ΤΕΙ ΚΡΗΤΗΣ – ΠΑΡΑΡΤΗΜΑ ΧΑΝΙΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

Μελέτη του αλγορίθμου κρυπτογράφησης Advanced Encryption Standard (AES) και υλοποίησή του μέσω λογισμικού.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

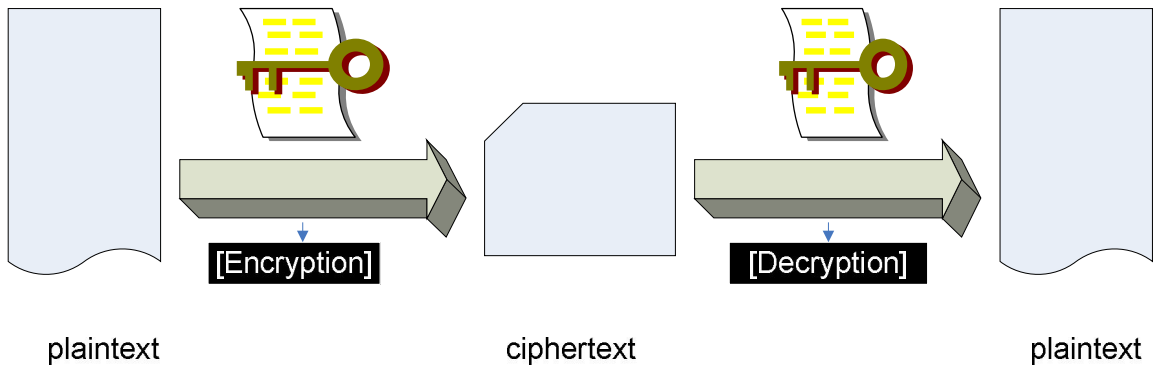
Νικολάου Δανδουλάκη

Επιβλέπων : Δρ. Μηχ. Νικόλαος Στ. Πετράκης
Καθηγητής Εφαρμογών

Χανιά 2011

1. Εισαγωγή.

Ο σκοπός αυτής της πτυχιακής εργασίας είναι η εισαγωγή του αναγνώστη στο θέμα της κρυπτογραφίας. Απευθύνεται στον καθένα ασχέτως από την ενημέρωση που έχει πάνω στο αντικείμενο αυτό. Η παρουσίαση του θέματος θα γίνεται σταδιακά έτσι ώστε να κατανοηθεί πλήρως όλη η δομή του. Θα μελετήσουμε την κρυπτογραφία σε βάθος καθώς θα ξεδιπλωθούν βασικές έννοιες, είδη κρυπτογραφίας επίσης γίνεται και μια σύντομη αναδρομή στο παρελθόν.
Έτσι στο δεύτερο κεφάλαιο που γίνεται η ιστορική ανάδρομη θα δούμε την εξέλιξη της κρυπτογραφίας χωρισμένη σε τρεις περιόδους: (α) Πρώτη περίοδος (1900 π.Χ. – 1900 μ.Χ.), (β) Δεύτερη περίοδος (1900 μ.Χ. – 1950 μΧ.), (γ) Τρίτη περίοδος (1950 μ.Χ. - Σήμερα).
Στη συνέχεια θα αναλυθούν τα είδη κρυπτοσυστημάτων όπου χωρίζονται σε Κλασσικά και σε Μοντέρνα. Θα υλοποιήσουμε ένα παράδειγμα κρυπτογραφίας και εν συνεχεία θα μελετήσουμε τις εφαρμογές της. Τέλος θα δούμε τρόπους και μεθόδους κρυπτογράφησης.
Στο τρίτο κεφάλαιο θα αναφερθούμε πλήρως στην ανάλυση του προτύπου AES. Έτσι για αρχή θα κοιτάξουμε τις εισόδους, εξόδους και την εσωτερική κατάσταση του αλγορίθμου κατά τη διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης. Έπειτα θα αναλύσουμε σε ψευδοκώδικα την λειτουργία της κρυπτογραφίας καθώς και τις συναρτήσεις που την απαρτίζουν. Έτσι θα ακούσουμε έννοιες όπως SubBytes, ShiftRows, MixColumns, AddRoundKey οι οποίες αναλύονται σταδιακά και λεπτομερώς με χρήση μαθηματικής φόρμουλας.
Την ίδια ακριβώς μελέτη κάνουμε και στην διαδικασία της αποκρυπτογράφησης του προτύπου, όπου επεξηγούμε τον ψευδοκώδικα που την απαρτίζει καθώς και τις αντίστοιχες συναρτήσεις οι οποίες είναι “αντίστροφες” με αυτές της κρυπτογράφησης, γι αυτό βάζουμε και την αρχική κατάληξη “inv” από το inverter (InvSubBytes, InvShiftRows, InvMixColumns).
Εν συνεχεία θα ενημερωθούμε για την επέκταση του κλειδιού, καθώς και εδώ αναλύουμε ψευδοκώδικα, βλέποντας έτσι την διαδικασία ανακατέματος του κλειδιού για την πλήρη ασφάλεια και μυστικότητα. Τέλος, θα πραγματοποιηθεί παράδειγμα λειτουργίας του αλγορίθμου δουλεύοντας στα 128bit. Στο παρακάτω σχήμα βλέπουμε την γενική λειτουργία του προτύπου.



Σχήμα 1.1: Πρότυπο AES.

Μπαίνοντας στο κεφάλαιο τέσσερα αναλύουμε τον αλγόριθμο AES σε γλώσσα προγραμματισμού όπου στην προκειμένη περίπτωση χρησιμοποιήσαμε τη ANSI-C. Θα γίνει ιστορική αναδρομή της γλώσσας καθώς και μια σύντομη αναφορά στις ιδιότητές της. Στη συνέχεια θα περιγράψουμε τον κώδικα για κάθε μια συνάρτηση και έπειτα θα αναλύσουμε διεξοδικά την διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης. Επίσης θα αναφερθούμε στις δυσκολίες που αντιμετωπίσαμε στην υλοποίηση του κώδικα και για το πως τελικά τις επιλύσαμε.

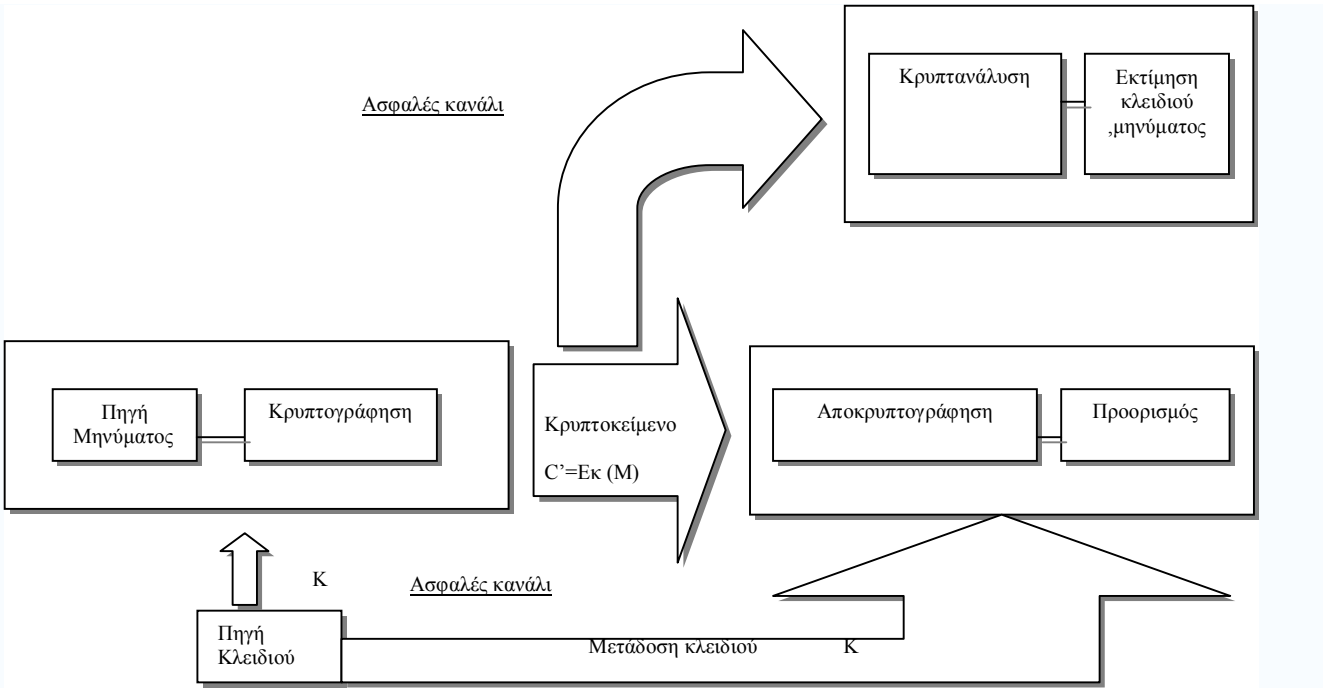
Τέλος, θα αναφερθούμε στις εισόδους, εξόδους και τις παραμέτρους που δημιουργήσαμε κατά τη διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης του αλγορίθμου (ανάλυση κώδικα).
Στο πέμπτο κεφάλαιο θα αναφερθούμε στη δημιουργία του γραφικού περιβάλλοντος του αλγορίθμου καθώς θα μελετήσουμε τη γλώσσα προγραμματισμού της visual basic 6. Θα γίνει μια αναφορά στα πλεονεκτήματα, γενικά στα χαρακτηριστικά της και για το ποιους λόγους επιλέξαμε τη γλώσσα αυτή. Θα γίνει πλήρης ανάλυση του γραφικού περιβάλλοντος με τη βοήθεια του οδηγού χρήσης, ο οποίος μας εξυπηρετεί στην επαρκή κατανόηση του προγράμματος.
Τέλος, στο κεφάλαιο έξι θα γίνει πλήρη αναφορά στα συμπεράσματα που δημιουργήθηκαν κατά τη διάρκεια κατανόησης και δημιουργίας του προτύπου AES καθώς και τα αποτελέσματα της λειτουργίας του, ως προς την ανταπόκριση του.

2. Κρυπτογραφία: Ιστορική Αναδρομή - Θεμελιώδεις Αρχές.

2.1. Γενικά.

Κρυπτογραφία είναι ο επιστημονικός κλάδος που πραγματεύεται τη μελέτη και σχεδίαση κρυπτογραφικών τεχνικών, συστημάτων και πρωτοκόλλων. Μαζί με τον κλάδο της Κρυπτανάλυσης, που ασχολείται με τη μελέτη τρόπων παραβίασης αυτών, απαρτίζουν την Επιστήμη της Κρυπτολογίας. Έτσι, Κρυπτολογία είναι η επιστήμη της απόκρυψης, από τη μια πλευρά και, από την άλλη, της αποκάλυψης του περιεχομένου κωδικοποιημένων μηνυμάτων ή δεδομένων.
Η επιθυμία προστασίας του περιεχομένου μηνυμάτων οδήγησε στην επινόηση και χρήση κρυπτογραφικών τεχνικών και συστημάτων τα οποία επιτρέπουν το μετασχηματισμό μηνυμάτων ή δεδομένων κατά τέτοιον τρόπο ώστε να είναι αδύνατη η υποκλοπή του περιεχομένου τους κατά τη μετάδοσή ή αποθήκευσή τους και, βεβαίως, την αντιστροφή του μετασχηματισμού. Η διαδικασία μετασχηματισμού καλείται κρυπτογράφηση και η αντίστροφή της αποκρυπτογράφηση.
Η συνάρτηση ή το σύνολο των κανόνων, στοιχείων και βημάτων που καθορίζουν την κρυπτογράφηση και την αποκρυπτογράφηση ονομάζεται κρυπτογραφικός αλγόριθμος. Η υλοποίηση του κρυπτογραφικού αλγόριθμου καλείται κρυπτογραφικό σύστημα. Μερικές φορές, ο κρυπτογραφικός αλγόριθμος καλείται και κωδικοποιητής (cipher). Πρωτόκολλα που χρησιμοποιούν κρυπτογραφικούς αλγόριθμους καλούνται κρυπτογραφικά πρωτόκολλα. Επειδή η αποθήκευση μπορεί να θεωρηθεί ως μετάδοση στη διάσταση του χρόνου, εφεξής θα μιλάμε για μετάδοση εννοώντας μετάδοση ή αποθήκευση.

Παράδειγμα:
Ο αντικειμενικός στόχος της κρυπτογραφίας είναι να δώσει την δυνατότητα σε 2 πρόσωπα, έστω τον Κώστα και την Βασιλική, να επικοινωνήσουν μέσα από ένα μη ασφαλές κανάλι με τέτοιο τρόπο ώστε ένα τρίτο πρόσωπο, μη εξουσιοδοτημένο (ένας αντίπαλος), να μην μπορεί να παρεμβληθεί στην επικοινωνία ή να κατανοήσει το περιεχόμενο των μηνυμάτων.



Σχήμα 2.1: Παράδειγμα κρυπτοσυστήματος.

Ένα κρυπτοσύστημα (σύνολο διαδικασιών κρυπτογράφησης - αποκρυπτογράφησης) αποτελείται από μία πεντάδα (P,C,k,E,D):

- Το P είναι ο χώρος όλων των δυνατών μηνυμάτων ή αλλιώς ανοικτών κειμένων
- Το C είναι ο χώρος όλων των δυνατών κρυπτογραφημένων μηνυμάτων ή αλλιώς κρυπτοκειμένων
- Το k είναι ο χώρος όλων των δυνατών κλειδιών ή αλλιώς κλειδοχώρος
- Η E είναι ο κρυπτογραφικός μετασχηματισμός ή κρυπτογραφική συνάρτηση
- Η D είναι η αντίστροφη συνάρτηση ή μετασχηματισμός αποκρυπτογράφησης

Η συνάρτηση κρυπτογράφησης E δέχεται δύο παραμέτρους, μέσα από τον χώρο P και τον χώρο k και παράγει μία ακολουθία που ανήκει στον χώρο C. Η συνάρτηση αποκρυπτογράφησης D δέχεται 2 παραμέτρους, τον χώρο C και τον χώρο k και παράγει μια ακολουθία που ανήκει στον χώρο P.
Το Σύστημα του Σχήματος λειτουργεί με τον ακόλουθο τρόπο :

- Ο αποστολέας επιλέγει ένα κλειδί μήκους n από τον χώρο κλειδιών με τυχαίο τρόπο, όπου τα n στοιχεία του K είναι στοιχεία από ένα πεπερασμένο αλφάβητο.
- Αποστέλλει το κλειδί στον παραλήπτη μέσα από ένα ασφαλές κανάλι.
- Ο αποστολέας δημιουργεί ένα μήνυμα από τον χώρο μηνυμάτων.

Η συνάρτηση κρυπτογράφησης παίρνει τις δυο εισόδους (κλειδί και μήνυμα) και παράγει μια κρυπτοακολουθία συμβόλων (έναν γρίφο) και η ακολουθία αυτή αποστέλλεται διαμέσου ενός μη ασφαλούς καναλιού.
Η συνάρτηση αποκρυπτογράφησης παίρνει ως όρισμα τις 2 τιμές (κλειδί και γρίφο) και παράγει την ισοδύναμη ακολουθία μηνύματος.
Ο αντίπαλος παρακολουθεί την επικοινωνία, ενημερώνεται για την κρυπτοακολουθία αλλά δεν έχει γνώση για την κλείδα που χρησιμοποιήθηκε και δεν μπορεί να αναδημιουργήσει το μήνυμα. Αν ο αντίπαλος επιλέξει να παρακολουθεί όλα τα μηνύματα θα προσανατολιστεί στην εξεύρεση του κλειδιού. Αν ο αντίπαλος ενδιαφέρεται μόνο για το υπάρχον μήνυμα θα παράγει μια εκτίμηση για την πληροφορία του μηνύματος.

2.2. Ιστορική Αναδρομή.

2.2.1. Πρώτη Περίοδος Κρυπτογραφίας (1900 π.Χ. – 1900 μ.Χ.).

Κατά την διάρκεια αυτής της περιόδου αναπτύχθηκε μεγάλο πλήθος μεθόδων και αλγορίθμων κρυπτογράφησης, που βασίζονταν κυρίως σε απλές αντικαταστάσεις γραμμάτων. Όλες αυτές δεν απαιτούσαν εξειδικευμένες γνώσεις και πολύπλοκες συσκευές, αλλά στηρίζονταν στην ευφυΐα και την ευρηματικότητα των δημιουργών τους. Όλα αυτά τα συστήματα έχουν στις μέρες μας κρυπταναλυθεί και έχει αποδειχθεί ότι, εάν είναι γνωστό ένα μεγάλο κομμάτι του κρυπτογραφημένου μηνύματος, τότε το αρχικό κείμενο μπορεί σχετικά εύκολα να επανακτηθεί.
Όπως προκύπτει από μία μικρή σφηνοειδή επιγραφή, που ανακαλύφθηκε στις όχθες του ποταμού Τίγρη, οι πολιτισμοί που αναπτύχθηκαν στην Μεσοποταμία ασχολήθηκαν με την κρυπτογραφία ήδη από το 1500 π.Χ. Η επιγραφή αυτή περιγράφει μία μέθοδο κατασκευής σμάλτων για αγγειοπλαστική και θεωρείται ως το αρχαιότερο κρυπτογραφημένο κείμενο (με βάση τον Kahn). Επίσης, ως το αρχαιότερο βιβλίο κρυπτοκωδικών στον κόσμο, θεωρείται μία σφηνοειδής επιγραφή στα Σούσα της Περσίας. η οποία περιλαμβάνει τους αριθμούς 1 έως 8 και από το 32 έως το 35, τοποθετημένους τον ένα κάτω από τον άλλο, ενώ απέναντι τους βρίσκονται τα αντίστοιχα για τον καθένα σφηνοειδή σύμβολα.
Η πρώτη στρατιωτική χρήση της κρυπτογραφίας αποδίδεται στους Σπαρτιάτες. Γύρω στον 5ο π.Χ. αιώνα εφηύραν την «σκυτάλη», την πρώτη κρυπτογραφική συσκευή, στην οποία χρησιμοποίησαν για την κρυπτογράφηση την μέθοδο της αντικατάστασης. Όπως αναφέρει ο Πλούταρχος, η «Σπαρτιατική Σκυτάλη» Σχήμα (2.2), ήταν μια ξύλινη ράβδος, ορισμένης διαμέτρου, γύρω από την οποία ήταν τυλιγμένη ελικοειδώς μια λωρίδα περγαμηνής. Το κείμενο ήταν γραμμένο σε στήλες, ένα γράμμα σε κάθε έλικα, όταν δε ξετύλιγαν τη λωρίδα, το κείμενο ήταν ακατάληπτο εξαιτίας της ανάμειξης των γραμμάτων. Το «κλειδί» ήταν η διάμετρος της σκυτάλης. Στην αρχαιότητα χρησιμοποιήθηκαν κυρίως

συστήματα, τα οποία βασίζονταν στην στεγανογραφία και όχι τόσο στην κρυπτογραφία. Οι Έλληνες συγγραφείς δεν αναφέρουν αν και πότε χρησιμοποιήθηκαν συστήματα γραπτής αντικατάστασης γραμμάτων, αλλά τα βρίσκουμε στους Ρωμαίους, κυρίως την εποχή του Ιουλίου Καίσαρα. Ο Ιούλιος Καίσαρας έγραφε στον Κικέρωνα και σε άλλους φίλους του, αντικαθιστώντας τα γράμματα του κειμένου, με γράμματα, που βρίσκονται 3 θέσεις μετά, στο Λατινικό Αλφάβητο. Έτσι, σήμερα, το σύστημα κρυπτογράφησης που στηρίζεται στην αντικατάσταση των γραμμάτων του αλφαβήτου με άλλα που βρίσκονται σε καθορισμένο αριθμό θέσης πριν ή μετά, λέγεται κρυπτοσύστημα αντικατάστασης του Καίσαρα. Ο Καίσαρας χρησιμοποίησε και άλλα, πιο πολύπλοκα συστήματα κρυπτογράφησης, για τα οποία έγραψε ένα βιβλίο ο Valerius Probus, το οποίο δυστυχώς δεν διασώθηκε, αλλά αν και χαμένο, θεωρείται το πρώτο βιβλίο κρυπτολογίας. Το σύστημα αντικατάστασης του Καίσαρα, χρησιμοποιήθηκε ευρύτατα και στους επόμενους αιώνες.
Στην διάρκεια του Μεσαίωνα, η κρυπτολογία ήταν κάτι το απαγορευμένο και αποτελούσε μια μορφή αποκρυφισμού και μαύρης μαγείας, κάτι που συντέλεσε στην καθυστέρηση της ανάπτυξης της. Η εξέλιξη, τόσο της κρυπτολογίας, όπως και των μαθηματικών, συνεχίζεται στον Αραβικό κόσμο. Στο γνωστό μυθιστόρημα «Χίλιες και μία νύχτες» κυριαρχούν οι λέξεις-αινίγματα, οι γρίφοι, τα λογοπαίγνια και οι αναγραμματισμοί. Έτσι, εμφανίστηκαν βιβλία που περιείχαν κρυπταλφάβητα, όπως το αλφάβητο «Dawoudi» που πήρε το όνομα του από τον βασιλιά Δαυίδ. Οι Άραβες είναι οι πρώτοι που επινόησαν αλλά και χρησιμοποίησαν μεθόδους κρυπτανάλυσης. Το κυριότερο εργαλείο στην κρυπτανάλυση, η χρησιμοποίηση των συχνοτήτων των γραμμάτων κειμένου, σε συνδυασμό με τις συχνότητες εμφάνισης στα κείμενα των γραμμάτων της γλώσσας, επινοήθηκε από αυτούς γύρω στον 14ο αιώνα. Η κρυπτογραφία, λόγω των στρατιωτικών εξελίξεων, σημείωσε σημαντική ανάπτυξη στους επόμενους αιώνες
. Ο Ιταλός Giovanni Batista Porta, το 1563, δημοσίευσε το περίφημο για την κρυπτολογία βιβλίο «De furtivis literarum notis», με το οποίο έγιναν γνωστά τα πολυαλφαβητικά συστήματα κρυπτογράφησης και τα διγραφικά κρυπτογραφήματα, στα οποία, δύο γράμματα αντικαθίστανται από ένα. Σημαντικός εκπρόσωπος εκείνης της εποχής είναι και ο Γάλλος Vigenere, του οποίου ο πίνακας πολυαλφαβητικής αντικατάστασης, χρησιμοποιείται ακόμη και σήμερα.

Σχήμα 2.2: Η Σπαρτιατική Σκυτάλη, μια πρώιμη συσκευή για την κρυπτογράφηση.

