reader να παράγει εικόνες των οποίων τα pixels περιέχουν 3 χρώματα (RGB)

```
reader.setBufferedImageTypeToGenerate(BufferedImage.TYPE_3BYTE_BGR);
```

*Δημιουργία του αντικειμένου writer που θα γράψει το νέο βίντεο αποτελέσματος.

```
final IMediaWriter writer = ToolFactory.makeWriter(outputFile, reader);
```

*Συσχετισμός του writer με έναν νέο listener (Βλέπε παρακάτω)

```
writer.addListener(new MediaListenerAdapter(){
```

Όταν ο writer προσθέσει στο αρχείο μια νέα ροή (βίντεο/ήχου) θα καλεσθεί η παρακάτω συνάρτηση

```
@Override

public void onAddStream(IAddStreamEvent event) {
```

*Παίρνω τον κωδικοποιητή (coder) για την νέα δομή που θα προστεθεί

```
IStreamCoder coder = event.getSource().getContainer().getStream(event.getStreamIndex()).getStreamCoder();
```

*Αν ο coder είναι κωδικοποιητής βίντεο(και όχι ήχου που δεν μας ενδιαφέρει)

```
if (coder.getCodecType().equals(Type.CODEC_TYPE_VIDEO)) {
```

*Αλλαγή του τύπου κωδικοποίησης σε μη απολεστικό και ρύθμιση του τύπου κωδικοποίησης ώστε τα pixels του νέου βίντεο να περιέχουν 3 χρώματα (RGB)

```
coder.setCodec(ID.CODEC_ID_RAWVIDEO);
coder.setPixelType(IPixelFormat.Type.BGR24)

}

}

});
```

*Συσχετισμός του reader με έναν νέο listener (Βλέπε παρακάτω)

```
reader.addListener(new MediaListenerAdapter(){
```

*Για κάθε frame του βίντεο που διαβάζει ο reader καλείται η παρακάτω συνάρτηση

```
@Override

public void onVideoPicture(IVideoPictureEvent arg0) {
```

*Παίρνουμε το τρέχον frame σαν αντικείμενο τύπου BufferedImage ώστε να μπορέσουμε να τροποποιήσουμε τα pixels του.

```
BufferedImage image = arg0.getImage();
```

*Διατρέχουμε κάθε pixel της εικόνας μέχρι ο hider έχει να κρύψει όλο το μήνυμα

```
for(int i=0; i<image.getWidth() && hider.hasMore(); i++){

for(int j=0; j<image.getHeight() && hider.hasMore();
j++){
```

*Καλούμε την μέθοδο hide του hider με παράμετρο το τρέχον pixel,ώστε να κρύψει σε αυτό το μήνυμα.

```
image.setRGB(i, j, hider.hide(image.getRGB(i, j)));

}

}
```

*Αποθηκεύουμε την τροποιημένη εικόνα.

```
VideoPictureEvent event = new
VideoPictureEvent(arg0.getSource(),image,
arg0.getTimeStamp(),arg0.getTimeUnit(),arg0.getStreamInde
x());

super.onVideoPicture(event);

}

});
```

*Συσχετίζουμε των writer με τον reader, ώστε ο writer να γράφει ότι διαβάζει ο reader

```
reader.addListener(writer);
```

Λέμε στον reader να διαβάσει όλο αρχείο

```
while(reader.readPacket() == null);
```

*Ενημερώνουμε τον χρήστη για το τέλος της διαδικασίας.

```
JOptionPane.showMessageDialog(null, "Η διαδικασία
ολοκληρώθηκε.", "Τέλος",
JOptionPane.INFORMATION_MESSAGE);

}

}
```

```
Package xuggleproject;

*Εισαγωγή των απαιτούμενων κλάσεων που θα χρησιμοποιήσει το πρόγραμμά.
import java.awt.image.BufferedImage;
import com.xuggle.xuggler.Global;
import com.xuggle.xuggler.IContainer;
import com.xuggle.xuggler.IPacket;
import com.xuggle.xuggler.IPixelFormat;
import com.xuggle.xuggler.IStream;
import com.xuggle.xuggler.IStreamCoder;
import com.xuggle.xuggler.ICodec;
import com.xuggle.xuggler.IVideoPicture;
import com.xuggle.xuggler.IVideoResampler;
import com.xuggle.xuggler.Utils;
import com.xuggle.xuggler.demos.VideoImage;
import hider.Unhider;
import java.awt.FlowLayout;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

public class Decoder {
private static JTextArea ta;

public static void main(String[] args) {
String filename = null;

*Δημιουργεί το παράθυρο επιλογής αρχείου, από το οποίο επιλέγουμε το αρχείο που θέλουμε να αναπαράγουμε.
JFileChooser chooser = new JFileChooser();

int returnVal = chooser.showOpenDialog(mScreen);
if (returnVal == JFileChooser.APPROVE_OPTION) {
filename = chooser.getSelectedFile().getAbsolutePath();
}

if(filename == null){
System.exit(0);
}

if (!IVideoResampler.isSupported(IVideoResampler.Feature.
FEATURE_COLORSPACECONVERSION)) {
throw new RuntimeException();
}
```