Ο C.Wheatstone, γνωστός από τις μελέτες του στον ηλεκτρισμό, παρουσίασε την πρώτη μηχανική κρυπτοσυσκευή, η οποία απετέλεσε τη βάση για την ανάπτυξη των κρυπτομηχανών της δεύτερης ιστορικής περιόδου της κρυπτογραφίας. Η μεγαλύτερη αποκρυπτογράφηση ήταν αυτή των αιγυπτιακών ιερογλυφικών τα οποία, επί αιώνες, παρέμεναν μυστήριο και οι αρχαιολόγοι μόνο εικασίες μπορούσαν να διατυπώσουν για τη σημασία τους. Ωστόσο, χάρη σε μία κρυπταναλυτική εργασία, τα ιερογλυφικά εν τέλει αναλύθηκαν και έκτοτε οι αρχαιολόγοι είναι σε θέση να διαβάζουν ιστορικές επιγραφές. Τα αρχαιότερα ιερογλυφικά χρονολογούνται περίπου από το 3000 π.Χ. Τα σύμβολα των ιερογλυφικών ήταν υπερβολικά πολύπλοκα για την καταγραφή των συναλλαγών εκείνης της εποχής. Έτσι, παράλληλα με αυτά, αναπτύχθηκε για καθημερινή χρήση η ιερατική γραφή, που ήταν μία συλλογή συμβόλων, τα οποία ήταν εύκολα τόσο στο γράψιμο όσο και στην ανάγνωση. Τον 17ο αιώνα αναθερμάνθηκε το ενδιαφέρον για την αποκρυπτογράφηση των ιερογλυφικών, έτσι το

1652 ο Γερμανός Ιησουΐτης Αθανάσιος Κίρχερ εξέδωσε ένα λεξικό ερμηνείας τους, με τίτλο «Oedipus Aegyptiacus». Με βάση αυτό προσπάθησε να ερμηνεύσει τις αιγυπτιακές γραφές, αλλά η προσπάθεια του αυτή ήταν κατά γενική ομολογία αποτυχημένη. Για παράδειγμα, το όνομα του Φαραώ Απρίη, το ερμήνευσε σαν «τα ευεργετήματα του θεϊκού Όσιρι εξασφαλίζονται μέσω των ιερών τελετών της αλυσίδας των πνευμάτων, ώστε να επιδαψιλεύσουν τα δώρα του Νείλου». Παρόλα αυτά, η προσπάθεια του άνοιξε τον δρόμο προς την σωστή ερμηνεία των ιερογλυφικών, που προχώρησε χάρη στην ανακάλυψη της «Στήλης της Ροζέτας». Ήταν μια πέτρινη στήλη που βρήκαν τα στρατεύματα του Ναπολέοντα στην Αίγυπτο και είχε χαραγμένο πάνω της το ίδιο κείμενο τρεις φορές. Μια με ιερογλυφικά, μια στα ελληνικά και μια ιερατική γραφή. Δύο μεγάλοι αποκρυπτογράφοι της εποχής, ο Γιάνγκ και ο Σαμπολιόν, μοιράστηκαν την δόξα της ερμηνείας τους. Οι προϊστορικοί πληθυσμοί χρησιμοποίησαν τρεις γραφές μέχρι να επινοήσουν αλφάβητο, γύρω στο 850 π.Χ .

Χρονολογικά, οι γραφές αυτές κατατάσσονται ως εξής:

- 3000 1600 π.Χ . : Εικονογραφική (Ιερογλυφική) γραφή
- 1850 1450 π.Χ .: Γραμμική γραφή Α
- 1450 1200 π.Χ .: Γραμμική Γραφή Β

Η Κρητική εικονογραφική (ή ιερογλυφική) γραφή δεν μας έχει αποκαλύψει τον κώδικα της, γνωρίζουμε ωστόσο ότι δεν πρόκειται για γραφή που χρησιμοποιεί εικόνες ως σημεία, αλλά για φωνητική γραφή, η οποία εξαντλείται σε περίπου διακόσιους σφραγιδόλιθους και συνυπήρχε με την γραμμική γραφή Α, τόσο χρονικά όσο και τοπικά, όπως προκύπτει από τις ανασκαφές στο ανάκτορο των Μαλίων της Κρήτης. Εμφανίζεται στον Δίσκο της Φαιστού (Σχήμα 2.3), που ανακαλύφθηκε το 1908 στην νότια Κρήτη. Πρόκειται για μια κυκλική πινακίδα, που χρονολογείται γύρω στο 1700 π.Χ . και φέρει γραφή με την μορφή δύο σπειρών. Τα σύμβολα δεν είναι χειροποίητα, αλλά έχουν χαραχθεί με την βοήθεια μίας ποικιλίας σφραγίδων, καθιστώντας τον Δίσκο ως το αρχαιότερο δείγμα στοιχειοθεσίας. Δεν υπάρχει άλλο ανάλογο εύρημα και έτσι η αποκρυπτογράφηση στηρίζεται σε πολύ περιορισμένες πληροφορίες. Μέχρι σήμερα δεν έχει αποκρυπτογραφηθεί και παραμένει η πιο μυστηριώδης αρχαία ευρωπαϊκή γραφή.

Σχήμα 2.3: Ο Δίσκος της Φαιστού.

Οι πρώτες επιγραφές με Γραμμική γραφή ανακαλύφθηκαν από τον Άρθουρ Έβανς (Sir Arthur Evans), τον μεγάλο Άγγλο αρχαιολόγο, που ανάσκαψε συστηματικά την Κνωσό το 1900. Ο ίδιος ονόμασε αυτή τη γραφή γραμμική, επειδή τα γράμματα της είναι γραμμές (ένα γραμμικό σχήμα) και όχι σφήνες, όπως στην σφηνοειδή γραφή ή εικόνες όντων, όπως στην αιγυπτιακή ιερατική. Η γραμμική γραφή Α είναι μάλλον η γραφή των Μινωιτών (από το μυθικό Μίνωα, βασιλιά της Κνωσού), των κατοίκων της αρχαίας Κρήτης και από αυτή ίσως να προήλθε το σημερινό ελληνικό αλφάβητο. Τα γράμματα της γραμμικής γραφής χαράζονταν με αιχμηρό αντικείμενο πάνω σε πήλινες πλάκες, οι

οποίες κατόπιν ξεραίνονταν σε φούρνους. Οι περισσότερες από τις επιγραφές με Γραμμική γραφή Α (περίπου 1500) είναι λογιστικές και περιέχουν εικόνες ή συντομογραφίες των εμπορεύσιμων προϊόντων και αριθμούς για υπόδειξη της ποσότητας ή οφειλής.
Ο Έβανς κατέγραψε 135 σύμβολα της. Χρησιμοποιήθηκε κυρίως στην Κρήτη, αν και ορισμένα πρόσφατα ευρήματα καταδεικνύουν ότι μπορεί να αποτέλεσε μέσο γραφής και αλλού, αφού επιγραφές με Γραμμική Α έχουν βρεθεί στην Κνωσό και Φαιστό της Κρήτης, αλλά και στη Μήλο και τη Θήρα. Πλάκες με επιγραφές σε γραμμική Α, εκτίθενται στο Μουσείο Ηρακλείου. Παρά την πρόοδο που έχει σημειωθεί, η γραμμική γραφή Α δεν έχει αποκρυπτογραφηθεί ακόμη. Ο Evans έδωσε και την ονομασία στην Γραμμική Γραφή Β, επειδή αναγνώρισε ότι πρόκειται για συγγενική γραφή με την γραμμική Α, πιο πρόσφατη ωστόσο και εξελιγμένη. Με βάση όσα γνωρίζουμε σήμερα, η γραφή αυτή υιοθετήθηκε αποκλειστικά για λογιστικούς σκοπούς. Πινακίδες χαραγμένες με την γραμμική γραφή Β βρέθηκαν στην Κνωσό, στα Χανιά αλλά και στην Πύλο, τις Μυκήνες, τη Θήβα και την Τίρυνθα. Σήμερα αποτελούν ένα σύνολο 10.000 τεμαχίων. Τα σχήματα των πινακίδων της γραφής αυτής ποικίλουν, επικρατούν όμως οι φυλλοειδείς και «σελιδόσχημες», οι οποίες διαφέρουν ως προς τις διαστάσεις, ανάλογα με τις προτιμήσεις του κάθε γραφέα. Έπλαθαν πηλό σε σχήμα κυλίνδρου, τον τοποθετούσαν σε λεία επιφάνεια και την πίεζαν μέχρι να γίνει επίπεδη, επιμήκης και συμπαγής πινακίδα, σαφώς διαφοροποιημένη σε δύο επιφάνειες: μία επίπεδη λειασμένη, που επρόκειτο να αποτελέσει την κύρια γραφική επιφάνεια και μία κυρτή, που συνήθως έμενε άγραφη. Πολλές φορές, όταν τα κείμενα απαιτούσαν περισσότερες από μία πινακίδες, έχουμε τις αποκαλούμενες «ομάδες» ή «πολύπτυχα» πινακίδων, οι οποίες εμφανίζουν κοινά χαρακτηριστικά και ως προς την αποξήρανση και το μίγμα του πηλού και κυρίως, ως προς το γραφικό χαρακτήρα του ίδιου του γραφέα. Τα πολύπτυχα αυτά φυλάσσονταν σε αρχειοφυλάκια και ταξινομούνταν κατά θέματα σε ξύλινα κιβώτια. Για να γνωρίζει ο ενδιαφερόμενος το περιεχόμενο των καλαθιών, κυρίως, χρησιμοποιούσαν ετικέτες: ένα σφαιρίδιο πηλού, εντυπωμένο στην πρόσθια πλευρά, στο οποίο καταγράφονταν συνοπτικές πληροφορίες. Συστηματικά, με την γραφή αυτή, με την οποία είχε πραγματικό πάθος, ασχολήθηκε ο Άγγλος αρχιτέκτονας και ερασιτέχνης αρχαιολόγος Μ. Βέντρις. Ήταν ο πρώτος που κατάλαβε ότι επρόκειτο για κάποιο είδος ελληνικής γραφής, αλλά η άποψη του αυτή δεν έγινε δεκτή αρχικά από τους ειδικούς. Στην συνέχεια, όμως, αρκετοί προσχώρησαν στην άποψή του. Ένας από αυτούς ήταν ο κρυπταναλυτής Τζον Τσάντγουικ, ο οποίος, στη διάρκεια του πολέμου, είχε εργασθεί στην ανάλυση της γερμανικής κρυπτομηχανής Enigma. Προσπάθησε να μεταφέρει την πείρα του στην κρυπτανάλυση της Γραμμικής Β, αλλά χωρίς επιτυχία μέχρι τότε. Όμως, ο συνδυασμός των δύο επιστημόνων έφερε το πολυπόθητο αποτέλεσμα. Το 1953 κατέγραψαν τα συμπεράσματά τους στο μνημειώδες έργο «Μαρτυρίες για την ελληνική διάλεκτο στα μυκηναϊκά αρχεία», που έγινε το πιο διάσημο άρθρο κρυπτανάλυσης. Η αποκρυπτογράφηση της Γραμμικής Β απέδειξε ότι επρόκειτο για ελληνική γλώσσα, ότι οι Μινωίτες της Κρήτης μιλούσαν ελληνικά και ότι η δεσπόζουσα δύναμη εκείνη την εποχή ήταν οι Μυκήνες. Η αποκρυπτογράφηση της Γραμμικής Β θεωρήθηκε επίτευγμα ανάλογο της κατάκτησης του Έβερεστ, που συνέβη την ίδια ακριβώς εποχή. Για αυτό και έγινε γνωστή σαν το «Έβερεστ της Ελληνικής αρχαιολογίας».

2.2.2. Δεύτερη Περίοδος Κρυπτογραφίας (1900 μΧ. – 1950 μΧ.).

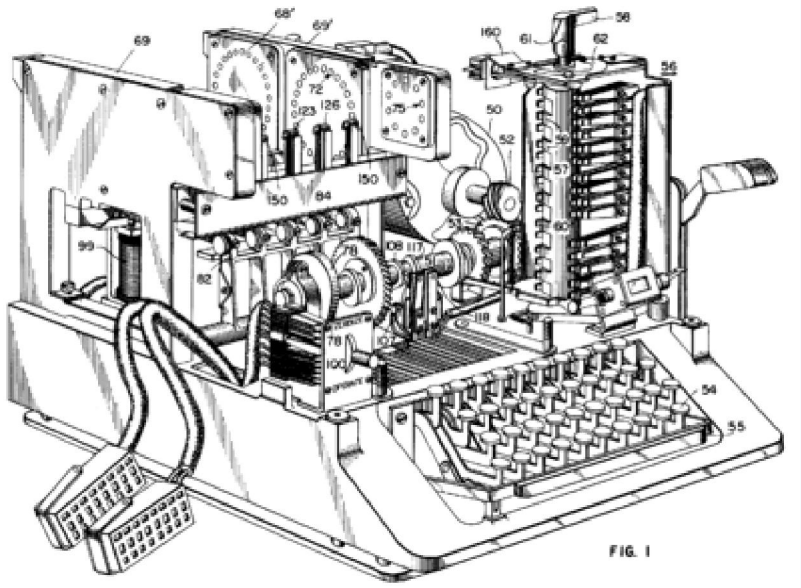
Η δεύτερη περίοδος της κρυπτογραφίας όπως προαναφέρθηκε τοποθετείται στις αρχές του 20ου αιώνα και φτάνει μέχρι το 1950. Καλύπτει, επομένως, τους δύο παγκόσμιους πολέμους, εξαιτίας των οποίων (λόγω της εξαιρετικά μεγάλης ανάγκης που υπήρξε για ασφάλεια κατά την μετάδοση ζωτικών πληροφοριών μεταξύ των στρατευμάτων των χωρών) αναπτύχθηκε η κρυπτογραφία τόσο όσο δεν είχε αναπτυχθεί τα προηγούμενα 3000 χρόνια. Τα κρυπτοσυστήματα αυτής της περιόδου αρχίζουν να γίνονται πολύπλοκα, και να αποτελούνται από μηχανικές και ηλεκτρομηχανικές κατασκευές, οι οποίες ονομάζονται «κρυπτομηχανές». Η κρυπτανάλυση τους, απαιτεί μεγάλο αριθμό προσωπικού, το οποίο εργαζόταν επί μεγάλο χρονικό διάστημα ενώ ταυτόχρονα γίνεται εξαιρετικά αισθητή η ανάγκη για μεγάλη υπολογιστική ισχύ. Παρά την πολυπλοκότητα που αποκτούν τα συστήματα κρυπτογράφησης κατά την διάρκεια αυτής της περιόδου η κρυπτανάλυση τους είναι συνήθως επιτυχημένη. Οι Γερμανοί έκαναν εκτενή χρήση (σε διάφορες παραλλαγές) ενός συστήματος γνωστού ως Enigma (Σχήμα 2.4).

Σχήμα 2.4 : Η μηχανή Αίνιγμα χρησιμοποιήθηκε ευρέως από την Γερμανία.

Ο Marian Rejewski, στην Πολωνία, προσπάθησε και, τελικά, παραβίασε την πρώτη μορφή του γερμανικού στρατιωτικού συστήματος Enigma (που χρησιμοποιούσε μια ηλεκτρομηχανική κρυπτογραφική συσκευή) χρησιμοποιώντας θεωρητικά μαθηματικά το 1932. Ήταν η μεγαλύτερη σημαντική ανακάλυψη στην κρυπτολογική ανάλυση της εποχής. Οι Πολωνοί συνέχισαν να αποκρυπτογραφούν τα μηνύματα που βασίζονταν στην κρυπτογράφηση με το Enigma μέχρι το 1939. Τότε, ο γερμανικός στρατός έκανε ορισμένες σημαντικές αλλαγές και οι Πολωνοί δεν μπόρεσαν να τις παρακολουθήσουν, επειδή η αποκρυπτογράφηση απαιτούσε περισσότερους πόρους από όσους μπορούσαν να διαθέσουν. Έτσι, εκείνο το καλοκαίρι μεταβίβασαν τη γνώση τους, μαζί με μερικές μηχανές που είχαν κατασκευάσει, στους Βρετανούς και τους Γάλλους. Ακόμη και ο Rejewski και οι μαθηματικοί και κρυπτογράφοι του, όπως ο Biuro Szyfrow, κατέληξαν σε συνεργασία με τους Βρετανούς και τους Γάλλους μετά από αυτή την εξέλιξη. Η συνεργασία αυτή συνεχίστηκε από τον Άλαν Τούρινγκ (Alan Turing), τον Γκόρντον Ουέλτσμαν (Gordon Welchman) και από πολλούς άλλους στο Μπλέτσλεϊ Παρκ (Bletchley Park), κέντρο της Βρετανικής Υπηρεσίας από/κρυπτογράφησης και οδήγησε σε συνεχείς αποκρυπτογραφήσεις των διαφόρων παραλλαγών του Enigma, με την βοήθεια και ενός υπολογιστή, που κατασκεύασαν οι Βρετανοί επιστήμονες, ο οποίος ονομάσθηκε Colossus και, δυστυχώς, καταστράφηκε με το τέλος του Πολέμου. Οι κρυπτογράφοι του αμερικανικού ναυτικού (σε συνεργασία με Βρετανούς και Ολλανδούς κρυπτογράφους μετά από το 1940) έσπασαν αρκετά κρυπτοσυστήματα του Ιαπωνικού ναυτικού. Το σπάσιμο ενός από αυτά, του JN-25, οδήγησε στην αμερικανική νίκη στην Ναυμαχία της Μιντγουέι καθώς και στην εξόντωση του Αρχηγού του Ιαπωνικού Στόλου Ιζορόκου Γιαμαμότο.

Το Ιαπωνικό Υπουργείο Εξωτερικών χρησιμοποίησε ένα τοπικά αναπτυγμένο κρυπτογραφικό σύστημα, (που καλείται Purple), και χρησιμοποίησε, επίσης, διάφορες παρόμοιες μηχανές για τις συνδέσεις μερικών ιαπωνικών πρεσβειών. Μία από αυτές αποκλήθηκε "Μηχανή-Μ" από τις ΗΠΑ, ενώ μια άλλη αναφέρθηκε ως «Red» (Κόκκινη). Μια ομάδα του αμερικανικού στρατού, η αποκαλούμενη SIS, κατάφερε να σπάσει το ασφαλέστερο ιαπωνικό διπλωματικό σύστημα κρυπτογράφησης (μια ηλεκτρομηχανική συσκευή, η οποία αποκλήθηκε "Purple" από τους Αμερικανούς) πριν καν ακόμη αρχίσει ο Β΄ Παγκόσμιος Πόλεμος. Οι Αμερικανοί αναφέρονται στο

αποτέλεσμα της κρυπτανάλυσης, ειδικότερα της μηχανής Purple, αποκαλώντας το ως Magic (Μαγεία).
Οι συμμαχικές κρυπτομηχανές που χρησιμοποιήθηκαν στον δεύτερο παγκόσμιο πόλεμο περιλάμβαναν το βρετανικό TypeX και το αμερικανικό SIGABA (Σχήμα 2.5). Και τα δύο ήταν ηλεκτρομηχανικά σχέδια παρόμοια στο πνεύμα με το Enigma, με σημαντικές εν τούτοις βελτιώσεις. Κανένα δεν έγινε γνωστό ότι παραβιάστηκε κατά τη διάρκεια του πολέμου. Τα στρατεύματα στο πεδίο μάχης χρησιμοποίησαν το M-209 και τη λιγότερη ασφαλή οικογένεια κρυπτομηχανών M-94. Οι Βρετανοί πράκτορες της Υπηρεσίας "SOE" χρησιμοποίησαν αρχικά ένα τύπο κρυπτογραφίας που βασιζόταν σε ποιήματα (τα απομνημονευμένα ποιήματα ήταν τα κλειδιά). Οι Γερμανοί, ώρες πριν την Απόβαση της Νορμανδίας συνέλαβαν ένα μήνυμα - ποίημα του Πολ Βερλέν, για το οποίο, χωρίς να το έχουν αποκρυπτογραφήσει, ήταν βέβαιοι πως προανήγγειλε την απόβαση. Η Γερμανική ηγεσία δεν έλαβε υπόψη της αυτή την προειδοποίηση.[1]
Οι Πολωνοί είχαν προετοιμαστεί για την εμπόλεμη περίοδο κατασκευάζοντας την κρυπτομηχανή LCD Lacida, η οποία κρατήθηκε μυστική ακόμη και από τον Rejewski. Όταν. τον Ιούλιο του 1941 ελέγχθηκε από τον Rejewski η ασφάλειά της, του χρειάστηκαν μερικές μόνον ώρες για να την "σπάσει" και έτσι αναγκάστηκαν να την αλλάξουν βιαστικά. Τα μηνύματα που εστάλησαν με Lacida δεν ήταν, εντούτοις, συγκρίσιμα με αυτά του Enigma, αλλά η παρεμπόδιση θα μπορούσε να έχει σημάνει το τέλος της κρίσιμης κρυπταναλυτικής Πολωνικής προσπάθειας.



Σχήμα 2.5: Κρυπτό-μηχανή SIGABA.

2.2.3. Τρίτη Περίοδος Κρυπτογραφίας (1950 μ.Χ. - Σήμερα).

Αυτή η περίοδος χαρακτηρίζεται από την έξαρση της ανάπτυξης στους επιστημονικούς κλάδους των μαθηματικών, της μικροηλεκτρονικής και των υπολογιστικών συστημάτων. Η εποχή της σύγχρονης κρυπτογραφίας αρχίζει ουσιαστικά με τον Claude Shannon, αναμφισβήτητα ο πατέρας των μαθηματικών συστημάτων κρυπτογραφίας. Το 1949 δημοσίευσε το έγγραφο «Θεωρία επικοινωνίας των συστημάτων μυστικότητας» (Communication Theory of Secrecy Systems) στο τεχνικό περιοδικό Bell System και λίγο αργότερα στο βιβλίο του, «Μαθηματική Θεωρία της Επικοινωνίας» (Mathematical Theory of Communication), μαζί με τον Warren Weaver. Αυτά, εκτός από τις άλλες εργασίες του επάνω στην θεωρία δεδομένων και επικοινωνίας καθιέρωσε μια στερεά θεωρητική βάση για την κρυπτογραφία και την κρυπτανάλυση. Εκείνη την εποχή η κρυπτογραφία εξαφανίζεται και φυλάσσεται από τις μυστικές υπηρεσίες κυβερνητικών επικοινωνιών όπως η NSA. Πολύ λίγες εξελίξεις δημοσιοποιήθηκαν ξανά μέχρι τα μέσα της δεκαετίας του '70, όταν όλα άλλαξαν.

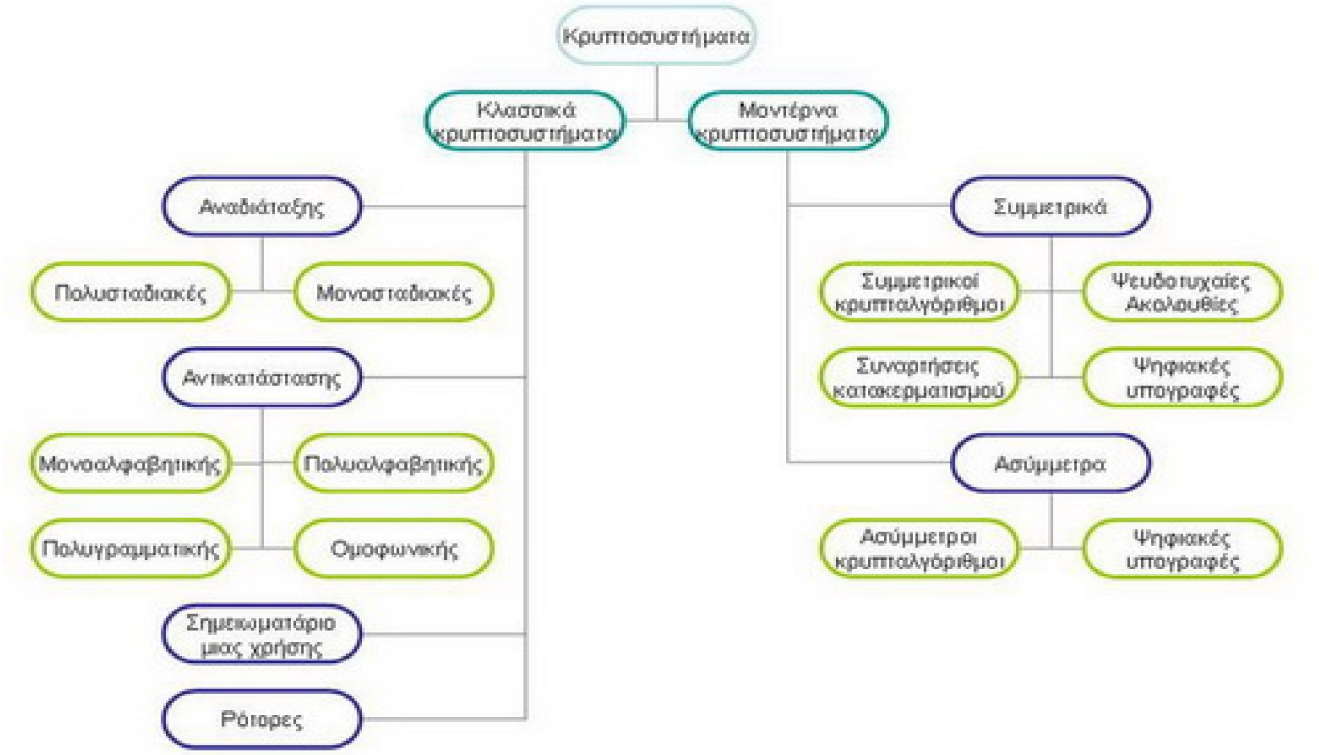
Στα μέσα της δεκαετίας του '70 έγιναν δύο σημαντικές δημόσιες (δηλ. μη-μυστικές) πρόοδοι. Πρώτα ήταν η δημοσίευση του σχεδίου προτύπου κρυπτογράφησης DES (Data Encryption Standard) στον ομοσπονδιακό κατάλογο της Αμερικής στις 17 Μαρτίου 1975. Το προτεινόμενο DES υποβλήθηκε από την IBM, στην πρόσκληση του Εθνικού Γραφείου των Προτύπων (τώρα γνωστό ως NIST), σε μια προσπάθεια να αναπτυχθούν ασφαλείς ηλεκτρονικές εγκαταστάσεις επικοινωνίας για επιχειρήσεις όπως τράπεζες και άλλες μεγάλες οικονομικές οργανώσεις. Μετά από τις συμβουλές και την τροποποίηση από την NSA, αυτό το πρότυπο υιοθετήθηκε και δημοσιεύθηκε ως ένα ομοσπονδιακή τυποποιημένο πρότυπο επεξεργασίας πληροφοριών το 1977 (αυτήν την περίοδο αναφέρεται σαν FIPS 46-3). Ο DES ήταν ο πρώτος δημόσια προσιτός αλγόριθμος κρυπτογράφησης που εγκρίνεται από μια εθνική αντιπροσωπεία όπως η NSA. Η απελευθέρωση της προδιαγραφής της από την NBS υποκίνησε μια έκρηξη δημόσιου και ακαδημαϊκού ενδιαφέροντος για τα συστήματα κρυπτογραφίας.
Ο DES αντικαταστάθηκε επίσημα από τον AES το 2001 όταν ανήγγειλε ο NIST το FIPS 197. Μετά από έναν ανοικτό διαγωνισμό, ο NIST επέλεξε τον αλγόριθμο Rijndael, που υποβλήθηκε από δύο Φλαμανδούς κρυπτογράφους, για να είναι το AES. Ο DES και οι ασφαλέστερες παραλλαγές του όπως ο 3DES ή TDES χρησιμοποιούνται ακόμα σήμερα, ενσωματωμένος σε πολλά εθνικά και οργανωτικά πρότυπα. Εντούτοις, το βασικό μέγεθος των 56-bit έχει αποδειχθεί ότι είναι ανεπαρκές να αντισταθεί στις επιθέσεις ωμής βίας (μια τέτοια επίθεση πέτυχε να σπάσει τον DES σε 56 ώρες ενώ το άρθρο που αναφέρεται ως το σπάσιμο του DES δημοσιεύτηκε από τον O'Reilly and Associates). Κατά συνέπεια, η χρήση απλής κρυπτογράφησης με τον DES είναι τώρα χωρίς την αμφιβολία επισφαλής για χρήση στα νέα σχέδια των κρυπτογραφικών συστημάτων και μηνύματα που προστατεύονται από τα παλαιότερα κρυπτογραφικά συστήματα που χρησιμοποιούν DES, και όλα τα μηνύματα που έχουν αποσταλεί από το 1976 με την χρήση DES, διατρέχουν επίσης σοβαρό κίνδυνο αποκρυπτογράφησης. Ανεξάρτητα από την έμφυτη ποιότητά του, το βασικό μέγεθος του DES (56-bit) ήταν πιθανά πάρα πολύ μικρό ακόμη και το 1976, πράγμα που είχε επισημάνει ο Whitfield Diffie. Υπήρξε επίσης η υποψία ότι κυβερνητικές οργανώσεις είχαν ακόμα και τότε ικανοποιητική υπολογιστική δύναμη ώστε να σπάσουν μηνύματα που είχαν κρυπτογραφηθεί με τον DES.

2.3. Είδη Κρυπτοσυστημάτων.

Τα κρυπτοσυστήματα χωρίζονται σε 2 μεγάλες κατηγορίες τα Κλασσικά Κρυπτοσυστήματα και τα Μοντέρνα κρυπτοσυστήματα.
Επιπροσθέτως, οι κρυπτογραφικοί αλγόριθμοι μπορούν να χωριστούν σε δύο διαφορετικές κατηγορίες με βάση τον τρόπο κρυπτογράφησης των μηνυμάτων:

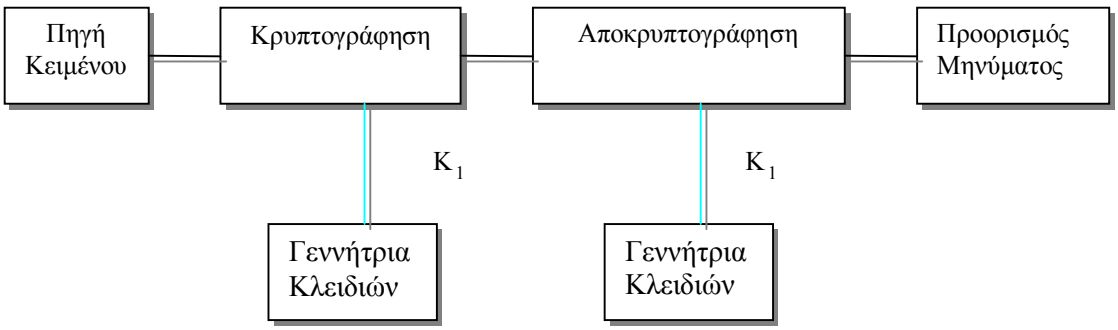
- Δέσμης (Block Ciphers), οι οποίοι χωρίζουν το μήνυμα σε κομμάτια και κρυπτογραφούν κάθε ένα από τα κομμάτια αυτά χωριστά.
- Ροής (Stream Ciphers), οι οποίοι κρυπτογραφούν μία ροή μηνύματος (stream) χωρίς να την διαχωρίζουν σε τμήματα.

-Κλασσικά Κρυπτοσυστήματα
-Μοντέρνα Κρυπτοσυστήματα
-Συμμετρικά Κρυπτοσυστήματα



Σχήμα 2.6: Μπλοκ ανάλυσης ειδών κρυπτοσυστήματος.

Συμμετρικό κρυπτοσύστημα είναι το σύστημα εκείνο το οποίο χρησιμοποιεί κατά την διαδικασία της κρυπτογράφησης αποκρυπτογράφησης ένα κοινό κλειδί (Σχ. 2.7). Η ασφάλεια αυτών των αλγορίθμων βασίζεται στην μυστικότητα του κλειδιού. Τα συμμετρικά κρυπτοσυστήματα προϋποθέτουν την ανταλλαγή του κλειδιού μέσα από ένα ασφαλές κανάλι επικοινωνίας ή μέσα από την φυσική παρουσία των προσώπων. Αυτό το χαρακτηριστικό καθιστά δύσκολη την επικοινωνία μεταξύ απομακρυσμένων ατόμων.



Σχήμα 2.7: Μοντέλο Συμμετρικού Κρυπτοσυστήματος.

Τα στάδια της επικοινωνίας του σχήματος 2.7 είναι τα ακόλουθα:

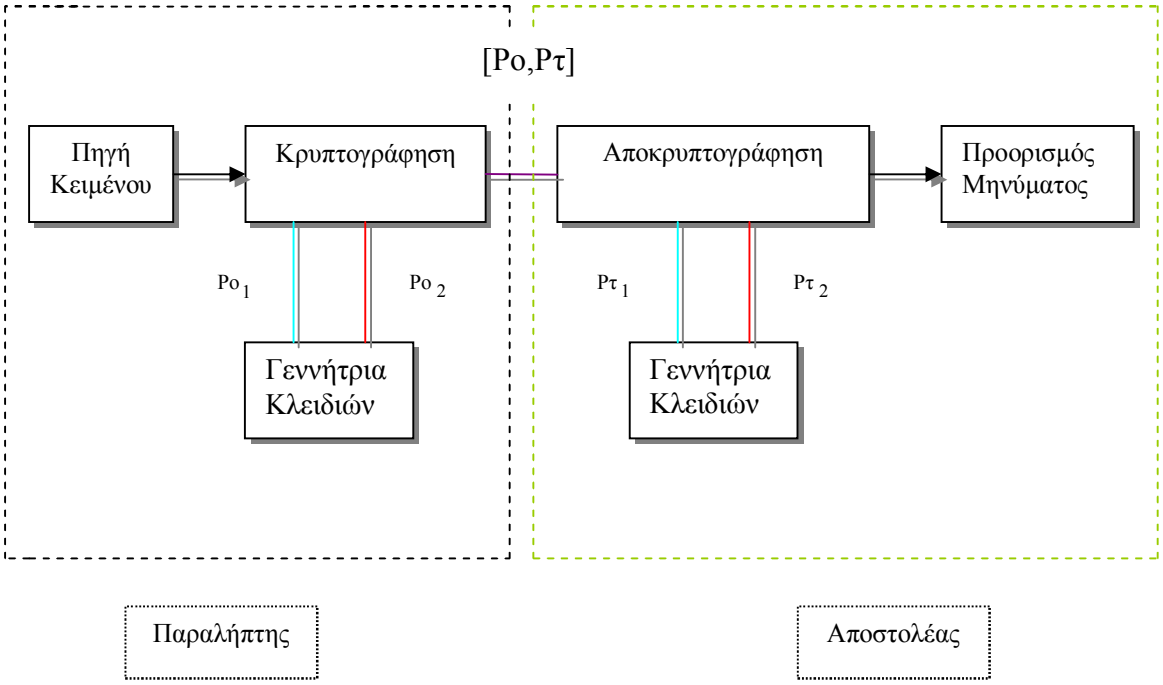
1. Ο Κώστας ή η Βασιλική αποφασίζει για ένα κλειδί το οποίο το επιλέγει τυχαία μέσα από τον κλειδοχώρο.
2. Η Βασιλική αποστέλλει το κλειδί στον Κώστα μέσα από ένα ασφαλές κανάλι.
3. Ο Κώστας δημιουργεί ένα μήνυμα όπου τα σύμβολα m ανήκουν στον χώρο των μηνυμάτων.
4. Κρυπτογραφεί το μήνυμα με το κλειδί που έλαβε από την Βασιλική και η παραγόμενη κρυπτοσυμβολοσειρά αποστέλλεται.
5. Η Βασιλική λαμβάνει την κρυπτοσυμβολοσειρά και στην συνέχεια με το ίδιο κλειδί την αποκρυπτογραφεί και η έξοδος που παράγεται είναι το μήνυμα.

Έχουμε το αρχικό μήνυμα, (ένα σύνολο δυαδικών ψηφίων (bits) {μi, όπου i = 1, 2, …, n}), και το κλειδί γνωστό σε αποστολέα και παραλήπτη, (ένα άλλο σύνολο δυαδικών ψηφίων {κi, όπου i = 1, 2, …, n}). Αν δημιουργήσουμε τον γρίφο που θα αποσταλεί, (ένα σύνολο δυαδικών ψηφίων γi, που να ικανοποιούν την σχέση {γi = μi ⊕ κi, όπου i = 1, 2, …, n}), τότε θα ισχύει επίσης ότι {μi = γi ⊕ κi, όπου i = 1, 2, …, n} και ο παραλήπτης του γρίφου με χρήση του κλειδιού θα αναδημιουργήσει το μήνυμα.
Μηνύματα μεγάλου μήκους μπορούν να κρυπτογραφούνται σε ομάδες των n δυαδικών ψηφίων. Το σύμβολο ⊕ συμβολίζει την πράξη αποκλειστικό Ή (XOR) που περιγράφεται στο άρθρο Λογικές συναρτήσεις.

Ασύμμετρα κρυπτοσυστήματα:

Το ασύμμετρο κρυπτοσύστημα ή κρυπτοσύστημα δημοσίου κλειδιού δημιουργήθηκε για να καλύψει την αδυναμία μεταφοράς κλειδιών που παρουσίαζαν τα συμμετρικά συστήματα. Χαρακτηριστικό του είναι ότι έχει δυο είδη κλειδιών ένα ιδιωτικό και ένα δημόσιο. Το δημόσιο είναι διαθέσιμο σε όλους ενώ το ιδιωτικό είναι μυστικό. Η βασική σχέση μεταξύ τους είναι : ότι κρυπτογραφεί το ένα, μπορεί να το αποκρυπτογραφήσει μόνο το άλλο (Σχ. 2.8).

Τα στάδια της επικοινωνίας του σχήματος 2.8 είναι τα ακόλουθα:
1. Η γεννήτρια κλειδιών του Μένιου παράγει 2 ζεύγη κλειδιών,
2. Η γεννήτρια κλειδιών της Ελένης παράγει 2 ζεύγη κλειδιών
3. Η Ελένη και ο Μένιος ανταλλάσσουν τα δημόσια ζεύγη
4. Ο Μένιος δημιουργεί ένα μήνυμα όπου τα σύμβολα m ανήκουν στον χώρο των μηνυμάτων.
5. Κρυπτογραφεί το μήνυμα με το δημόσιο κλειδί της Ελένης και η παραγόμενη κρυπτοσυμβολοσειρά αποστέλλεται
6. Η Ελένη λαμβάνει την κρυπτοσυμβολοσειρά και στην συνέχεια με το ιδιωτικό της κλειδί την αποκρυπτογραφεί και η έξοδος που παράγεται είναι το μήνυμα.



Σχήμα 2.8:Μοντέλο Ασύμμετρου Κρυπτοσυστήματος.

2.3.1. Εφαρμογές Κρυπτογραφίας.

Η εξέλιξη της χρησιμοποίησης της κρυπτογραφίας ολοένα αυξάνεται καθιστώντας πλέον αξιόπιστη την μεταφορά της πληροφορίας για διάφορους λειτουργικούς σκοπούς:

- Ασφάλεια συναλλαγών σε τράπεζες δίκτυα - ATM
- Κινητή τηλεφωνία (ΤΕΤΡΑ-ΤΕΤΡΑΠΟΛ-GSM)
- Σταθερή τηλεφωνία (crypto phones)
- Διασφάλιση Εταιρικών πληροφοριών
- Στρατιωτικά δίκτυα (Τακτικά συστήματα επικοινωνιών μάχης)
- Διπλωματικά δίκτυα (Τηλεγραφήματα)
- Ηλεκτρονικές επιχειρήσεις (πιστωτικές κάρτες, πληρωμές)
- Ηλεκτρονική ψηφοφορία
- Ηλεκτρονική δημοπρασία
- Ηλεκτρονικό γραμματοκιβώτιο
- Συστήματα συναγερμών
- Συστήματα βιομετρικής αναγνώρισης
- Έξυπνες κάρτες
- Ιδιωτικά δίκτυα (VPN)
- Word Wide Web
- Δορυφορικές εφαρμογές (δορυφορική τηλεόραση)
- Ασύρματα δίκτυα (Hipperlan, Bluetooth, 802.11x)
- Συστήματα ιατρικών δεδομένων και άλλων βάσεων δεδομένων
- Τηλεσυνδιάσκεψη - Τηλεφωνία μέσω διαδικτύου (VOIP)

2.4. Τρόποι και Μέθοδοι Κρυπτογράφησης.

Οι τρόποι κρυπτογράφησης μυστικού κλειδιού τυπικά χωρίζονται σε 2 κατηγορίες. Η πρώτη περιλαμβάνει διαδικασίες κρυπτογράφησης που εφαρμόζονται πάνω σε ένα μοναδικό bit (ή byte ή word) και υλοποιούν κάποιο μηχανισμό ανατροφοδότησης έτσι ώστε το κλειδί να αλλάζει συνεχώς. Για αυτό και ονομάζονται κρυπτογράφοι ροής (stream ciphers). Η δεύτερη κατηγορία (κρυπτογράφοι μπλοκ - block ciphers) αποτελείται από αλγορίθμους κρυπτογράφησης που λειτουργούν πάνω σε ομάδες δεδομένων κάθε χρονική στιγμή χρησιμοποιώντας το ίδιο κλειδί για κάθε ομάδα. Έτσι στην γενική περίπτωση, όταν το ίδιο μυστικό κλειδί χρησιμοποιείται, η ίδια ομάδα δεδομένων ενός plaintext θα κρυπτογραφηθεί στο ίδιο ciphertext όταν η κρυπτογράφηση γίνεται με έναν αλγόριθμο κρυπτογράφησης μπλοκ αλλά σε διαφορετικό ciphertext όταν χρησιμοποιηθεί ένας κρυπτογράφος ροής. Για τους κρυπτογράφους μπλοκ, έχουν επινοηθεί αρκετοί τρόποι λειτουργίας (modes) ώστε να βελτιωθούν κάποια χαρακτηριστικά τους όπως η ασφάλεια που προσφέρουν ή να γίνουν πιο κατάλληλοι για διάφορες εφαρμογές. Τέσσερις είναι οι κυριότεροι τρόποι λειτουργίας :

Electronic Codebook (ECB)
Αυτός ο τρόπος λειτουργίας είναι ο απλούστερος και ο πλέον προφανής. Το μυστικό κλειδί χρησιμοποιείται για την κρυπτογράφηση κάθε μπλοκ δεδομένων του plaintext. Κατά συνέπεια με την χρήση του ίδιου κλειδιού, το ίδιο plaintext μπλοκ θα μετατρέπεται πάντα στο ίδιο ciphertext μπλοκ. Είναι ο πλέον κοινός τρόπος λειτουργίας των κρυπτογράφων μπλοκ γιατί είναι ο απλούστερος και άρα ο πιο εύκολα υλοποιήσιμος και συνάμα ο πιο γρήγορος καθώς δεν χρησιμοποιείται κάποιου είδους ανατροφοδότηση. Μειονέκτημα του είναι ότι είναι ο πιο ευάλωτος τρόπος κρυπτογράφησης σε επιθέσεις τύπου brute-force (ως επίθεση brute-force θεωρείται η προσπάθεια εύρεσης του μυστικού κλειδιού με την εξαντλητική δοκιμή πιθανών κλειδιών).

Cipher Block Chaining (CBC)

Χρησιμοποιώντας την CBC λειτουργία, προστίθεται σε έναν κρυπτογράφο μπλοκ ένας μηχανισμός ανατροφοδότησης. Ο τρόπος αυτός λειτουργίας ορίζει ότι προτού να γίνει η κρυπτογράφηση ενός νέου μπλοκ plaintext, γίνεται XOR (αποκλειστικό-Ή) του μπλοκ αυτού και του ciphertext μπλοκ που μόλις πριν έχει παραχθεί. Με τον τρόπο αυτό, 2 ταυτόσημα μπλοκ plaintext δεν κρυπτογραφούνται ποτέ στο ίδιο ciphertext. Σε σχέση με τον ECB προσφέρεται μεγαλύτερη ασφάλεια, με κόστος όμως κυρίως στην ταχύτητα κρυπτογράφησης καθώς για να ξεκινήσει η επεξεργασία ενός μπλοκ plaintext είναι απαραίτητο να έχει ολοκληρωθεί πλήρως η κρυπτογράφηση του προηγούμενου μπλοκ. Αποτρέπεται έτσι η χρήση τεχνικών pipelining (software ή hardware) που μπορούν να επιταχύνουν την διαδικασία.

Cipher Feedback (CFB)

Ο τρόπος αυτός λειτουργίας επιτρέπει σε έναν κρυπτογράφο μπλοκ να συμπεριφερθεί σαν ένας κρυπτογράφος ροής. Αυτό είναι θεμιτό όταν πρέπει να κρυπτογραφούνται δεδομένα που μπορεί να έχουν μέγεθος μικρότερο από ένα μπλοκ. Παράδειγμα τέτοιας εφαρμογής μπορεί να είναι η διαδικασία κρυπτογράφησης ενός terminal session. Περιληπτικά, κατά την CFB λειτουργία χρησιμοποιείται ένας shift καταχωρητής στο μέγεθος του block μέσα στον οποίο τοποθετούνται τα δεδομένα προς κρυπτογράφηση. Όλος ο καταχωρητής κρυπτογραφείται και αυτό που προκύπτει είναι το ciphertext. Η ποσότητα των δεδομένων που μπαίνουν μέσα στον shift καταχωρητή καθορίζεται από την εφαρμογή.