*Δημιουργία του παραθύρου στο οποίο θα εμφανίζεται το μήνυμα, με τίτλο Message

```
JFrame frame = new JFrame("Message");
frame.setLayout(new FlowLayout());
ta = new JTextArea(5, 20);
frame.add(new JScrollPane(ta));
ta.setText("Hidden message");
frame.pack();
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);

IContainer container = IContainer.make();

if (container.open(filename, IContainer.Type.READ,
null) < 0) {
throw new IllegalArgumentException("could not open file:
" + filename);
}
```

*Το βίντεο αποτελείται από πολλές ροές και πάει και παίρνει κάθε μια από αυτές ξεχωριστά. Παίρνουμε τον αριθμό των ροών βίντεο και ήχου που περιέχει το αρχείο, δηλαδή ο container και τις διατρέχουμε με την for.(Κάθε αρχείο είναι ένας περιέκτης(container) με ροές βίντεο και ήχου)

```
Int numStreams = container.getNumStreams ();

int videoStreamId=-1;
IstreamCodervideoCoder=null;
for (int i = 0; i < numStreams; i++) {
IStream stream = container.getStream (i);
IStreamCoder coder = stream.getStreamCoder ();
```

*Αν βρούμε ροή βίντεο την κρατάμε και σταματάμε την for, αφού αυτή μας ενδιαφέρει.

```
If (coder.getCodecType () ==
ICodec.Type.CODEC_TYPE_VIDEO) {
videoStreamId = i;
videoCoder = coder;
break;
}
}
```

*Αν δεν υπάρχει ροή βίντεο δεν μπορούμε να ψάξουμε για μήνυμα, πετάμε Exception.

```
If (videoStreamId == -1) {
throw new RuntimeException ("could not find video stream in container: " +
filename);
}
```

*Αν δεν μπορούμε να αποκωδικοποιήσουμε την συγκεκριμένη ροή βίντεο δεν μπορούμε να ψάξουμε για μήνυμα, πετάμε Exception.

```
 if (videoCoder.open() < 0) {
throw new RuntimeException("could not open video decoder for container: " +
filename);
}

IVideoResampler resampler = null;
if (videoCoder.getPixelType() != IPixelFormat.Type.RGB24) {
resampler = IVideoResampler.make(videoCoder.getWidth(), videoCoder.getHeight(),
IPixelFormat.Type.RGB24, videoCoder.getWidth(), videoCoder.getHeight(),
videoCoder.getPixelType());
if (resampler == null) {
throw new RuntimeException("could not create color space
resampler for: " + filename);
}
}
openJavaWindow();

IPacket packet = IPacket.make();
long firstTimestampInStream = Global.NO_PTS;
long systemClockStartTime = 0;

while (container.readNextPacket(packet) >= 0) {
if (packet.getStreamIndex() == videoStreamId) {
IVideoPicture picture =
IVideoPicture.make(videoCoder.getPixelType(),
videoCoder.getWidth(), videoCoder.getHeight());

int offset = 0;
while (offset < packet.getSize()) {
int bytesDecoded = videoCoder.decodeVideo(picture,
packet, offset);
if (bytesDecoded < 0) {
throw new RuntimeException("got error decoding video in:
" + filename);
}
```

```
offset += bytesDecoded;

if (picture.isComplete()) {
IVideoPicture newPic = picture;
if (resampler != null) {
newPic=IVideoPicture.make(resampler.getOutputPixelFormat(
), picture.getWidth(), picture.getHeight());

if (resampler.resample(newPic, picture) < 0) {
throw new RuntimeException("could not resample video
from: " + filename);
}
}

if (newPic.getPixelType() != IPixelFormat.Type.RGB24) {
throw new RuntimeException("could not decode video as RGB
32 bit data in: " + filename);
}

if (firstTimestampInStream == Global.NO_PTS) {
firstTimestampInStream = picture.getTimeStamp();
systemClockStartTime = System.currentTimeMillis();
} else {
long systemClockCurrentTime = System.currentTimeMillis();
long millisecondsClockTimeSinceStartofVideo =
systemClockCurrentTime - systemClockStartTime;
long millisecondsStreamTimeSinceStartOfVideo =
(picture.getTimeStamp() - firstTimestampInStream) / 1000;

final long millisecondsTolerance = 50; // and we give
ourselfs 50 ms of tolerance
final long millisecondsToSleep=(millisecondsStreamTimeSin
ceStartOfVideo -(millisecondsClockTimeSinceStartofVideo +
millisecondsTolerance));

if (millisecondsToSleep > 0) {
try {
Thread.sleep(millisecondsToSleep);
} catch (InterruptedException e) {
e.printStackTrace();
return;
}
}
}
```

```
BufferedImage javaImage =
Utils.videoPictureToImage(newPic);
IConverter
converter=ConverterFactory.createConverter(javaImage,
IPixelFormat.Type.ABGR.RGB24);

Unhider unhider = new Unhider();
for(int i=0; i<8; i++){
int rgb = javaImage.getRGB(120, 120 + i);
unhider.unhide(rgb);
}

if(unhider.reconstructMessage().endsWith("##")){
ta.setText(unhider.reconstructMessage());
}

updateJavaWindow(javaImage);
}
}
} else {
;
}

}

if (videoCoder != null) {
videoCoder.close();
videoCoder = null;
}

if (container != null) {
container.close();
container = null;
}
closeJavaWindow();

}
private static VideoImage mScreen = null;

private static void updateJavaWindow(BufferedImage
javaImage) {
mScreen.setImage(javaImage);
}
private static void openJavaWindow() {
mScreen = new VideoImage();
}
private static void closeJavaWindow() {
System.exit(0);
}
}
```

12.ΘΕΩΡΗΤΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ

Η Στεγανογραφία είναι μια αρχαία τέχνη που έχει διαδοθεί και αναπτυχθεί με την εμφάνιση του διαδικτύου και γενικά από τα ψηφιακά μέσα. Δεν είναι πλέον μια μέθοδος που περιορίζεται στη μυστική επικοινωνία μεταξύ δύο κατασκόπων ή κάποια άλλη χρήση της όπως στη διάρκεια του πολέμου. Τα εργαλεία είναι τώρα προσιτά στους χρήστες και σε αρκετές περιπτώσεις δωρεάν μέσω του διαδικτύου, τα οποία δεν απαιτούν καμία ειδική γνώση για τη χρήση τους.

Σε θέμα ασφάλειας η στεγανογραφία σε συνδυασμό με την κρυπτογραφία θα μπορούσε να πετύχει ένα πολύ καλό και υψηλό επίπεδο ασφαλείας και απορρήτου. Ο στόχος της στεγανογραφίας είναι να αποφύγει να χαράξει υποψία για τη μετάδοση του ενός κρυφού μηνύματος. Αν υποψία εγείρεται τότε ο στόχος αυτός έχει ηττηθεί. Αυτό μπορεί από τη μία να αυξάνει την δυνατότητα προστασίας της μυστικότητάς μας, ή την επικοινωνία μας όταν το απαιτούν οι συνθήκες, αλλά παράλληλα δίνει τη δυνατότητα στους εγκληματίες και στους τρομοκράτες να επικοινωνούν μεταξύ τους χωρίς να ανιχνεύονται από την δικαιοσύνη. Η απαγόρευση της τεχνολογίας δεν είναι επαρκής για να σταματήσει την εγκληματική χρήση.

Δεδομένου ότι η ανίχνευση δεν μπορεί να δώσει μια εγγύηση για την εξεύρεση όλων των κρυφών πληροφορίες, μπορεί να χρησιμοποιηθεί σε συνδυασμό με τις μεθόδους της νικώντας τη στεγανογραφία, για να ελαχιστοποιηθούν οι πιθανότητες κρυμμένης επικοινωνίας που λαμβάνει χώρα. Ακόμη και τότε, τέλεια στεγανογραφία θα περάσει απαρατήρητη, επειδή η πηγή κάλυψη δεν περιέχει πληροφορίες σχετικά με το μυστικό μήνυμα.

Η Στεγανάλυση ενώ θα μπορούσε να γίνει αποτελεσματική, αντιμετωπίζει πολλά εμπόδια για να αναγνωριστεί σαν μια αξιόπιστη μέθοδος ανίχνευσης στεγανογραφικής δραστηριότητας. Η Στεγανογραφία και η Στεγανάλυση είναι ακόμα σε στάδια έρευνας και ανάπτυξης. Δεδομένου ότι οι τεχνικές για το κρύψιμο πληροφορίας βελτιώνονται, η μόνη επιλογή είναι η συνεχής πρόοδος και έρευνα.