Output Feedback (OFB)

Στόχος και αυτού του τρόπου λειτουργίας των μπλοκ κρυπτογράφων είναι να εξασφαλίσει ότι το ίδιο plaintext μπλοκ δεν μπορεί να παράγει το ίδιο ciphertext μπλοκ. Σε σχέση με το CBC, χρησιμοποιείται και εδώ ένας μηχανισμός ανατροφοδότησης παρόλα αυτά είναι εσωτερικός και ανεξάρτητος από τα plaintext και ciphertext δεδομένα.

Σημαντικοί αλγόριθμοι αυτής της κατηγορίας είναι οι DES (Data Encryption Standard), 3DES, DESX, ο AES (Advanced Encryption Standard), οι RC2, RC4, RC5 και IDEA (International Data Encryption Algorithm). Οι αλγόριθμοι της σειράς DES είναι οι πλέον χρησιμοποιούμενοι σήμερα αλγόριθμοι, αν και πλέον αντικαθιστούνται από τον AES. Επινοήθηκαν από την IBM την δεκαετία του '70 και υιοθετήθηκαν από το National Bureau of Standards (νυν NIST) των ΗΠΑ. Οι DES αλγόριθμοι χρησιμοποιούν κλειδιά μήκους 56 bits (ο 3DES και ο DESX επεκτείνουν κατάλληλα αυτόν τον αριθμό χρησιμοποιώντας περισσότερα κλειδιά) και επεξεργάζονται μπλοκ των 64 bits. Ο AES αλγόριθμος είναι το πρότυπο που καθιερώθηκε από το NIST ως διάδοχος του DES και πλέον αποτελεί τον προτεινόμενο αλγόριθμο κρυπτογράφησης για εφαρμογές υψηλής ασφάλειας. Οι αλγόριθμοι RC είναι αλγόριθμοι μεταβλητού κλειδιού από την RSA Security ενώ ο IDEA χρησιμοποιείται στο πρότυπο PGP (Pretty Good Privacy).

3. Το Πρότυπο AES.

3.1. Γενικά.

Το πρότυπο AES περιγράφει μια συμμετρική μπλοκ διαδικασία κρυπτογράφησης μυστικού κλειδιού. Το πρότυπο υποστηρίζει την χρήση κλειδιών μήκους 128, 192 και 256 bits. Ανάλογα με το ποιο μήκος κλειδιού χρησιμοποιείται, συνήθως χρησιμοποιείται η συντόμευση AES-128, AES-192 και AES-256 αντίστοιχα. Ανεξάρτητα από το μήκος κλειδιού, ο αλγόριθμος επενεργεί πάνω σε μπλοκ δεδομένων μήκους 128 bits. Η διαδικασία κρυπτογράφησης είναι επαναληπτική. Αυτό σημαίνει ότι σε κάθε μπλοκ δεδομένων γίνεται μια επεξεργασία η οποία επαναλαμβάνεται έναν αριθμό από φορές ανάλογα με το μήκος κλειδιού. Κάθε επανάληψη ονομάζεται γύρος (round). Στον πρώτο γύρο επεξεργασίας ως είσοδος είναι ένα plaintext μπλοκ και το αρχικό κλειδί, ενώ στους γύρους που ακολουθούν ως είσοδος είναι το μπλοκ που έχει προκύψει από τον προηγούμενο γύρο καθώς και ένα κλειδί που έχει παραχθεί από το αρχικό με βάση κάποια διαδικασία που ορίζει ο αλγόριθμος. Το τελικό προϊόν της επεξεργασίας είναι το κρυπτογραφημένο μπλοκ (ciphertext). Το μπλοκ αυτό πρέπει να σημειωθεί ότι έχει ακριβώς το ίδιο μέγεθος (128 bits) με το plaintext μπλοκ.

3.2. Είσοδοι, Έξοδοι και Εσωτερική Κατάσταση.

Όπως ήδη αναφέρθηκε, ο AES τροφοδοτείται με ακολουθίες από bits των 128 bits (μπλοκ) καθώς και από κλειδιά, που μπορεί να έχουν μέγεθος 128, 192 ή 256 bits. Τα κλειδιά αυτά ονομάζονται κλειδιά κρυπτογράφησης (cipher keys) για να διαχωριστούν από τα κλειδιά που παράγονται κατά την λειτουργία του αλγορίθμου.
Η βασική μονάδα επεξεργασίας στον AES είναι το byte. Έτσι τα bits ενός μπλοκ ή ενός κλειδιού χωρίζονται σε ομάδες των 8 για να σχηματιστούν τα bytes. Κάθε byte στον AES αντιστοιχεί σε ένα πολυώνυμο (αριθμητική πεπερασμένων πεδίων - finite field arithmetic). Αν υποθέσουμε ότι τα bits που αποτελούν ένα byte είναι τα {b7, b6, b5, b4, b3, b2, b1, b0}, τότε το byte αυτό αναπαριστά το πολυώνυμο :

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0 = \sum_{i=0}^{7} b_i x^i$$

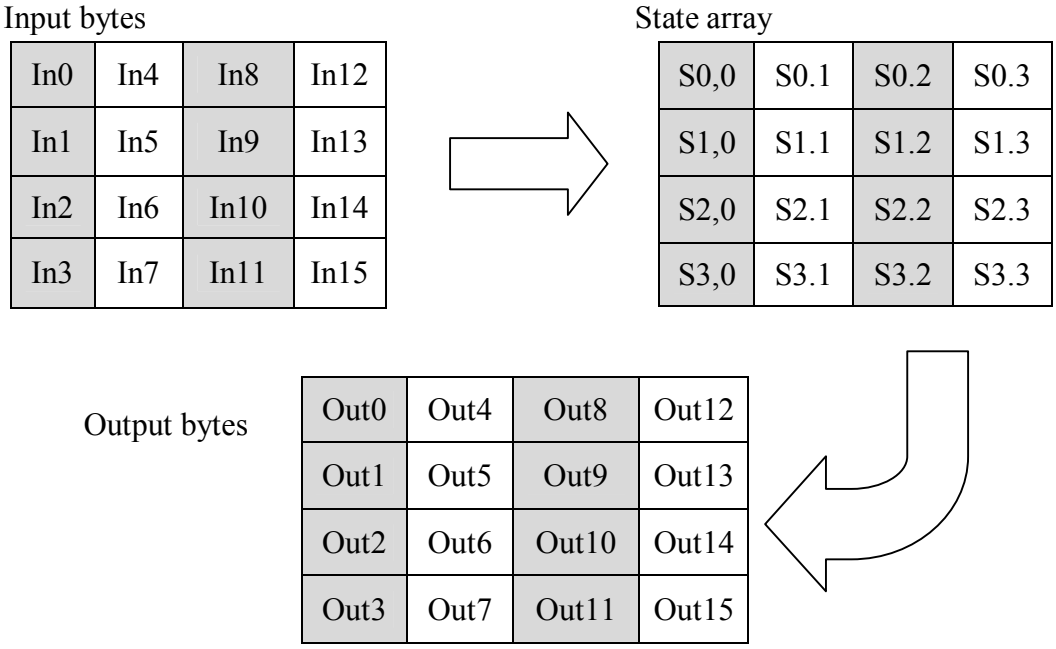
Έτσι για παράδειγμα το byte {11001101} αντιστοιχεί στο πολυώνυμο $x^7 + x^6 + x^3 + x^2 + 1$.
Κλείνοντας την αναφορά στις μονάδες των δεδομένων που διαχειρίζεται ο AES, πρέπει να αναφερθεί το πώς γίνεται η δεικτοδότηση των bits και των bytes στα μπλοκ και στα κλειδιά. Το Σχήμα 2 δείχνει την αντιστοιχία.

Input bit sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…….
Byte number	0								1								…….
Bit numbers in byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	…….

Σχήμα 3.1: Δεικτοδότηση των bits και bytes.

Όλες οι λειτουργίες που επιτελεί ο αλγόριθμος γίνονται πάνω σε ένα δισδιάστατο πίνακα που αποκαλείται Κατάσταση (State). Ο πίνακας αυτός περιλαμβάνει τέσσερις γραμμές από bytes, με κάθε μία γραμμή να αποτελείται από Nb bytes. Ο αριθμός που αντιστοιχεί στην ποσότητα Nb υπολογίζεται αν διαιρεθεί το μήκος του μπλοκ με το 32. Εφόσον στον AES υποστηρίζονται μπλοκ μεγέθους μόνο 128 bits, το Nb θα έχει τιμή 4.

Το μπλοκ εισόδου περιλαμβάνει 16 bytes, τα οποία δεικτοδοτούνται in0 έως in15. Το κρυπτογραφημένο μπλοκ εξόδου περιλαμβάνει επίσης 16 bytes που δεικτοδοτούνται ως out0 έως out15. Η State χρησιμοποιεί την μεταβλητή s με δύο δείκτες που δηλώνουν την θέση κάθε byte στον πίνακα. Η πρώτη λοιπόν και τελευταία λειτουργία που μπορεί να υποτεθεί ότι γίνεται στον AES είναι να αντιστοιχηθούν τα bytes εισόδου σε κάποια θέση του πίνακα της State και το αντίστροφο στην έξοδο. Το Σχήμα 3.2 δείχνει πώς γίνεται αυτό.



Σχήμα 3.2: Αντιστοίχηση των bytes εισόδου σε κάποια θέση του πίνακα της State και το αντίστροφο στην έξοδο.

Η αντιστοίχηση που περιγράφηκε παραπάνω μπορεί να περιγραφεί μαθηματικά. Η αντιστοίχηση εισόδου στην State περιγράφεται από την σχέση :

$s[r,c] = in[r+4c]$ για $0 \le r < 4$ και $0 \le c < Nb$

ενώ η αντιγραφή της State στην έξοδο από την σχέση :
$out[r+4c] = s[r,c]$ για $0 \le r < 4$ και $0 \le c < Nb$

Ένας άλλος τρόπος να δει κάποιος τα περιεχόμενα της State είναι σαν 32-bit λέξεις (words) αντί για byte. Μια 32-bit word περιλαμβάνει τα 4 bytes μιας στήλης, οπότε τα 4 words που αποτελούν την State είναι τα ακόλουθα :

w(0) = s(0,0) s(1,0) s(2,0) s(3,0)
w(1) = s(0,1) s(1,1) s(2,1) s(3,1)
w(2) = s(0,2) s(1,2) s(2,2) s(3,2)
w(3) = s(0,3) s(1,3) s(2,3) s(3,3)

Περιγραφή Αλγορίθμου: Όπως αναφέρθηκε στην προηγούμενη ενότητα, το πρότυπο AES ορίζει ότι τα μπλοκ που επεξεργάζεται ο αλγόριθμος έχουν μέγεθος 128 bits και αυτό ορίζεται από την ποσότητα Nb = 4, που συμβολίζει τον αριθμό των 32-bit λέξεων στο μπλοκ. Από την άλλη, τα κλειδιά που χρησιμοποιούνται για την κρυπτογράφηση, μπορούν να έχουν μήκος 128, 192 ή 256 bits. Η μεταβλητή Nk συμβολίζει τον αριθμό των 32-bit λέξεων που μπορεί να περιλαμβάνει ένα κλειδί και κατά συνέπεια μπορεί να πάρει τις τιμές 4, 6 και 8. Ανάλογα με το μήκος κλειδιού που θα επιλεχθεί

για την κρυπτογράφηση, ο αλγόριθμος ορίζει έναν αριθμό από γύρους επεξεργασίας που απαιτούνται για την ολοκλήρωση της. Η μεταβλητή Nr χρησιμοποιείται για να δηλώσει το πλήθος των γύρων. Αν χρησιμοποιηθεί μήκος κλειδιού 128 bits τότε απαιτούνται 10 γύροι επεξεργασίας. Για μήκη κλειδιού ίσα με 192 και 256 bits απαιτούνται αντίστοιχα 12 και 14 γύροι.

	Key Length (Nk words)	**Block Size (Nb words)**	**Number Of Rounds (Nr)**
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Σχήμα 3.3: Μέγεθος των μεταβλητών του αλγορίθμου.

Nr = Αριθμός των γύρων
Nb = Αριθμός των byte
Nk = Μέγεθος κλειδιού

Να σημειωθεί ότι οι παραπάνω συνδυασμοί μήκους μπλοκ, μήκους κλειδιού και γύρων επεξεργασίας είναι αυτοί που ορίζονται αυστηρά στο πρότυπο AES. Ο αλγόριθμος κρυπτογράφησης Rijndael στον οποίο βασίζεται ο AES δίνει την δυνατότητα πραγματοποίησης περισσότερων συνδυασμών. Έτσι, καταλαβαίνει κανείς ότι ο AES ουσιαστικά ορίζει ένα υποσύνολο του αλγορίθμου Rijndael.
Τόσο κατά την διάρκεια της διαδικασίας κρυπτογράφησης όσο και αποκρυπτογράφησης, κάθε γύρος επεξεργασίας αποτελείται από μια σειρά μετασχηματισμών σε επίπεδο byte. Για την ακρίβεια, χρησιμοποιούνται 4 τύποι μετασχηματισμών :

- ένας μετασχηματισμός αντικατάστασης bytes χρησιμοποιώντας κάποιον σχετικό πίνακα αντικατάστασης
- ένας μηχανισμός ολίσθησης των bytes της State κατά διαφορετικά offsets
- μια διαδικασία ανάμειξης των bytes της State
- μια πρόσθεση ενός κλειδιού στην State

3.3. Ο Αλγόριθμος Κρυπτογράφησης.

Στην αρχή της διαδικασίας κρυπτογράφησης ένα μπλοκ εισόδου (plaintext) αντιγράφεται στην State. Μετά από έναν αρχικό γύρο πρόσθεσης κλειδιού, ακολουθούν 10, 12 ή 14 γύροι επεξεργασίας, με τον τελευταίο γύρο να διαφέρει από τους υπόλοιπους. Η τελική State αντιγράφεται στην έξοδο και η επεξεργασία για το συγκεκριμένο block ολοκληρώνεται (παραγωγή του ciphertext μπλοκ).
Το μυστικό κλειδί κρυπτογράφησης που χρησιμοποιείται σαν είσοδος στον αλγόριθμο είναι το κλειδί που προστίθεται στο μπλοκ εισόδου πριν αρχίσει η επεξεργασία. Σε καθέναν από τους γύρους επεξεργασίας, όπως αναφέρθηκε παραπάνω, υπάρχει μια φάση κατά την οποία προστίθεται στο μπλοκ και ένα κλειδί. Το κλειδί που προστίθεται στις περιπτώσεις αυτές, δεν είναι το αρχικό μυστικό κλειδί αλλά κάποιο που έχει προκύψει με μια συγκεκριμένη διαδικασία από το μυστικό κλειδί και είναι διαφορετικό για κάθε γύρο. Για τον λόγο αυτό, τα κλειδιά αυτά ονομάζονται round keys. Η διαδικασία με την οποία προκύπτουν τα round κλειδιά ονομάζεται Επέκταση Κλειδιού και θα αναλυθεί σε επόμενη ενότητα.
Αυτό που πρέπει να διευκρινιστεί είναι η έννοια της πρόσθεσης στον AES αλγόριθμο. Σε προηγούμενη ενότητα έχει αναφερθεί ότι τα bytes της πληροφορίας κατά την επεξεργασία τους λαμβάνονται ως πολυώνυμα. Έτσι, η πράξη της πρόσθεσης είναι ουσιαστικά μια διαδικασία πρόσθεσης πολυωνύμων. Η πρόσθεση μεταξύ πολυωνύμων πραγματοποιείται με την πρόσθεση των συντελεστών των αντίστοιχων όρων (δυνάμεων) των πολυωνύμων. Η πρόσθεση γίνεται modulo-2,

δηλαδή μέσω μιας XOR πράξης. Να ενθυμηθεί ότι η XOR πράξη μεταξύ δύο bits (συμβολίζεται με) έχει τον εξής πίνακα αληθείας :

$0 \oplus 0 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$
$1 \oplus 1 = 0$

Αν κάθε βασικός μετασχηματισμός του AES αναπαρασταθεί από μια συνάρτηση που επενεργεί στην State, τότε ο αλγόριθμος κρυπτογράφησης μπορεί να περιγραφεί από τον ψευδοκώδικα που παρουσιάζεται στο Σχήμα 3.4.

```
Cipher (byte in [4*Nb], byte out [4*Nb], byte key [4*Nb]
        word w [Nb*(Nr+1)])
Begin
     Byte state [4, Nb]
     State=in
     KeyExpansion(key,w);
     AddRoundKey (state, w [0, Nb-1])

     For round=1 step 1 to Nr-1
       SubBytes (state)
       ShiftRows (state)
       MixColumns (state)
       AddRoundKey (state, w [round*Nb, (round+1)*Nb-1])
     End for

     SubBytes (state)
     ShiftRows (state)
     AddRoundKey (state, w [Nr*Nb, (Nr+1)*Nb-1]
     Out=state
 End
```

Σχήμα 3.4 Ψευδοκώδικας Κρυπτογράφησης.

Οι συναρτήσεις αυτές αναφέρονται ως SubBytes(), ShiftRows(), MixColumns() και AddRoundKey() και αντιστοιχούν (με αυτήν την σειρά) στους μετασχηματισμούς 1 έως 4 όπως αναφέρθηκαν παραπάνω. Να σημειωθεί ότι το array w χρησιμοποιείται για να δηλώσει την συλλογή των round keys που παράγονται από την διαδικασία επέκτασης κλειδιού.Εξήγηση του ψευδοκώδικα για την διαδικασία κρυπτογράφησης.Όπως βλέπουμε παίρνει σαν είσοδο (In) το plaintext βγάζει σαν έξοδο (out) το cipher text και παίρνουμε και τον πινάκα W ο οποίος έχει δημιουργηθεί κατά την λειτουργία της συνάρτησης key expansion όπου εκεί έχει ανακατευτεί το κλειδί που έχουμε δώσει.
Λοιπόν στον γύρο 0 στο state όρισμα αντιγράφουμε το plaintext και στη συνέχεια καλούμε τη συνάρτηση AddRoundKey.
Μετά από το γύρο 1 έως τον Nr-1 (ανάλογα το bits αλγορίθμου που έχουμε επιλέξει για να δουλέψουμε 128,196,256) καλούμε με τη σειρά τις συναρτήσεις SubBytes, Shiftrows, MixColumns και AddRoundKey όπου εκεί θα ανακατευτεί διαδοχικά το state.
Στον τελευταίο γύρο το state θα ανακατευτεί με τις συναρτήσεις SubBytes, Shiftrows και AddRoundKey όπου το τελικό αποτέλεσμα θα είναι το cipher text.

3.3.1. Συναρτήσεις του Αλγορίθμου Κρυπτογράφησης.

3.3.1.1. Ο Μετασχηματισμός SubBytes.

X/Y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	fb	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	fd	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	fd	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Σχήμα 3.5: Ο πίνακας αντικατάστασης S-Box.

Ο μετασχηματισμός SubBytes αποτελεί μια μη γραμμική αντικατάσταση των bytes της State με την χρήση ενός πίνακα αντικατάστασης (S-Box). Οι τιμές του πίνακα αυτού (που είναι αντιστρέψιμος) υπολογίζονται με την σύνθεση των δύο ακόλουθων μετασχηματισμών παίρνοντας τον πολλαπλασιαστικό αντίστροφο στο πεπερασμένο πεδίο GF(28) - με το στοιχείο {00} να αντιστοιχίζεται στον εαυτό του εφαρμόζοντας τον ακόλουθο μετασχηματισμό (στο GF(2)):

$$b_i' = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

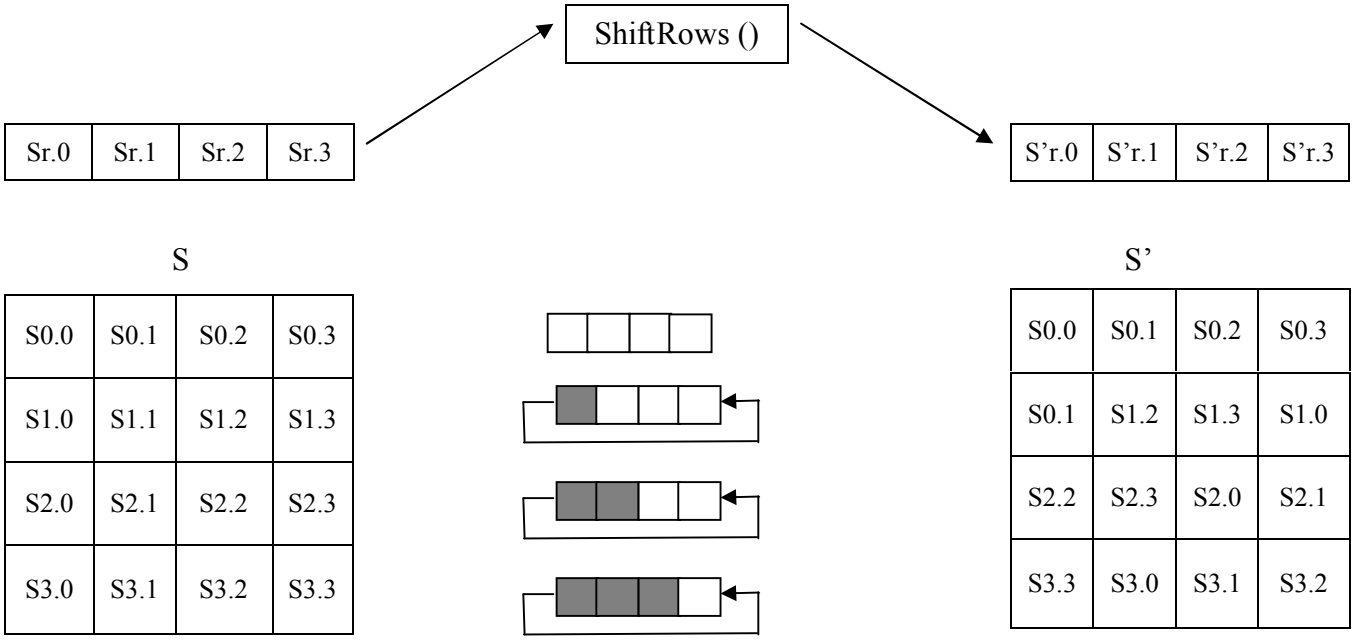
για $0 \le i < 8$, όπου bi είναι το i-οστό bit του byte, και ci είναι το i-οστό bit του byte c με την τιμή {63} or {01100011}.
Ο πίνακας S-Box τυπικά δεν υπολογίζεται κατά την διαδικασία της κρυπτογράφησης, αλλά οι τιμές του έχουν προϋπολογιστεί. Στο Σχήμα 3.5 που ακολουθεί παρατίθενται οι τιμές του πίνακα S-Box όπως τις παρουσιάζει το NIST στο επίσημο έγγραφο για τον AES. Για όσους από τους αναγνώστες δεν είναι εξοικειωμένοι με την γραφή αριθμών στο δεκαεξαδικό σύστημα, να σημειωθεί ότι ένας αριθμός στο δεκαεξαδικό σύστημα χρειάζεται 4 bits για να αναπαρασταθεί. Κατά συνέπεια, ένα byte αναπαρίσταται από 2 δεκαεξαδικά ψηφία χωρίζοντας το σε 2 ομάδες των 4 bits. Έτσι η γραμμή x του πίνακα αναφέρεται στα πρώτα 4 bits του byte και η στήλη y στα επόμενα 4.
Εφόσον αναφέρθηκε ο θεωρητικός τρόπος με τον οποίο προκύπτει ο πίνακας S-Box, καλό θα ήταν να διευκρινιστεί τι σημαίνει πολλαπλασιασμός στο GF(28). Είναι ο πολλαπλασιασμός μεταξύ πολυωνύμων modulo ένα irreducible πολυώνυμο βαθμού 8. Irreducible ονομάζεται ένα πολυώνυμο αν διαιρείται μονάχα από τον εαυτό του και την μονάδα. Το πολυώνυμο που έχει επιλεχθεί για το AES είναι το :

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Η modulo πράξη εξασφαλίζει ότι το πολυώνυμο που θα προκύψει θα είναι ένα δυαδικό πολυώνυμο βαθμού μικρότερου του 8, άρα θα μπορεί να αναπαρασταθεί από ένα byte. Να σημειωθεί ότι το ουδέτερο στοιχείο της πράξης είναι το {01} και ότι το σύμβολο που χρησιμοποιείται για να διακρίνει την πράξη αυτή από έναν κοινό αριθμητικό πολλαπλασιασμό είναι το ● .

3.3.1.2. Ο Μετασχηματισμός ShiftRows.

Ο μετασχηματισμός αυτός επιβάλει την κυκλική ολίσθηση των bytes των γραμμών της State. Η πρώτη γραμμή παραμένει ανέπαφη, ενώ στις υπόλοιπες τα bytes ολισθαίνουν με διαφορετικό offset. Το Σχήμα 6 παρουσιάζει ενδεικτικά πώς γίνεται ο μετασχηματισμός αυτός. Όπως μπορεί να παρατηρηθεί από το Σχήμα, η δεύτερη γραμμή ολισθαίνει αριστερά κατά μία θέση με αποτέλεσμα το πρώτο byte της γραμμής να βρεθεί τελευταίο (κυκλική ολίσθηση). Με αντίστοιχο τρόπο ολισθαίνουν και οι γραμμές 3 και 4 αλλά κατά 2 και 3 θέσεις αντίστοιχα.



Σχήμα 3.6: Ο μετασχηματισμός ShiftRows ολισθαίνει κυκλικά προς τα αριστερά τις τρεις τελευταίες γραμμές του State.

3.3.1.3. Ο Μετασχηματισμός MixColumns.

Ο μετασχηματισμός αυτός εφαρμόζεται στις στήλες της State. Η κάθε στήλη θεωρείται σαν πολυώνυμο τρίτης τάξης με συντελεστές τις τιμές των bytes της στήλης.

$$s(x)_i = s_{3,i} \cdot x^3 + s_{2,i} \cdot x^2 + s_{1,i} \cdot x + s_{0,i}$$

Τα πολυώνυμα πολλαπλασιάζονται modulo (x4 + 1) με ένα καθορισμένο πολυώνυμο που δίνεται από την σχέση :

$$a(x) = \{03\} \cdot x^3 + \{01\} \cdot x^2 + (01) \cdot x + (02)$$

Η διαδικασία αυτή του υπολογισμού της αρχικής πράξης, $s'(x) = a(x) \otimes s(x)$ όπου με $\otimes$ συμβολίζεται ο modulo πολλαπλασιασμός μετασχηματίζεται τελικά στις εξής σχέσεις :

$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$

$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$

$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$

$s'_{2,c} = \{03\} \bullet s_{0,c} \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$

για $0 \leq c < Nb$

3.3.1.4. Ο Μετασχηματισμός AddRoundKey.

Ο μετασχηματισμός αυτός επιβάλει την πρόσθεση της τιμής της ποσότητας round key στα bytes των στηλών του πίνακα State. Επειδή κάθε τιμή του round key αποτελείται από Nb λέξεις, επιλέγεται κάθε φορά η επιθυμητή λέξη. Η πράξη αυτή υλοποιείται σαν απλή XOR πράξη ανάμεσα στα bits των ποσοτήτων (bitwise XOR). Η πράξη αυτή μεταφράζεται μαθηματικά στην εξής σχέση:

$$[S'_{0,c}, S'_{1,c}, S'_{2,c}, S'_{3,c}] = [S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] \oplus [W_{round*Nb+c}]$$

για $0 \leq c < Nb$

3.4. Ο Αλγόριθμος Αποκρυπτογράφησης.

Οι μετασχηματισμοί της διαδικασίας κρυπτογράφησης (όπως περιγράφηκαν στην προηγούμενη ενότητα) μπορούν να αντιστραφούν και να τοποθετηθούν σε αντίστροφη σειρά ώστε να παραχθεί μια διαδικασία που θα αποκρυπτογραφεί ένα ciphertext του AES. Έτσι όπως και κατά την κρυπτογράφηση, υπάρχουν τέσσερις διακριτοί μετασχηματισμοί που επενεργούν πάνω στην State κατά την αποκρυπτογράφηση, οι InvShiftRows, InvSubBytes, InvMixColumns και AddRoundKey. Στο παρακάτω σχήμα παρουσιάζεται ο ψευδοκώδικας που περιγράφει την διαδικασία αποκρυπτογράφησης. Να σημειωθεί ότι το array w που εμφανίζεται είναι το ίδιο ακριβώς array με τα κλειδιά (cipher και round) που χρησιμοποιήθηκε και κατά την κρυπτογράφηση. Η διαδικασία παραγωγής των κλειδιών είναι ταυτόσημη με αυτή που περιγράφηκε στην προηγούμενη ενότητα.

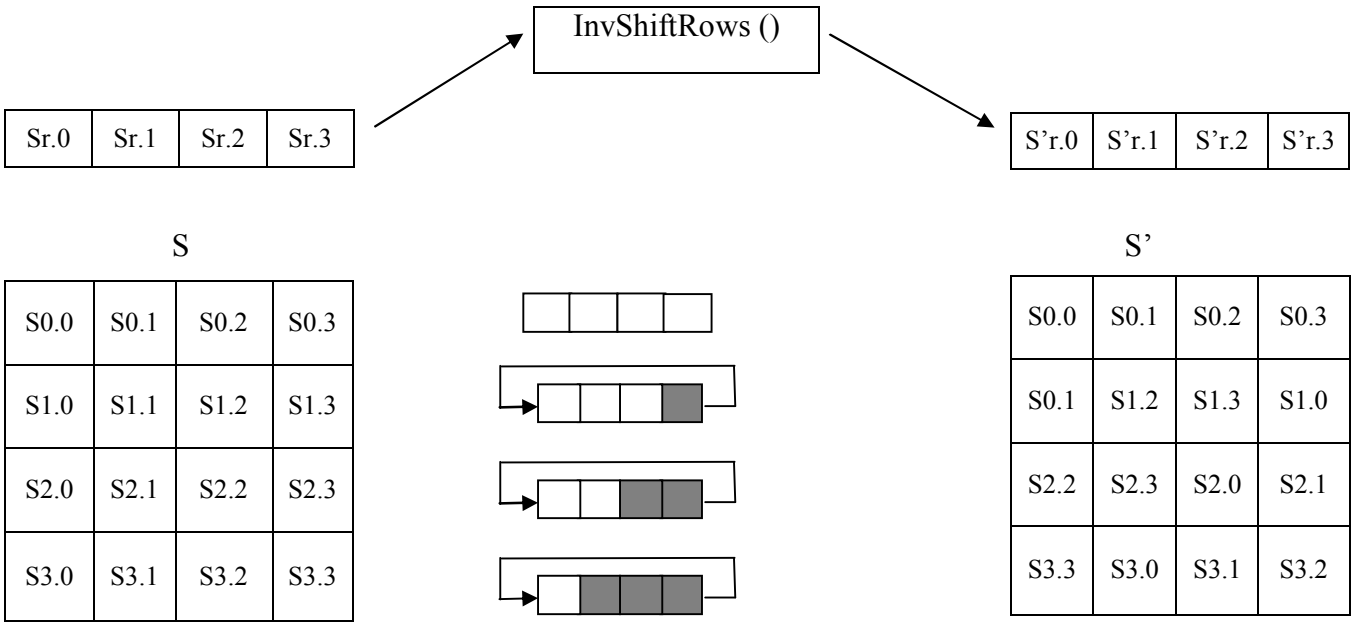
```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
Begin
     Byte state [4, Nb]
            State=in
     AddRoundKey (state, w [Nr*Nb, (Nr+1)*Nb-1])

     for round=Nr-1 step -1 downto 1
          InvShiftRows (state)
          InvSubBytes (state)
          AddRoundKey (state, w [round*Nb, (round+1)*Nb-1])
          InvMixColumns (state)
     End for

          InvShiftRows (state)
          InvSubBytes (state)
         AddRoundKey (state, w [0, Nb-1])
         Out=state
 End
```

Σχήμα 3.7: Ψευδοκώδικας Αποκρυπτογράφησης

3.4.1. Συναρτήσεις του Αλγορίθμου Αποκρυπτογράφησης.

3.4.1.1. Ο Μετασχηματισμός InvShiftRows.

Ο μετασχηματισμός InvShiftRows είναι ο αντίστροφος του ShiftRows της διαδικασίας κρυπτογράφησης. Τα bytes στις τελευταίες τρεις γραμμές της State ολισθαίνουν κατά διαφορετικά offsets, με αντίθετη φορά από ότι στην ShiftRows διαδικασία. Στο Σχήμα 3.8 παρουσιάζεται ο ακριβής μηχανισμός.



Σχήμα 3.8: Ο μετασχηματισμός InvShiftRows.

3.4.1.2. Ο Μετασχηματισμός InvSubBytes.

Ο μετασχηματισμός αυτός, όπως δηλώνει και το όνομα του, είναι ο αντίστροφος του μετασχηματισμού αντικατάστασης bytes της κρυπτογράφησης. Έτσι, στην περίπτωση αυτή, αντί για το S-Box πίνακα χρησιμοποιείται ο αντίστροφος του (inverse S-Box), ο οποίος και παρουσιάζεται στο Σχήμα 3.9.

X/Y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	Ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	A0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Σχήμα 3.9: Ο πίνακας inverse S-Box.

3.4.1.3. Ο Μετασχηματισμός InvMixColumns.

Είναι ο αντίστροφος του μετασχηματισμού MixColumns. Όπως και ο MixColumns εφαρμόζεται πάνω στις στήλες της State, θεωρώντας κάθε μία από αυτές ένα πολυώνυμο τεσσάρων όρων. Κάθε στήλη, θεωρείται πολυώνυμο του GF(28) και πολλαπλασιάζεται modulo x4 + 1 με ένα καθορισμένο πολυώνυμο a-1(x) :

$$a^{-1}(x) = \{0b\} + \{0d\} \cdot x^2 + \{09\} \cdot x + \{0e\}$$

Σαν αποτέλεσμα του παραπάνω πολλαπλασιασμού, τα 4 bytes σε μια στήλη αντικαθίστανται από τα ακόλουθα bytes :

$$s'_{0,C} = (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$
$$s'_{1,C} = (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c})$$
$$s'_{2,C} = (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c})$$
$$s'_{3,C} = (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c})$$

3.4.1.4. Ο Αντίστροφος Μετασχηματισμός AddRoundKey.

Εφόσον ο μετασχηματισμός αυτός είναι μια απλή XOR πράξη, είναι από μόνος του αντιστρέψιμος και κατά συνέπεια είναι ταυτόσημος με τον μετασχηματισμό που περιγράφηκε στην 3.3.1.4. ενότητα.

3.5. Επέκταση Κλειδιού.

Στη διαδικασία αυτή δεχόμαστε σαν είσοδο το κλειδί όπου επεξεργάζεται και τροποποιείται με μια ακολουθία συναρτήσεων (SubWord, RotWord).
Σκοπός της ‘επέκτασης κλειδιού’ είναι να διαφυλαχτεί η ασφάλεια και η μυστικότητα του κλειδιού κατά τη διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης.

3.6. Παράδειγμα Λειτουργίας Αλγορίθμου.

Legend for CIPHER (ENCRYPT) (round number r = 0 to 10, 12 or 14):
input: cipher input
start: state at start of round[r]
s_box: state after SubBytes()
s_row: state after ShiftRows()
m_col: state after MixColumns()
k_sch: key schedule value for round[r]
output: cipher output

Legend for INVERSE CIPHER (DECRYPT) (round number r = 0 to 10, 12 or 14):
iinput: inverse cipher input
istart: state at start of round[r]
is_box: state after InvSubBytes()
is_row: state after InvShiftRows()
ik_sch: key schedule value for round[r]
ik_add: state after AddRoundKey()
ioutput: inverse cipher output

AES-128 (*Nk*=4, *Nr*=10)

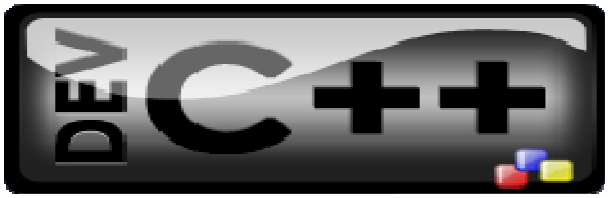
```
PLAINTEXT: 00112233445566778899aabbccddeeff
KEY: 000102030405060708090a0b0c0d0e0f

CIPHER (ENCRYPT):36
round[ 0]. [input]    00112233445566778899aabbccddeeff
round[ 0]. [k_sch]    000102030405060708090a0b0c0d0e0f
round[ 1]. [start]    00102030405060708090a0b0c0d0e0f0
round[ 1]. [s_box]    63cab7040953d051cd60e0e7ba70e18c
round[ 1]. [s_row]    6353e08c0960e104cd70b751bacad0e7
round[ 1]. [m_col]    5f72641557f5bc92f7be3b291db9f91a
round[ 1]. [k_sch]    d6aa74fdd2af72fadaa678f1d6ab76fe
round[ 2]. [start]    89d810e8855ace682d1843d8cb128fe4
round[ 2]. [s_box]    a761ca9b97be8b45d8ad1a611fc97369
round[ 2]. [s_row]    a7be1a6997ad739bd8c9ca451f618b61
round[ 2]. [m_col]    ff87968431d86a51645151fa773ad009
round[ 2]. [k_sch]    b692cf0b643dbdf1be9bc5006830b3fe
round[ 3]. [start]    4915598f55e5d7a0daca94fa1f0a63f7
round[ 3]. [s_box]    3b59cb73fcd90ee05774222dc067fb68
round[ 3]. [s_row]    3bd92268fc74fb735767cbe0c0590e2d
round[ 3]. [m_col]    4c9c1e66f771f0762c3f868e534df256
round[ 3]. [k_sch]    b6ff744ed2c2c9bf6c590cbf0469bf41
round[ 4]. [start]    fa636a2825b339c940668a3157244d17
round[ 4]. [s_box]    2dfb02343f6d12dd09337ec75b36e3f0
round[ 4]. [s_row]    2d6d7ef03f33e334093602dd5bfb12c7
round[ 4]. [m_col]    6385b79ffc538df997be478e7547d691
round[ 4]. [k_sch]    47f7f7bc95353e03f96c32bcfd058dfd
round[ 5]. [start]    247240236966b3fa6ed2753288425b6c
round[ 5]. [s_box]    36400926f9336d2d9fb59d23c42c3950
round[ 5]. [s_row]    36339d50f9b539269f2c092dc4406d23
round[ 5]. [m_col]    f4bcd45432e554d075f1d6c51dd03b3c
round[ 5]. [k_sch]    3caaa3e8a99f9deb50f3af57adf622aa
round[ 6]. [start]    c81677bc9b7ac93b25027992b0261996
round[ 6]. [s_box]    e847f56514dadde23f77b64fe7f7d490
round[ 6]. [s_row]    e8dab6901477d4653ff7f5e2e747dd4f
round[ 6]. [m_col]    9816ee7400f87f556b2c049c8e5ad036
round[ 6]. [k_sch]    5e390f7df7a69296a7553dc10aa31f6b
round[ 7]. [start]    c62fe109f75eedc3cc79395d84f9cf5d
round[ 7]. [s_box]    b415f8016858552e4bb6124c5f998a4c
round[ 7]. [s_row]    b458124c68b68a014b99f82e5f15554c
round[ 7]. [m_col]    c57e1c159a9bd286f05f4be098c63439
round[ 7]. [k_sch]    14f9701ae35fe28c440adf4d4ea9c026
round[ 8]. [start]    d1876c0f79c4300ab45594add66ff41f
round[ 8]. [s_box]    3e175076b61c04678dfc2295f6a8bfc0
round[ 8]. [s_row]    3e1c22c0b6fcbf768da85067f6170495
round[ 8]. [m_col]    baa03de7a1f9b56ed5512cba5f414d23
round[ 8]. [k_sch]    47438735a41c65b9e016baf4aebf7ad2
round[ 9]. [start]    fde3bad205e5d0d73547964ef1fe37f1
round[ 9]. [s_box]    5411f4b56bd9700e96a0902fa1bb9aa1
round[ 9]. [s_row]    54d990a16ba09ab596bbf40ea111702f
round[ 9]. [m_col]    e9f74eec023020f61bf2ccf2353c21c7
round[ 9]. [k_sch]    549932d1f08557681093ed9cbe2c974e
round[10]. [start]    bd6e7c3df2b5779e0b61216e8b10b689
round[10]. [s_box]    7a9f102789d5f50b2beffd9f3dca4ea7
round[10]. [s_row]    7ad5fda789ef4e272bca100b3d9ff59f
round[10]. [k_sch]    13111d7fe3944a17f307a78b4d2b30c5
round[10]. [output]   69c4e0d86a7b0430d8cdb78070b4c55a
```

```
INVERSE CIPHER (DECRYPT):
round[ 0]. [iinput]   69c4e0d86a7b0430d8cdb78070b4c55a
round[ 0]. [ik_sch]   13111d7fe3944a17f307a78b4d2b30c5
round[ 1]. [istart]   7ad5fda789ef4e272bca100b3d9ff59f
round[ 1]. [is_row]   7a9f102789d5f50b2beffd9f3dca4ea7
round[ 1]. [is_box]   bd6e7c3df2b5779e0b61216e8b10b689
round[ 1]. [ik_sch]   549932d1f08557681093ed9cbe2c974e
round[ 1]. [ik_add]   e9f74eec023020f61bf2ccf2353c21c7
round[ 2]. [istart]   54d990a16ba09ab596bbf40ea111702f
round[ 2]. [is_row]   5411f4b56bd9700e96a0902fa1bb9aa1
round[ 2]. [is_box]   fde3bad205e5d0d73547964ef1fe37f1
round[ 2]. [ik_sch]   47438735a41c65b9e016baf4aebf7ad2
round[ 2]. [ik_add]   baa03de7a1f9b56ed5512cba5f414d23
round[ 3]. [istart]   3e1c22c0b6fcbf768da85067f6170495
round[ 3]. [is_row]   3e175076b61c04678dfc2295f6a8bfc0
round[ 3]. [is_box]   d1876c0f79c4300ab45594add66ff41f
round[ 3]. [ik_sch]   14f9701ae35fe28c440adf4d4ea9c026
round[ 3]. [ik_add]   c57e1c159a9bd286f05f4be098c63439
round[ 4]. [istart]   b458124c68b68a014b99f82e5f15554c
round[ 4]. [is_row]   b415f8016858552e4bb6124c5f998a4c
round[ 4]. [is_box]   c62fe109f75eedc3cc79395d84f9cf5d
round[ 4]. [ik_sch]   5e390f7df7a69296a7553dc10aa31f6b
round[ 4]. [ik_add]   9816ee7400f87f556b2c049c8e5ad036
round[ 5]. [istart]   e8dab6901477d4653ff7f5e2e747dd4f
round[ 5]. [is_row]   e847f56514dadde23f77b64fe7f7d490
round[ 5]. [is_box]   c81677bc9b7ac93b25027992b0261996
round[ 5]. [ik_sch]   3caaa3e8a99f9deb50f3af57adf622aa
round[ 5]. [ik_add]   f4bcd45432e554d075f1d6c51dd03b3c
round[ 6]. [istart]   36339d50f9b539269f2c092dc4406d23
round[ 6]. [is_row]   36400926f9336d2d9fb59d23c42c3950
round[ 6]. [is_box]   247240236966b3fa6ed2753288425b6c
round[ 6]. [ik_sch]   47f7f7bc95353e03f96c32bcfd058dfd
round[ 6]. [ik_add]   6385b79ffc538df997be478e7547d691
round[ 7]. [istart]   2d6d7ef03f33e334093602dd5bfb12c7
round[ 7]. [is_row]   2dfb02343f6d12dd09337ec75b36e3f0
round[ 7]. [is_box]   fa636a2825b339c940668a3157244d17
round[ 7]. [ik_sch]   b6ff744ed2c2c9bf6c590cbf0469bf41
round[ 7]. [ik_add]   4c9c1e66f771f0762c3f868e534df256
round[ 8]. [istart]   3bd92268fc74fb735767cbe0c0590e2d
round[ 8]. [is_row]   3b59cb73fcd90ee05774222dc067fb68
round[ 8]. [is_box]   4915598f55e5d7a0daca94fa1f0a63f7
round[ 8]. [ik_sch]   b692cf0b643dbdf1be9bc5006830b3fe
round[ 8]. [ik_add]   ff87968431d86a51645151fa773ad009
round[ 9]. [istart]   a7be1a6997ad739bd8c9ca451f618b61
round[ 9]. [is_row]   a761ca9b97be8b45d8ad1a611fc97369
round[ 9]. [is_box]   89d810e8855ace682d1843d8cb128fe4
round[ 9]. [ik_sch]   d6aa74fdd2af72fadaa678f1d6ab76fe
round[ 9]. [ik_add]   5f72641557f5bc92f7be3b291db9f91a
round[10]. [istart]   6353e08c0960e104cd70b751bacad0e7
round[10]. [is_row]   63cab7040953d051cd60e0e7ba70e18c
round[10]. [is_box]   00102030405060708090a0b0c0d0e0f0
round[10]. [ik_sch]   000102030405060708090a0b0c0d0e0f
round[10]. [ioutput]  00112233445566778899aabbccddeeff
```

4. Υλοποίηση Αλγορίθμου AES σε Λογισμικό.

4.1. Επιλογή Προγράμματος Υλοποίησης Αλγορίθμου.