Αναφορές-Παραπομπές

http://www.eurojournals.com/ejsr_31_2_01.pdf

Attack on LSB Steganography in Color and Grayscale Images

http://www.cs.ucl.ac.uk/staff/ingemar/papers/2008/icip2008.pdf
Detection of +/-1 LSB Steganography based on the Amplitude

http://www.ijicic.org/05-054.pdf
A LSB STEGANOGRAPHY APPROACH AGAINST PIXELS SAMPLE PAIRS...

http://www.jatit.org/volumes/research-papers/Vol5No6/15Vol5No6.pdf
EFFICIENT METHOD OF AUDIO STEGANOGRAPHY BY MODIFIED LSB ALGORITHM...

http://www.infosecwriters.com/text_resources/pdf/Steganography_AMangarae.pdf

Steganography FAQ

http://www.mva-org.jp/Proceedings/2009CD/papers/13-14.pdf
An Optical **Video** Cryptosystem with Adaptive **Steganography**

http://www.krenn.nl/univ/cry/steg/article.pdf
Steganography and Steganalysis

http://www.jiit.ac.in/jiit/ic3/IC3_2008/IC3-2008/APP2_21.pdf
A Tutorial Review on **Steganography**

http://www.lncc.br/~borges/doc/Steganography%20with%20Public-Key%20Cryptography%20for%20Videoconference.pdf
Steganography with Public-Key Cryptography for Videoconference

http://www.waset.org/journals/waset/v54/v54-63.pdf
A Genetic-Algorithm-Based Approach for Audio **Steganography**

http://www.laccei.org/proceedings2004/FinalPapers/ET_019.pdf
Additional Data Security with Dynamic **Steganography**

http://txspace.tamu.edu/bitstream/handle/1969.1/3901/etd-tamu-2005A-ENGR-Budhia.pdf?sequence=1
STEGANALYSIS OF **VIDEO** SEQUENCES USING COLLUSION SENSITIVITY A...

http://kulino.ninehub.com/file.php/1/nhrestore/1/Jurnal_dan_Artikel_Ilmiah/Image_Processing/steganalysis.pdf
Steganalysis: The Investigation of Hidden Information I...

http://isis.poly.edu/~steganography/pubs/spie03.pdf
Steganography Capacity: A **Steganalysis** Perspective

http://pages.csam.montclair.edu/~robila/SECURITY/F2005_P/P5/Steganography_pap.pdf
Steganography -**Steganalysis**

http://www.krenn.nl/univ/cry/steg/article.pdf
Steganography and **Steganalysis**

http://www.eurasip.org/Proceedings/Eusipco/Eusipco2005/defevent/papers/cr1887.pdf
SPEECH STEGANALYSIS **USING CHAOTIC-TYPE FEATURES**

http://www.securityknox.com/Steg_project.pdf

Steganalysis: An Study of an Internet Search Engine and

http://www.nlpr.ia.ac.cn/2008papers/gjhy/gh87.pdf

BLIND IMAGE **STEGANALYSIS** BASED ON RUN-LENGTH HISTOGRAM ANALYSIS

http://eprints.kfupm.edu.sa/66887/1/66887.pdf

Steganalysis: The Investigation of Hidden Information

http://www.ece.rochester.edu/~gsharma/papers/OrsdemirSteganoAwareStegoEI2008_6819_41.pdf
Steganalysis-aware steganography: statistical indistinguishability

http://www.ece.stevens-tech.edu/~mouli/activesteg.pdf
Active **Steganalysis** of Sequential Steganography

http://grothoff.org/christian/stego.pdf
Translation-Based **Steganography**

http://www.infosecwriters.com/text_resources/pdf
Steganography_AMangarae.pdf
Steganography FAQ
-
http://www.eurojournals.com/ejsr_31_2_01.pdf
Attack on **LSB Steganography** in Color and Grayscale Images Using

http://www.mediateam.oulu.fi/publications/pdf/618.pdf
Increasing Robustness of **LSB** Audio **Steganography** by Reduced

http://www.mva-org.jp/Proceedings/2009CD/papers/13-14.pdf
An Optical **Video** Cryptosystem with Adaptive **Steganography** 13-14

An Optical Video Cryptosystem with Adaptive Steganography 13-14
LSB Technique for Secure Data Communication

http://www.scipub.org/fulltext/jcs/jcs5133-38.pdf
Image **Steganography** by Mapping Pixels to Letters

http://www.collegeduc.com/Journal/comp4eng.pdf
Hiding Data Using **LSB**-3

http://www.waset.org/journals/waset/v46/v46-16.pdf
Novel Security Strategy for Real Time Digital **Videos**

http://www.aicit.org/jdcta/ppl/jdcta_vol2no2_6.pdf
JDCTA: Call for Papers

http://debii.curtin.edu.au/~vidy/publications/INC_2004_Visibly%20Invisible%20Ciphertext%20as%20a%20Steganographic%20Carrier.pdf
Visibly Invisible: Ciphertext as a **Steganographic** Carrier