Σε αυτήν την ενότητα θα μελετήσουμε βήμα – βήμα τον αλγόριθμο AES σε λογισμικό. Θα αναλύσουμε τις συναρτήσεις, για το πως υλοποιήθηκαν καθώς και τις δυσκολίες που εμφανίστηκαν και πως επιλύθηκαν.Το λογισμικό που βασιστήκαμε για τη δημιουργία του αλγορίθμου είναι το πρόγραμμα Dev-C/C++ της εταιρίας Bloodshed.
Το Dev-C/C++ είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης για τις γλώσσες C και C++. Χρησιμοποιεί τον μεταφραστή GCC και παράγει τριανταδυάμπιτες (32bit) εφαρμογές (Win32) τόσο για τη γραμμή εντολών (console applications) όσο και για το γραφικό περιβάλλον των Windows(GUI_applications).
Ανάμεσα στα πιο εξαιρετικά χαρακτηριστικά αυτού του περιβάλλοντος εργασίας, μπορούμε να αναφέρουμε το περιβάλλον διεπαφής του το οποίο χρησιμοποιεί σύστημα πολλαπλών συστημάτων το οποίο είναι πολύ εύκολο στην κατανόηση και χρήση.
Μεταξύ των εργασιών που μπορούν να χρησιμοποιηθούν με το Dev-C/C++ μπορούμε να αναφέρουμε τον καθαρισμό του κώδικα, την αυτόματη αναγνώριση σύνταξης, τη δημιουργία εκτελέσιμων αρχείων, τη διόρθωση λαθών, και πολλών άλλων πραγμάτων. Επιπλέον, το εργαλείο αυτό απλοποιεί και τελειώνει τις εργασίες με τη βοήθεια εκατοντάδων βιβλιοθηκών που συμπεριλαμβάνονται ήδη, επιτρέπει την εκτύπωση και έχει την ικανότητα να αναζητά και να αντικαθιστά οποιοδήποτε τμήμα του κώδικα.
Προσβάσιμο για κάθε χρήστη, μπορείτε να αποκτήσετε πρόσβαση στο Dev-C++ σε διάφορες γλώσσες.
Το περιβάλλον Dev-C/C++ διατίθεται δωρεάν κάτω από τη γενική άδεια χρήσης της GNU(GNU GENERAL PUBLIC LICENSE).



Σχήμα 4.1: Λογότυπο Dev-C/C++

4.2. Γλώσσα Προγραμματισμού C/ANSI.

Τα βασικά χαρακτηριστικά κάθε συνάρτησης (όνομα, τύπος παραμέτρων, τύπος τιμής επιστροφής) περιλαμβάνονται σε ένα αρχείο που καλείται αρχείο επικεφαλίδα, αλλά η υλοποίηση των συναρτήσεων αυτών είμαι χωριστά σε ένα αρχείο βιβλιοθήκης. Τα ονόματα αυτών των αρχείων-επικεφαλίδων της πρότυπης βιβλιοθήκης έχουν πλέον τυποποιηθεί αλλά όχι και η οργάνωση των βιβλιοθηκών της. Η πρότυπη βιβλιοθήκη παρέχεται συνήθως μαζί με τον μεταγλωττιστή. Όμως επειδή οι μεταγλωττιστές της C παρέχουν συχνά και επιπλέον λειτουργίες που δεν καθορίζονται στο πρότυπο ANSI C, οι πρότυπες βιβλιοθήκες των διαφόρων μεταγλωττιστών εμφανίζουν διαφορές μεταξύ τους.
Η γλώσσα προγραμματισμού C, πριν δημιουργηθεί κάποια πρότυπη μορφή της, δεν παρείχε ενσωματωμένες λειτουργίες όπως λειτουργίες E/E . Με τον καιρό οι κοινότητες χρηστών της C αντάλλασσαν μεταξύ τους ιδέες και υλοποιήσεις γι' αυτό που πλέον καλούμε C πρότυπη βιβλιοθήκη προκειμένου να παρέχεται αυτή η λειτουργικότητα που έλλειπε. Πολλές από αυτές τις ιδέες ενσωματώθηκαν τελικά στον ορισμό του προτύπου της γλώσσα προγραμματισμού C. Το Unix και η γλώσσα προγραμματισμού C δημιουργήθηκαν στα εργαστήρια της AT&T Bell στα τέλη της δεκαετίας 1960 και στις αρχές τις δεκαετίας 1970. Κατά την δεκαετία του 1970 η γλώσσα

προγραμματισμού C έγινε πολύ δημοφιλής. Πολλά πανεπιστήμια και οργανισμοί άρχισαν να δημιουργούν δικές τους παραλλαγές της γλώσσας για τα δικά τους εγχειρήματα. Στις αρχές της δεκαετίας του 1980 άρχισαν να γίνονται εμφανή τα προβλήματα συμβατότητας που παρουσίαζαν οι διάφορες υλοποιήσεις μεταξύ τους. Το 1983 η Αμερικανικός Εθνικός Οργανισμός Τυποποίησης (ANSI) δημιούργησε μια επιτροπή για να δημιουργήσει μια στάνταρ εκδοχή της γλώσσας που έγινε γνωστή σαν "ANSI C". Αυτή η εργασία κορυφώθηκε με την δημιουργία του αποκαλούμενου C89 στάνταρ το 1989. Μέρος του προτύπου που προέκυψε ήταν και ένα σύνολο βιβλιοθηκών που καλείται ANSI C πρότυπη βιβλιοθήκη. Μετέπειτα αναθεωρήσεις του προτύπου της C πρόσθεσαν αρκετά νέα αρχεία επικεφαλίδες στην βιβλιοθήκη. Μετέπειτα αναθεωρήσεις του προτύπου της C πρόσθεσαν αρκετά νέα αρχεία επικεφαλίδες στην βιβλιοθήκη. Η υποστήριξη γι' αυτές τις επεκτάσεις διαφέρει από υλοποίηση σε υλοποίηση.
Τα αρχεία επικεφαλίδες <iso646.h>, <wchar.h>, και <wctype.h> προστέθηκαν με το Κανονιστική τροποποίηση 1 (Normative Amendment 1) (NA1), που αποτελούσε προσθήκη στο πρότυπο της C που επικυρώθηκε το 1995.
Τα αρχεία επικεφαλίδες <complex.h>, <fenv.h>, <inttypes.h>, <stdbool.h>, <stdint.h>, και <tgmath.h> προστέθηκαν με το στάνταρ C99, που ήταν μια αναθεώρηση του προτύπου της C που δημοσιεύθηκε το 1999.
Η ANSI C πρότυπη βιβλιοθήκη αποτελείται από 24 αρχεία επικεφαλίδες της C που μπορούν αν περιληφθούν στο εγχείρημα ενός προγραμματιστή με μια εντολή. Κάθε αρχείο επικεφαλίδα περιέχει δηλώσεις συναρτήσεων, ορισμούς τύπων και μακροεντολές. Το περιεχόμενο τους περιγράφεται παρακάτω.
Σε σύγκριση με άλλες γλώσσες πχ Java η πρότυπη βιβλιοθήκη είναι μικροσκοπική. Παρέχει ένα βασικό σύνολο μαθηματικών συναρτήσεων, χειρισμό αλφαριθμητικών, μετατροπές τύπων και είσοδο / έξοδο (βασισμένο σε κονσόλα). Δεν περιέχει ένα τυποποιημένο σύνολο από "εμπεριέχοντες τύπους" (container types) όπως η Πρότυπη βιβλιοθήκη προτύπων της C++, πόσο μάλλον υποστήριξη για πλήρες σύνολο εργαλείων γραφικής διασύνδεσης χρήστη (GUI), δίκτυα και αφθονία επιπλέον λειτουργιών που περιέχει π.χ. η Java. Το κύριο πλεονέκτημα μιας τόσο μικρής πρότυπης βιβλιοθήκης είναι ότι καθιστά ευκολότερη τη μεταφορά της C σε νέες πλατφόρμες.
Πολλές βιβλιοθήκες έχουν αναπτυχθεί για να παρέχουν την επιπλέον λειτουργικότητα που λείπει από την πρότυπη βιβλιοθήκη της C. Π.χ. το εγχείρημα ανάπτυξης περιβάλλοντος επιφάνειας εργασίας GNOME έχει αναπτύξει το σύνολο εργαλείων γραφικών GTK+ και την GLib, μια βιβλιοθήκη από εμπεριέχοντες δομές δεδομένων. Η ποικιλία των διαθέσιμων βιβλιοθηκών συνεπάγεται την αξία αυτών που έχουν επικρατήσει μέσω μιας διαδικασίας εξονυχιστικής επιλογής. Το μειονέκτημα όμως αυτής της πληθώρας βιβλιοθηκών είναι ότι παρουσιάζονται ασυμβατότητες όταν χρησιμοποιούνται στο ίδιο πρόγραμμα, και δυσχεραίνεται η συνεργασία μεταξύ των προγραμματιστών αφού γνωρίζουν διαφορετικές βιβλιοθήκες.

4.3. Υλοποίηση των Συναρτήσεων στη C.

4.3.1. Συνάρτηση SubBytes.

```
void SubBytes(unsigned char state[4][4])
{ int i,j;
  for (i=0; i<4; i++)
    for (j=0; j<4; j++)
    state[i][j]=Esbox[(state[i][j] & 0xF0)>>4][state[i][j] & 0x0F];
  return;
}
```

Σχήμα 4.2: Κώδικας SubBytes.

Με loop σαρώνουμε τον state πίνακα όπου σε κάθε βήμα παίρνουμε ένα byte και το μοιράζουμε στη μέση. Πχ. εάν το state[1][1]= 53, κόβουμε το 53 σε 5 και 3 και πάμε στον πινάκα του Sbox (βλ. σελ.19, σχήμα 3.5) όπου βρίσκουμε την τιμή που έχει στην $5^{η}$ στήλη και την $3^{η}$ σειρά και την

αντικαθιστούμε. Το μοίρασμα της τιμής το επιτυγχάνουμε με τη λογική πράξη AND. Πχ. state[1][1]= 53, στο δυαδικό σύστημα είναι 0101 0011 εάν κάνουμε την λογική πράξη με το F0 τότε 0101 0011 & 1111 0000 = 0101 0000 που μας κάνει το νούμερο 50 αλλά με μια ολίσθηση τεσσάρων θέσεων δεξιά παίρνουμε το 05 και το τοποθετούμε στη αντίστοιχη στήλη του Sbox μετά κάνουμε πάλι πράξη AND αλλά αυτή τη φορά με το 0F άρα 0101 0011 & 0000 1111= 0000 0011 = 03 όπου αυτό το νούμερο το τοποθετούμε στη σειρά του sbox και έτσι γίνεται η αντικατάσταση των δεδομένων του πίνακα state με αυτών του sbox.

4.3.2. Συνάρτηση ShiftRows.

```
void InvShiftRows(unsigned char state[4][4])
{ unsigned char table;
                    //1H Σειρά ολίσθησης
                    //του πίνακα state

  table=state[1][3];              //2H Σειρά ολίσθησης
  state[1][3]=state[1][2];        //του πίνακα state
  state[1][2]=state[1][1];
  state[1][1]=state[1][0];
  state[1][0]=table;

  table=state[2][3];              //3H Σειρά ολίσθησης
  state[2][3]=state[2][1];        //του πίνακα state
  state[2][1]=table;
  table=state[2][2];
  state[2][2]=state[2][0];
  state[2][0]=table;

  table=state[3][3];               //4H Σειρά ολίσθησης
  state[3][3]=state[3][0];         //του πίνακα state
  state[3][0]=state[3][1];
  state[3][1]=state[3][2];
  state[3][2]=table;
  return;
}
```

Σχήμα 4.3: Κώδικας Shiftrows.

Σε αυτή τη συνάρτηση χρησιμοποιούμε μια προσωρινή μεταβλητή table έτσι ώστε να μας βοηθά στη αριστερή κυκλική ολίσθηση των byte του πινάκα state, συμφώνα με το manual του αλγορίθμου που είδαμε στην 3.3.1.2. ενότητα (σελ. 20).

4.3.3. Συνάρτηση MixColumns.

Εδώ υπήρξε η πρώτη δυσκολία που αντιμετωπίσαμε στον κώδικα όπου η αποτύπωση του manual του αλγορίθμου προγραμματιστικά δεν λειτουργούσε για αυτήν τη συνάρτηση.
Όπως είδαμε στην 3.3.1.3. (σελ. 21) ενότητα ο αλγόριθμος μας δίνει μια μαθηματική φόρμουλα για την υλοποίηση αυτής της συνάρτησης έτσι ώστε να επιτύχουμε το ανακάτεμα των στηλών του state πινάκα.

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{2,c} = \{03\} \bullet s_{0,c} \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

```
void MixColumns(unsigned char state[4][4])
{ int j,tmp;
  for (j=0; j<4; j++)
  {
    tmp=state[0][j]^state[1][j]^state[2][j]^state[3][j];
    state[0][j]=xtime(state[0][j]^state[1][j])^tmp^state[0][j];
    state[1][j]=xtime(state[1][j]^state[2][j])^tmp^state[1][j];
    state[2][j]=xtime(state[2][j]^state[3][j])^tmp^state[2][j];
    state[3][j]=state[0][j]^state[1][j]^state[2][j]^tmp;
  }
  return;
}
```

Σχήμα 4.4: Κώδικας MixColumn μέθοδος Rijndael.

Μετά από έρευνα ανακαλύψαμε ότι υπάρχει και άλλη μαθηματική φόρμουλα με τη μέθοδο Rijndael όπου μας εξυπηρέτησε στην υλοποίηση της συνάρτησης. Η xtime είναι συνάρτηση πολλαπλασιασμού από 00000010 στο GF(28).
Μπορούμε να εφαρμόσουμε xtime ως αριστερή μετατόπιση και XOR με 00011011=1b εάν το υψηλότερο bit είναι ίσον με 1.

```
unsigned char xtime(unsigned char x)
{

 return((x<<1)^(((x>>7 & 1)*0x1b)));

}
```

Σχήμα 4.5: Κώδικας Xtime.

```
Xtime function

Input:
x=(x7,x6.....x0)
Output   y=xtime(x)

y←(x<<1)^FF16
if  x7=1 then
      y←y⊕1B16
end
```

Σχήμα 4.6: Ψευδοκώδικας Xtime.

4.3.4. Συνάρτηση AddRoundKey.

Όπως μελετήσαμε σε παραπάνω ενότητα σε αυτή τη συνάρτηση το κλειδί (πίνακας w) που διαμορφώθηκε στην διαδικασία του Key Expansion ανακατεύεται με τον state πίνακα σύμφωνα με την εξίσωση του αλγορίθμου.
Η συνάρτηση αυτή είναι κοινή κατά τη διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης.

```
void AddRoundKey(unsigned char s[4][4], unsigned char w[Nb*(Nr+1)][4],
int round)
{ int k, c; unsigned char temp[4];
  for (c=0; c<4; c++)
     for (k=0; k<4; k++)
         s[k][c]=s[k][c]^w[round*Nb+c][k];

  return ;
}
```

Σχήμα 4.7: Κώδικας AddRoundKey.

4.3.5. Συνάρτηση KeyExpansion.

Η διαδικασία επέκτασης του κλειδιού (Key Expansion) δέχεται ως είσοδο το αρχικό μυστικό κλειδί (cipher key K), μήκους Nb λέξεων, και παράγει μια ακολουθία κλειδιών. Παράγονται συνολικά από αυτή την διαδικασία Nb(Nr+1) λέξεις καθώς σε κάθε έναν από τους Nr γύρους επεξεργασίας απαιτούνται Nb λέξεις από δεδομένα κλειδιού. Το τελικό key schedule αποτελείται από έναν γραμμικό πίνακα wi με $0 \leq i < Nb(Nr+1)$ που περιέχει λέξεις των τεσσάρων bytes.

```
void KeyExpansion (unsigned char key [4*Nk], unsigned char w[Nb*(Nr+1)][4])
{ int i, j; unsigned char temp [4];
  for (i=0; i<Nk; i++)
    for (j=0; j<Nb; j++)
      w [i][j]=key[4*i+j];
  for (i=Nk; i<Nb*(Nr+1); i++)
  { for (j=0; j<Nb; j++)
      temp[j]=w[i-1][j];
    if (i%Nk==0)
    { RotWord(temp);
      SubWord(temp);
      temp[0]=temp[0]^Rcon[i/Nk -1];
    }
    else if (Nk>6 && i%Nk ==4)           //  για μεγαλύτερα κλειδιά
    {  SubWord(temp);
    }
    for (j=0; j<Nb; j++)
      w[i][j]=w[i-Nk][j]^temp[j];
  }
  return ;
}
```

Σχήμα 4.8: Κώδικας KeyExpansion.

Η διαδικασία SubWord εκτελεί ουσιαστικά την ίδια διαδικασία με την συνάρτηση SubByte μόνο που αυτή την φορά προσδιορίζεται η αντίστοιχη τιμή μιας ολόκληρης λέξης, δηλαδή 4 bytes, μέσω του πίνακα S-box. Είναι, τελικά, σαν να εκτελείται διαδοχικά 4 φορές η συνάρτηση SubByte(). Η

συνάρτηση RotWord() δέχεται ως είσοδο μια λέξη και ολισθαίνει κυκλικά τα bytes που την αποτελούν. Αν για παράδειγμα, δοθεί σαν είσοδος μια λέξη [a0,a1,a2,a3] τότε σαν έξοδο έχουμε την λέξη [a1,a2,a3,a0].
Τέλος, το διάνυσμα Rcon[i] (Round constant word array) περιέχει τις τιμές που δίνονται από την σχέση [xi-1,{00,{00},{00}] με τις τιμές του i να ξεκινούν από 1 και όχι 0.

Rcon [i/Nk] =1 → 0100000000
Rcon [i/Nk] =2 → 0200000000
Rcon [i/Nk] =3 → 0400000000
Rcon [i/Nk] =4 → 0800000000
Rcon [i/Nk] =5 → 1000000000
Rcon [i/Nk] =6 → 2000000000
Rcon [i/Nk] =7 → 4000000000
Rcon [i/Nk] =8 → 8000000000
Rcon [i/Nk] =9 → 1b00000000
Rcon [i/Nk] =10 → 3600000000

```
void SubWord (unsigned char tmp[])
{ int i;
  for (i=0; i<4 ; i++)
    tmp[i]=Esbox[(tmp[i]&0xF0)>>4][(tmp[i]&0x0F)];
  return;
}
```

Σχήμα 4.9: Κώδικας SubWord.

```
void RotWord(unsigned char tmp[])
{ unsigned char t;    //Nb =4
  t=tmp[0];
  tmp[0]=tmp[1];
  tmp[1]=tmp[2];
  tmp[2]=tmp[3];
  tmp[3]=t;
  return;
}
```

Σχήμα 4.10: Κώδικας RotWord.

4.3.6. Συνάρτηση InvSubBytes.

Η διαδικασία για τη δημιουργία της InvSubBytes είναι ακριβώς η ίδια με τη SubBytes όπως είδαμε στην 4.3.1. ενότητα (σελ. 35) όπου η μόνη διαφορά είναι ο πίνακας Sbox που είναι διαφορετικό κατά τη λειτουργία του αλγορίθμου στην αποκρυπτογράφηση.

```
void InvSubBytes(unsigned char state[4][4])
{ int i,j;
  for (i=0; i<4; i++)
    for (j=0; j<4; j++)
      state[i][j]=DEsbox[(state[i][j] & 0xF0)>>4][state[i][j] & 0x0F];
  return;
}
```

Σχήμα 4.11: Κώδικας InvSubBytes.

4.3.7. Συνάρτηση InvShiftRows.

```
void InvShiftRows(unsigned char state[4][4])
{ unsigned char table;

                                  //1Η Σειρά ολίσθησης
                                  //του πίνακα state

  table=state[1][3];
  state[1][3]=state[1][2];        //2Η Σειρά ολίσθησης
  state[1][2]=state[1][1];        //του πίνακα state
  state[1][1]=state[1][0];
  state[1][0]=table;

  table=state[2][3];              //3Η Σειρά ολίσθησης
  state[2][3]=state[2][1];        //του πίνακα state
  state[2][1]=table;
  table=state[2][2];
  state[2][2]=state[2][0];
  state[2][0]=table;

  table=state[3][3];              //4Η Σειρά ολίσθησης
  state[3][3]=state[3][0];        //του πίνακα state
  state[3][0]=state[3][1];
  state[3][1]=state[3][2];
  state[3][2]=table;
  return;
}
```

Σχήμα 4.12: Κώδικας InvShiftRows.

Η διαφορά από την ShiftRows της κρυπτογράφησης είναι ότι τώρα κάνουμε δεξιά κυκλική ολίσθηση των byte σύμφωνα με το σχήμα 3.8. σελ.22.
Έτσι με μια προσωρινή μεταβλητή όπου τη χρησιμοποιούμε για να αποθηκεύσουμε το πρώτο byte που θέλουμε να ολισθήσουμε.

4.3.8. Συνάρτηση InvMixColumns.

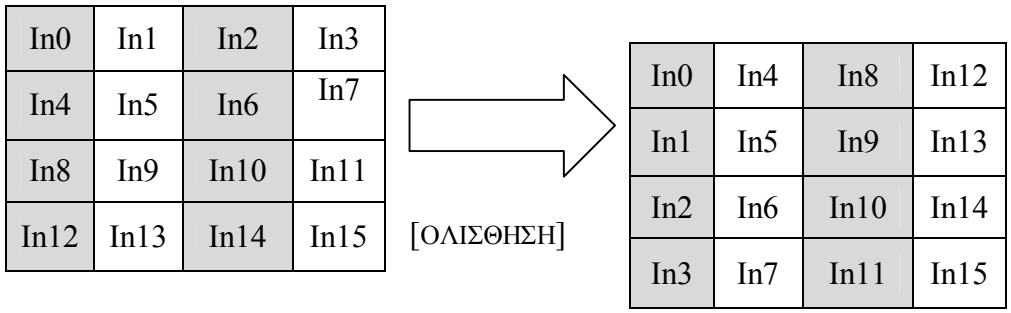
```
void InvMixColumns(unsigned char state[4][4])
{ int j,tmp,tmp2,xe,xo;
  for (j=0; j<4; j++)
  { tmp=state[0][j]^state[1][j]^state[2][j]^state[3][j];
    xe=xtime(state[0][j]^state[2][j]);
    xo=xtime(state[1][j]^state[3][j]);
    tmp2=xtime(xtime(xo^xe))^tmp;
    state[0][j]=xtime(state[0][j]^state[1][j]^xe)^tmp2^state[0][j];
    state[1][j]=xtime(state[1][j]^state[2][j]^xo)^tmp2^state[1][j];
    state[2][j]=xtime(state[2][j]^state[3][j]^xe)^tmp2^state[2][j];
    state[3][j]=state[0][j]^state[1][j]^state[2][j]^tmp;
  }
  return;
}
```

Σχήμα 4.13: Κώδικας InvMixColumns.

Όπως με την MixColumns έτσι και εδώ χρησιμοποιούμε τη μέθοδο Rijndael για να υλοποιήσουμε αυτή τη συνάρτηση.
Έτσι βλέπουμε ένα ανακάτεμα στηλών με την χρήση της λογικής πράξης XOR ⊕ και της συνάρτησης xtime (σχήμα 4.5).

4.4. Υλοποίηση Κρυπτογράφησης.

Ένα πρόβλημα που δημιουργήθηκε και δεν λύθηκε στα υπόψη από την αρχή ήταν κατά την κρυπτογράφηση η σύνταξη του πίνακα εισόδου.



Σχήμα 4.14: Η αρχική μορφή της εισόδου και η μετέπειτα από την ολίσθηση.

Η καταμέτρηση των byte γίνεται κάθετα και όχι οριζόντια όπως ορίζεται από την αρχή ο πίνακας εισόδου. Έτσι πριν την αντιγραφή της εισόδου στην state κατάσταση πρέπει να γίνει μια ολίσθηση των byte για να δρομολογηθεί η διαδικασία της κρυπτογράφησης. Μετά την ολίσθηση αντιγράφουμε τον πίνακα εισόδου 4x4 (plaintext) στον state.
Στη συνέχεια καλούμε τη συνάρτηση KeyExpansion και την AddRoundKey για να διαμορφωθεί το κλειδί. Για να ξεκινήσει η κρυπτογράφηση, αρχικά γίνεται κάλεσμα των συναρτήσεων SubBytes, ShiftRows, MixColumns, AddRoundKey με τη συγκεκριμένη σειρά και γίνονται τόσοι γύροι καλέσματος ανάλογα τα bit αλγορίθμου που χρησιμοποιούμε (Aes 128 bit 10 γύροι, Aes 198 bit 12 γύροι, Aes 256 bit 14 γύροι). Στην προκείμενη περίπτωση έχουμε υλοποιήσει τον Aes 128 bit. Στον τελευταίο γύρο καλούμε την ιδία σειρά τις συναρτήσεις παραλείποντας αυτή τη φορά τη συνάρτηση MixColumns.Αφού τελειώσουν όλοι οι γύροι το τελικό state το αντιγράφουμε στον πίνακα out (ciphertext).

```
void encryption(unsigned char in[4][4], unsigned char intkey[4*Nk],
unsigned char out[4][4])
{ unsigned char state[4][4];
  unsigned char w[Nb*(Nr+1)][4],t; int round=0,i,j;

  t=in[0][1];in[0][1]=in[1][0];in[1][0]=t;   // Ολίσθηση του πίνακα
                                             // εισόδου για να κάνουμε
  t=in[0][2];in[0][2]=in[2][0];in[2][0]=t;   // αντιστοίχηση των bytes
                                             // κατά στήλη.
  t=in[0][3];in[0][3]=in[3][0];in[3][0]=t;
  t=in[2][1];in[2][1]=in[1][2];in[1][2]=t;
  t=in[3][1];in[3][1]=in[1][3];in[1][3]=t;
  t=in[3][2];in[3][2]=in[2][3];in[2][3]=t;

  for (i=0; i<4; i++)
    for (j=0; j<4; j++)      //Αντιγραφή του πίνακα εισόδου στον state.
        state[i][j]=in[i][j];

  KeyExpansion (intkey, w);
  AddRoundKey (state, w, round);

  for (round=1; round<Nr; round++) //Κάλεσμα των συναρτήσεων κρυπτογράφησης
  {                            //τόσες φορές ανάλογα τα bit του αλγορίθμου.
    SubBytes (state);
    ShiftRows (state);
    MixColumns (state);
    AddRoundKey (state, w, round);
    SubBytes (state);             //Ο τελευταίος γύρος του καλέσματος των
    ShiftRows (state);            // συναρτήσεων.

    AddRoundKey (state, w, round);

  for (i=0; i<4; i++)
    for (j=0; j<4; j++)       //Αντιγραφή του state στον out (ciphertext).
      out[i][j]=state[i][j];

  return;
}
```

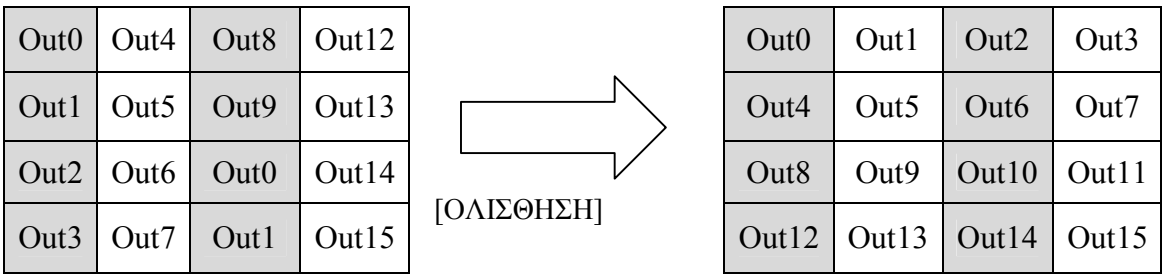
Σχήμα 4.15: Κώδικας Κρυπτογράφησης.

4.5. Υλοποίηση Αποκρυπτογράφησης.

```
void decryption(unsigned char in[4][4], unsigned char intkey[4*Nk],
unsigned char out[4][4])
{ unsigned char state[4][4],t;
  unsigned char dw[Nb*(Nr+1)][4];
  int round=Nr,i,j;
  for (i=0; i<4; i++)
    for (j=0; j<4; j++)
     state[i][j]=in[i][j];
     //Αντιγραφή του πίνακα εισόδου στον state.

   KeyExpansion (intkey, dw);
   AddRoundKey (state, dw, round);
   for (round=Nr-1; round>=1; round--)
                          //Κάλεσμα των συναρτήσεων όπου οι γύροι
   {                     //μετρούνται αντίστροφα και οι συναρτήσεις
     InvShiftRows (state); // είναι αντίστροφες της κρυπτογράφησης.
     InvSubBytes (state);
     AddRoundKey (state, dw, round);
     InvMixColumns (state);
   }

  InvShiftRows (state);        // Ο τελευταίος γύρος καλέσματος.
  InvSubBytes (state);
  AddRoundKey (state, dw, round);

  for (i=0; i<4; i++)
    for (j=0; j<4; j++)
       out[i][j]=state[i][j];
                 //Αντιγραφή του state πίνακα στον out(plaintext).

  t=out[0][1];out[0][1]=out[1][0];out[1][0]=t;
                               //Ολίσθηση και επαναφορά του πίνακα
  t=out[0][2];out[0][2]=out[2][0];out[2][0]=t;
                               //στην αρχική του κατάσταση.
  t=out[0][3];out[0][3]=out[3][0];out[3][0]=t;
  t=out[2][1];out[2][1]=out[1][2];out[1][2]=t;
  t=out[3][1];out[3][1]=out[1][3];out[1][3]=t;
  t=out[3][2];out[3][2]=out[2][3];out[2][3]=t;

  return;

}
```

Σχήμα 4.16: Κώδικας Αποκρυπτογράφησης

Η διαδικασία της αποκρυπτογράφησης είναι σχεδόν η ίδια με της κρυπτογράφησης.
Οι συναρτήσεις καλούνται με τον ίδιο τρόπο και σειρά όπως στην κρυπτογράφηση απλά είναι οι αντίστροφες (InvSuBytes, InvShiftRows, InvMixColumns).
Οι διαφορές τους είναι ότι, οι γύροι που καλούνται οι συναρτήσεις μετρούνται αντίστροφα.
Τέλος, αφού γίνει η αντιγραφή του state στο out(plaintext) τότε κάνουμε την ολίσθηση του πίνακα για να τον φέρουμε στην αρχική του κατάσταση, έτσι ώστε να γίνεται η σάρωση του οριζόντια.



Σχήμα 4.17: Η αρχική μορφή της εξόδου και η μετέπειτα από την ολίσθηση.

4.6. Παράμετροι, Είσοδοι και Έξοδοι.

4.6.1. Κρυπτογράφηση.

Σε αυτήν την ενότητα θα μελετήσουμε τις παραμέτρους που ορίσαμε στον αλγόριθμο κατά τη δημιουργία του κώδικα .Σκοπός των παραμέτρων είναι να γίνει ο σωστός έλεγχος της εισόδου και της εξόδου. Έτσι δημιουργήσαμε 4 παραμέτρους για να βάλουμε σε διάταξη το εκτελέσιμο αρχείο που δημιουργήθηκε από το compile & run του προγράμματος, ένα όρισμα e για encryption η d για decryption, το κλειδί και τέλος το αρχείο που θέλουμε να επεξεργαστούμε.
Πχ. e(παράμετρος 2) 000102030405060708090a0b0c0d0e0f(παράμετρος 3) c:\aa\kar.exe (παράμετρος 4). Η παράμετρος 1 είναι προκαθορισμένη από το πρόγραμμα κατά τη εκτέλεση του κώδικα. Όπως βλέπουμε στο σχήμα 4.15, χρησιμοποιούμε μια εντολή έλεγχου if για να ορίσουμε την διαδικασία μας εάν είναι κρυπτογράφηση (“e”) ή αποκρυπτογράφηση (“d”).
Η παρακάτω συνάρτηση hex2char παίρνει σαν είσοδο το κλειδί όπου από 16δικο το μετατρέπουμε σε χαρακτήρα, γιατί κατά τη λειτουργία Run του προγράμματος η εκτέλεση του γίνεται στο DOS, όπου εκεί διαβάζονται όλα σαν χαρακτήρες. Η επόμενη συνάρτηση star2space είναι ένα τέχνασμα που δημιουργήσαμε έτσι ώστε όταν το μονοπάτι του αρχείου στην παράμετρο 4 έχει κενό τότε να αναγνωρίζεται από τη γραμμή εντολών του DOS.
Οι πίνακες χαρακτήρων path_fn1, path_fn2 χρησιμοποιούνται στην αντιγραφή του μονοπατιού του αρχείου. Στο path_fn2 βάζουμε την κατάληξη DT$, όπου το αρχείο αυτό δημιουργήθηκε κατά την κρυπτογράφηση έτσι ώστε να μας βοηθήσει στην καταμέτρηση των κρυπτογραφημένων χαρακτήρων. Στη συνέχεια αντιγράφουμε το αρχείο που μας υποδεικνύει το path_fn2 στο αρχείο path_fn1 όπου στο τέλος κολλάμε την κατάληξη DTe που μας δηλώνει ότι είναι το πλήρες κρυπτογραφημένο αρχείο. Εάν ανοίξουμε με σημειωματάριο το κρυπτογραφημένο αρχείο θα δούμε στην αρχή μια επιγραφή «AES Encrypted File by Dandoulakis» συν το σύνολο των χαρακτήρων, αυτή η φράση λειτουργεί σαν σφραγίδα έλεγχου κατά τη διαδικασία της αποκρυπτογράφησης. Η ανάγνωση του αρχικού αρχείου έγινε με binary data, έτσι ώστε να αποφύγουμε τη δημιουργία EOF από κάποιο τυχαίο συνδυασμό των δεδομένων κατά την κρυπτογράφηση. Για την ευκολότερη ευκρίνεια των αποτελεσμάτων της κρυπτογράφησης χρησιμοποιήσαμε τη συνάρτηση clock, η οποία κατά τον τερματισμό της διαδικασίας μας επιστρέφει το χρόνο που χρειάστηκε. Τέλος οι συνάρτησης put16charsE και get16charsE μας βοηθάνε για να διαβάσουμε του χαρακτήρες του αρχικού αρχείου και μετά κατά την κρυπτογράφηση να τους αντιγράψουμε σε ένα άλλο αρχείο (αυτό που μας ορίζει το path_fn2 στην περίπτωση μας). Έτσι η συνάρτηση get16charsE έχει σαν δομή να διαβάζει το ciphertext ανά byte προσθέτοντας κατά ένα τον μετρητή nc και τέλος να δημιουργεί συνεχείς ομάδες των 16 byte. Όταν όμως δεν έχει συμπληρωθεί η τελευταία ομάδα ανάγνωσης τότε της προσθέτουμε τυχαία bytes έτσι ώστε να φτάσουμε τα 16. Τέλος η get16charsE αντιγράφει έναν έναν τους

χαρακτήρες που διαβάσαμε. Στις δύο αυτές συναρτήσεις κατά την ανάγνωση και την εγγραφή μας βοήθησαν οι μεταβλητές τύπου δείκτη διεύθυνσης σε αρχείο (file pointer), fr και fw αντίστοιχα.

```
int main(int argc, char *argv[])
{ unsigned char ciphertext[4][4]; char path_fn1[250], path_fn2[250];
  int value=0, i ,j;
  char s1[35], s2[15], s3[15], s4[15], s5[15];
  unsigned char ch1, ch2;
  if (argc==4)
    if (strcmp(argv[1],"e")==0)
     { printf( " ENCRYPTION\n\n");
       hex2char(key,argv[2]);
       star2space(argv[3]);
       if ((fr=fopen(argv[3],"rb"))==NULL)
       { printf("File %s not found!\n",argv[3]); system("PAUSE");
         exit(1);
       }
       strcpy(path_fn1, argv[3]);
       strcpy(path_fn2, path_fn1);
       strcat(path_fn2, "DT$");
       if ((fw=fopen(path_fn2,"wb"))==NULL)
       { printf("Cannot open %s !\n",path_fn1); system("PAUSE");
         exit(2);
      }
       clock_t start = clock();
       while(value==0)
       { value=get16charsE(plaintext);
         encryption (plaintext, key, ciphertext);
         put16charsE(ciphertext);
       }
      printf("Encryption Time: %f \n",  ((double)clock() - start) /
       LOCKS_PER_SEC);
       fclose(fw); fclose(fr);
       if ((fr=fopen(path_fn2,"rb"))==NULL)
       { printf("File Problems 1!\n"); system("PAUSE"); exit(1); }
       strcat(path_fn1, "DTe");
       if ((fw=fopen(path_fn1,"wb"))==NULL)
       { printf("File Problems 2!\n"); system("PAUSE"); exit(2); }
       fprintf(fw,"AES Encrypted File by Dandoulakis: %d:\n",nc);
       fprintf(stdout,"AES Encrypted File by Dandoulakis: %d:\n",nc);
       while (nc>0)
       {
        ch1=getc(fr);
        fwrite(&ch1 , sizeof(ch1), 1 , fw);
        nc--;
       }
       time_t t;
       t=time(NULL);
       printf("Elapsed Time: %f \n", ((double)clock() - start) /
       CLOCKS_PER_SEC);
       printf("Date/Time: %s" , ctime(&t));
       }
```

Σχήμα 4.18: Κώδικας παραμετροποίησης των εισόδων και εξόδων κατά την κρυπτογράφηση.

4.6.2. Αποκρυπτογράφηση.

Για την δημιουργία των παραμέτρων κατά την αποκρυπτογράφηση εργαστήκαμε στο ίδιο μοτίβο όπως με την κρυπτογράφηση. Όπως είδαμε παραπάνω, στο κρυπτογραφημένο αρχείο έχουμε κολλήσει στην πρώτη γραμμή μια σφραγίδα, η οποία κατά την διαδικασία της αποκρυπτογράφησης ελέγχεται, για να ανιχνεύσουμε εάν το αρχείο έχει όντως κρυπτογραφηθεί από το πρόγραμμα μας. Εάν δεν υπάρχει η σφραγίδα μας ειδοποιεί ότι «File is not AES Encrypted by Dandoulakis».

```
..else if (strcmp(argv[1],"d")==0)
     { printf( " DECRYPTION\n\n");
       hex2char(key,argv[2]);
       star2space(argv[3]);
       if ((fr=fopen(argv[3],"rb"))==NULL)
       { printf("File not found!\n"); system("PAUSE");
         exit(1);
       }
       fscanf(fr,"%s%s%s%s%s%s%s%d",s1,s2,s3,s4,s5,&nc);
       strcat(s1," "); strcat(s1, s2); strcat(s1," ");
       strcat(s1, s3); strcat(s1," "); strcat(s1, s4);
       strcat(s1," "); strcat(s1, s5); strcat(s1," ");
       printf("%s%d:\n",s1,nc);
       if (strcmp(s1,"AES Encrypted File by Dandoulakis: ")!=0)
       { printf("File is not AES Encrypted File by Dandoulakis!\n");
       system("PAUSE"); exit(1); }
       strcpy(path_fn1, argv[3]);
       strcat(path_fn1, "DTd");
       if ((fw=fopen(path_fn1,"wb"))==NULL)
       { printf("Cannot open file!\n"); system("PAUSE");
         exit(2);
       }
       clock_t start = clock();
       while(value==0)
       { value=get16charsD(ciphertext);
         decryption (ciphertext, key, plaintext);
         put16charsD(plaintext);
       }
       time_t t;
       t=time(NULL);
       printf("Decryption Time: %f \n", ((double)clock() - start) /
       CLOCKS_PER_SEC);
       printf("Date/Time: %s" , ctime(&t));
       fclose(fw); fclose(fr);
     }
     else
       printf( " Wrong parameter \n\n");
   else
     printf( " Wrong number of parameters \n\n");
  system("PAUSE");
  return 0;
}
```

Σχήμα 4.19: Κώδικας παραμετροποίησης των εισόδων και εξόδων κατά την αποκρυπτογράφηση.

Οι πίνακες χαρακτήρων s1,s2,s3,s4,s5 δημιουργήθηκαν για να αποθηκεύσουν τη σφραγίδα μας «AES Encrypted File by Dandoulakis» η οποία αποτελείται από πέντε λέξεις, γι αυτό και η ύπαρξη τόσων πινάκων. Μετά ενσωματώσαμε όλους τους πινάκες σε έναν (s1) για να γίνει πιο απλή η διαδικασία. Έτσι μετά τον έλεγχο σφραγίδας αντιγράφουμε το μονοπάτι από το

argv [3] στο path_fn1 και του κολλάμε και την κατάληξη DTd, για να ξεχωρίζουμε ότι είναι το αποκρυπτογραφημένο αρχείο. Τέλος κάνουμε την αποκρυπτογράφηση, όπου οι συναρτήσεις get16charsD και put16charsD χρησιμοποιούνται για την ανάγνωση του κρυπτογραφημένου αρχείου και για την εγγραφή των αποκρυπτογραφημένων χαρακτήρων αντίστοιχα. Η get16charsD λειτουργεί παρόμοια με την get16charsE με τη διαφορά όμως ότι διαβάζει το κρυπτογραφημένο αρχείο έως ότου φτάσει τον αριθμό των ψηφιολέξεων (byte) που έχει ορίσει ο μετρητής nc αφαιρώντας έτσι τις τυχαίες ψηφιολέξεις που προστέθηκαν κατά την εγγραφή των χαρακτήρων στη διαδικασία της κρυπτογράφησης για να συμπληρωθεί η ομάδα των 128 bit (128bits block ή αλλιώς 16 bytes). Η εγγραφή των χαρακτήρων κατά την αποκρυπτογράφηση γίνεται με τη συνάρτηση put16charsD ανά byte έως ότου μηδενιστεί ο μετρητής nc.

5. Οδηγίες Χρήσης Λογισμικού.

5.1. Επιλογή Προγράμματος Εξομοίωσης Αλγορίθμου.

Μετά τη δημιουργία του αλγορίθμου στη C/ANSI δεύτερο μέλημα μας ήταν να δημιουργήσουμε το γραφικό περιβάλλον (interface), έτσι ώστε να γίνει πιο απλή η χρήση του αλγορίθμου. Έτσι το λογισμικό που επιλέξαμε ήταν η visual basic.
Η Visual Basic (VB) είναι γλώσσα προγραμματισμού τρίτης γενιάς, οδηγούμενη από συμβάντα (event driven) και έχει ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) από τη Microsoft για το μοντέλο προγραμματισμού COM. Η VB θεωρείται επίσης μία σχετικά εύκολη γλώσσα προγραμματισμού στην εκμάθηση και τη χρησιμοποίηση, λόγω των χαρακτηριστικών της, καθώς έχει Γραφικό Περιβάλλον Χρήστη και συγγένεια με την γλώσσα προγραμματισμού BASIC. Η Visual Basic προέρχεται από τη BASIC και επιτρέπει την ταχεία ανάπτυξη εφαρμογών (RAD) με Γραφικό Περιβάλλον Χρήστη (GUI), πρόσβαση σε βάσεις δεδομένων χρησιμοποιώντας αντικείμενα (Data Access Objects, Remote Data Objects, ή ActiveX Data Objects), και τη δημιουργία στοιχείων ελέγχου ActiveX και αντικειμένων. Οι γλώσσες προγραμματισμού τύπου "scripting", όπως η VBA και VBScript συντακτικά είναι παρόμοιες με τη Visual Basic, αλλά έχουν διαφορετικές επιδόσεις. Ένας προγραμματιστής μπορεί να ολοκληρώσει μια εφαρμογή χρησιμοποιώντας τα στοιχεία που παρέχονται με την Visual Basic. Προγράμματα γραμμένα σε Visual Basic μπορούν, επίσης, να χρησιμοποιήσουν το Windows API, αλλά κάτι τέτοιο απαιτεί δηλώσεις εξωτερικών συναρτήσεων.
Η τελική έκδοση 6 βγήκε το 1998. Η εκτεταμένη υποστήριξη της Microsoft έληξε το Μάρτιο του 2008 και ορίστηκε διάδοχος της η Visual Basic.NET (γνωστή απλά ως Visual Basic).

5.2. Τα Χαρακτηριστικά της Visual Basic.

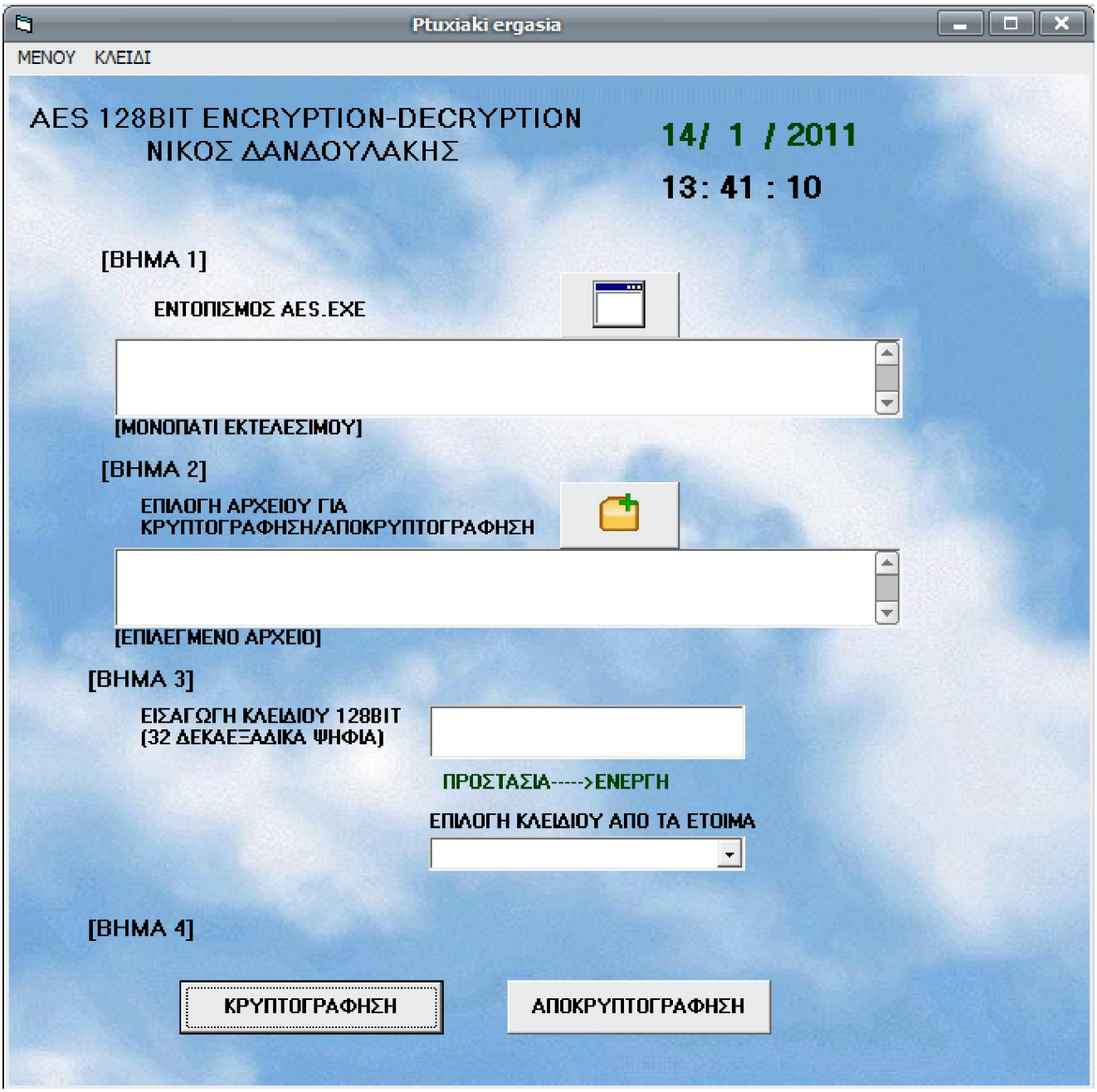
Όπως και η γλώσσα προγραμματισμού BASIC, η Visual Basic έχει σχεδιαστεί για να είναι εύκολη στην εκμάθηση και το χειρισμό. Η γλώσσα δεν επιτρέπει στους προγραμματιστές να δημιουργήσουν μόνο απλές εφαρμογές GUI , αλλά μπορούν, επίσης, να αναπτύξουν πολύπλοκες εφαρμογές. Ο προγραμματισμός σε VB συνίσταται από τον οπτικό συνδυασμό στοιχείων ή ελέγχων σε μια φόρμα, τον προσδιορισμό χαρακτηριστικών και ενεργειών αυτών των στοιχείων και την σύνταξη επιπλέον γραμμών κώδικα για αυξημένη λειτουργικότητα. Καθώς υπάρχουν προεπιλεγμένα χαρακτηριστικά και ενέργειες για τα επιμέρους στοιχεία, μπορεί να δημιουργηθεί ένα απλό πρόγραμμα χωρίς ο προγραμματιστής να γράψει πολλές γραμμές κώδικα. Στις προηγούμενες εκδόσεις υπήρχαν προβλήματα επιδόσεων, αλλά με τους ταχύτερους υπολογιστές και τη μεταγλώττιση εγγενούς κώδικα αυτό παύει να είναι ένα τόσο σημαντικό ζήτημα.
Αν και τα προγράμματα μπορούν να μετατραπούν σε εγγενή εκτελέσιμο κώδικα από την έκδοση 5 και μετά, αυτά εξακολουθούν να απαιτούν την παρουσία των βιβλιοθηκών χρόνου εκτέλεσης (runtime) με μέγεθος περίπου 1 MB. Οι βιβλιοθήκες runtime υπάρχουν στα Windows 2000 και αργότερα, αλλά στις παλαιότερες εκδόσεις των Windows όπως τα 95/98/NT πρέπει να διανέμονται μαζί με το εκτελέσιμο αρχείο.
Οι φόρμες δημιουργούνται χρησιμοποιώντας τεχνικές "σύρε κι άσε" (drag-and-drop). Χρησιμοποιείται ένα εργαλείο για την τοποθέτηση στοιχείων ελέγχου (π.χ. πλαίσια κειμένου, κουμπιά, κλπ.) στη φόρμα (παράθυρο). Τα στοιχεία ελέγχου έχουν χαρακτηριστικά και χειριστές συμβάντων συνδεδεμένους με αυτά. Οι προεπιλεγμένες τιμές παρέχονται όταν δημιουργείται το στοιχείο ελέγχου, αλλά μπορούν να τροποποιηθούν από τον προγραμματιστή. Πολλές τιμές χαρακτηριστικών είναι δυνατό να τροποποιηθούν κατά το χρόνο εκτέλεσης από ενέργειες του χρήστη

ή αλλαγές του περιβάλλοντος, παρέχοντας έτσι μια δυναμική εφαρμογή. Για παράδειγμα, μπορεί να εισαχθεί κώδικας στον χειριστή συμβάντων αλλαγής διαστάσεων της φόρμας, ώστε ένα στοιχείο ελέγχου να παραμένει πάντα στο κέντρο της φόρμας ή να μεγαλώσει ώστε να την γεμίσει, κλπ. Με την προσθήκη κώδικα μέσα σε ένα χειριστή συμβάντων για το πάτημα των πλήκτρων σε ένα πλαίσιο κειμένου, το πρόγραμμα μπορεί αυτόματα να μετατρέψει το εισαγόμενο κείμενο σε κεφαλαία ή πεζά ή ακόμα και να εμποδίσει ορισμένους από τους χαρακτήρες να εμφανιστούν.
Με τη Visual Basic είναι δυνατή η δημιουργία εκτελέσιμων (EXE) αρχείων, στοιχείων ελέγχου ActiveX ή αρχείων DLL, αλλά χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών για τα Windows και τη διασύνδεση συστημάτων βάσεων δεδομένων. Πλαίσια διαλόγου με λιγότερες λειτουργίες μπορούν να χρησιμοποιηθούν για pop-up δυνατότητες. Τα στοιχεία ελέγχου παρέχουν τις βασικές λειτουργίες της εφαρμογής, ενώ οι προγραμματιστές μπορούν να εισαγάγουν επιπλέον λογική μέσα στο κατάλληλο χειριστή γεγονότων. Για παράδειγμα, ένα πτυσσόμενο πλαίσιο θα εμφανίζει αυτόματα μια λίστα που θα επιτρέπει στο χρήστη να επιλέξει οποιοδήποτε στοιχείο. Ένας χειριστής γεγονότων καλείται όταν ένα αντικείμενο είναι επιλεγμένο, και στη συνέχεια μπορεί να εκτελεστεί πρόσθετος κώδικας που δημιουργείται από τον προγραμματιστή για να εκτελεστεί κάποια ενέργεια που βασίζεται στο στοιχείο που έχει επιλεχθεί.
Εναλλακτικά, ένα συστατικό της Visual Basic μπορεί να μην έχει Γραφικό Περιβάλλον Χρήστη, αλλά, αντ' αυτού, να παρέχει αντικείμενα ActiveX σε άλλα προγράμματα μέσω Component Object Model (COM). Αυτό επιτρέπει επεξεργασία στην πλευρά του διακομιστή (server-side processing) ή τη δημιουργία πρόσθετων μορφωμάτων (add-in module).
Η γλώσσα έχει αυτόματη διαχείριση μνήμης με την τεχνική της συλλογής σκουπιδιών (garbage collection) χρησιμοποιώντας υπολογισμό αναφορών και έχει μια μεγάλη βιβλιοθήκη με βοηθητικά αντικείμενα καθώς και βασική αντικειμενοστραφή υποστήριξη. Από τα πιο κοινά στοιχεία που περιλαμβάνονται στο προεπιλεγμένο πρότυπο έργου, ο προγραμματιστής σπάνια χρειάζεται να καθορίσει πρόσθετες βιβλιοθήκες. Αντίθετα με πολλές άλλες γλώσσες προγραμματισμού η Visual Basic γενικά δεν διαχωρίζει τους πεζούς από τους κεφαλαίους χαρακτήρες, αν και θα μετατρέψει τις λέξεις-κλειδιά σε μία τυπική διαμόρφωση. Οι συγκρίσεις συμβολοσειρών διαχωρίζουν τα πεζά από τα κεφαλαία από προεπιλογή, αλλά μπορεί να αλλάξει αυτό, εφόσον το επιθυμείτε.
Ο μεταγλωττιστής της Visual Basic είναι κοινός με τις άλλες γλώσσες του Visual Studio (C, C++), αλλά οι περιορισμοί στον IDE δεν επιτρέπουν τη δημιουργία ορισμένων στόχων (μοντέλα Windows DLL) και σε μοντέλα νημάτων.

Σχήμα 5.1: Λογότυπο Visual Basic.

5.3. Οδηγός Χρήσης.

Σε αυτήν την παράγραφο θα αναλύσουμε με τη σειρά τα βήματα που πρέπει να γίνουν τη σωστή λειτουργία του προγράμματος.



Σχήμα 5.2: Γραφικό περιβάλλον του αλγορίθμου.

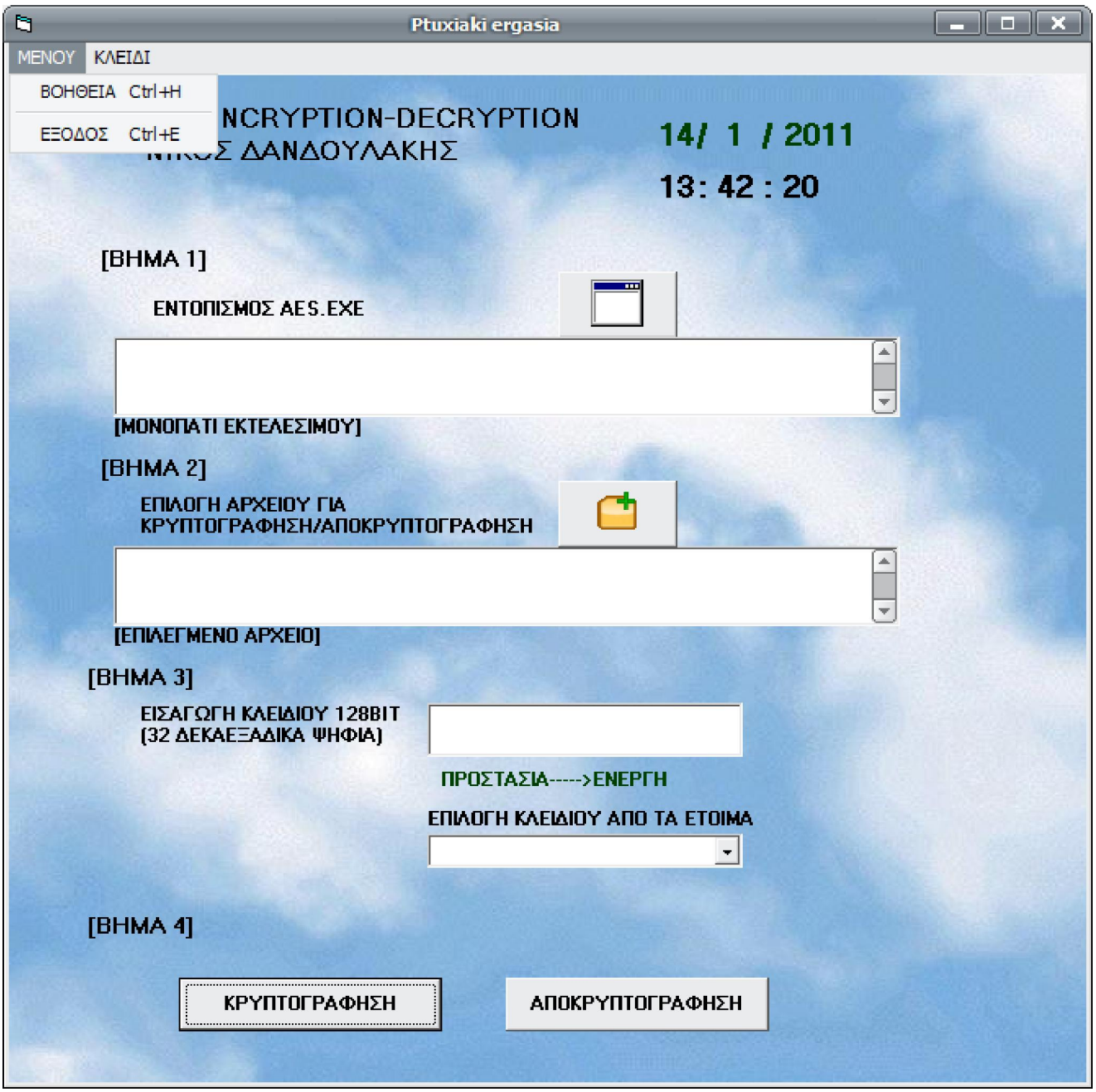
Βήμα 1: Αναζήτηση και επιλογή του εκτελέσιμου αρχείου AES.exe, το οποίο έχει δημιουργηθεί κατά τη εκτέλεση του αλγορίθμου στο πρόγραμμα DEV C/C++.

Βήμα 2: Αναζήτηση και επιλογή αρχείου που θέλουμε να κρυπτογραφήσουμε ή να αποκρυπτογραφήσουμε. Εάν έχουμε ήδη κρυπτογραφήσει κάποιο αρχείο, τότε το ψάχνουμε με κατάληξη DTe, αλλιώς θα μας πετάξει μήνυμα ότι δεν έχει κρυπτογραφηθεί από μας (ενότητα 4.6.2. έλεγχος σφραγίδας). Αποθήκευση του μονοπατιού του αρχείου για να θυμόμαστε πιο κρυπτογραφήσαμε τελευταία (c:\ file.txt).

Βήμα 3: Εισαγωγή μυστικού κλειδιού το οποίο αποτελείται από 32 δεκαεξαδικά ψηφιά. Επιπρόσθετα έχουμε την επιλογή προστασίας, έτσι ώστε να μετατρέψουμε τους αστερίσκους σε αριθμούς και το ανάποδο, για τυχόν αλλαγές του κλειδιού. Η αλλαγή μπορεί να επιτευχτεί με τη συντόμευση ctrl+p.

Βήμα 4: Επιλογή διαδικασίας. Εάν θέλουμε κρυπτογράφηση ή αποκρυπτογράφηση.

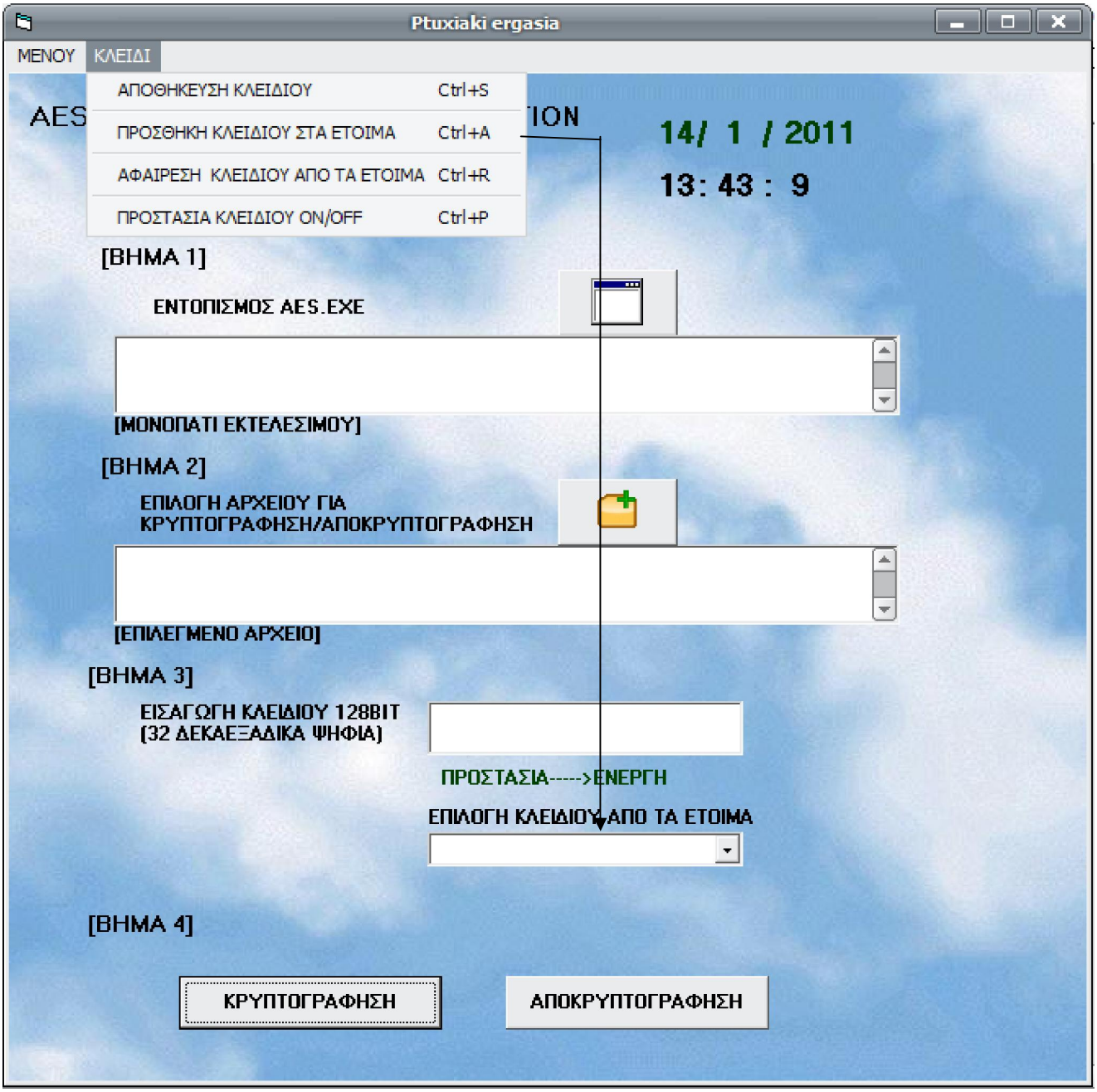
Επίσης έχει δημιουργηθεί μενού για την ευκολότερη καθοδήγηση καθώς στην επιλογή ΒΟΗΘΕΙΑ μας αναφέρει και εκεί τα βήματα με τη σειρά. Η επιλογή ΕΞΟΔΟΣ μας δίνει τη δυνατότητα να βγούμε από το πρόγραμμα με θελημένο σκοπό.



Σχήμα 5.3α: Μενού επιλογών.

Η δεύτερη επιλογή του μενού απαρτίζεται από την «ΑΠΟΘΗΚΕΥΣΗ ΚΛΕΙΔΙΟΥ», όπου ο χρήστης ηθελημένα αποθηκεύει το κλειδί για τυχόν υπενθύμιση. Η εισαγωγή του αποθηκευμένου κλειδιού γίνεται στο αρχείο c:\save_key.txt. Η «ΠΡΟΣΘΗΚΗ ΚΛΕΙΔΙΟΥ ΣΤΑ ΕΤΟΙΜΑ» είναι επιλογή η οποία αποθηκεύει το κλειδί στη λίστα με τα έτοιμα. Η «ΑΦΑΙΡΕΣΗ ΚΛΕΙΔΙΟΥ ΑΠΟ ΤΑ ΕΤΟΙΜΑ» κάνει την ανάποδη διαδικασία όπου αφαιρει το ήδη αποθηκευμένο κλειδί από τη λίστα με

τα έτοιμα. Τέλος η «ΠΡΟΣΤΑΣΙΑ ΚΛΕΙΔΙΟΥ ON/OFF» μετατρέπει τα ψηφιά του κλειδιού από αστερίσκους σε αριθμούς και το ανάποδο.



Σχήμα 5.3β: Μενού επιλογών.

6. Συμπεράσματα.

Κατά τη υλοποίηση του αλγορίθμου ο τομέας που μας δυσκόλεψε περισσότερο ήταν η κατανόηση του, καθώς είχε πολλές ιδιαιτερότητες με τον τρόπο λειτουργίας του.
Οι ιδιαιτερότητες αυτές επισημαίνονται κατά την επεξήγηση των μαθηματικών εξισώσεων που εξελίσσονται στο Galois field (GF). Η πρώτη μεγάλη δυσκολία που εμφανίστηκε λόγο αυτών των εξισώσεων ήταν στην συνάρτηση MixColumns και στην InvMixColumns, όπου έπρεπε να χρησιμοποιηθεί η υποσυνάρτηση xtime και να οδηγηθεί σε αλληλένδετες εξισώσεις οι οποίες έχουν δημιουργηθεί με τη μέθοδο Rijndael .
Η δημιουργία του αλγορίθμου στη C/ANSI είχε αρκετές δυσκολίες, καθώς η αναζήτηση κάποιων βοηθημάτων ήταν επίπονη, για το λόγο του ότι η γλώσσα έχει παραγκωνιστεί με τον καιρό από τη C++ και από τη C#.
Η χρήση όμως της C ήταν ικανοποιητική καθώς η ανταπόκριση ήταν πολύ γρήγορη κατά τη διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης. Στο παρακάτω σχήμα 6.1 παρατάσσονται τα δεδομένα μαζί με το χρόνο που απαιτείται για να ολοκληρωθεί η λειτουργία.
Βλέπουμε ότι θέλουμε περισσότερο χρόνο κατά τη λειτουργία της αποκρυπτογράφησης, για το λόγο ότι υπάρχει διαφορά μεταξύ των συναρτήσεων put16charsD-put16charsE get16charsD- get16charsE όπως είδαμε σε παραπάνω ενότητα. Οι συναρτήσεις αυτές διαβάζουν και τοποθετούν τους χαρακτήρες της διαδικασίας (κρυπτογράφηση-E, αποκρυπτογράφηση-D).
Αυτό που συμπεραίνουμε είναι ότι εάν λειτουργούμε τον αλγόριθμο οι χρόνοι ανταπόκρισης κατά την κρυπτογράφηση/αποκρυπτογράφηση, επηρεάζονται σημαντικά εάν οι πόροι της μνήμης Ram είναι κατειλημμένοι από άλλες εφαρμογές. Γι αυτό οι τιμές που βλέπουμε στο σχήμα είναι ενδεικτικές.

Μέγεθος Αρχείου (MegaByte-MB)	**Χρόνος κρυπτογράφησης (sec)**	**Χρόνος κρυπτογράφησης + Χρόνος αντιγραφής (sec)**	**ΧΡΟΝΟΣ αποκρυπτογράφησης (sec)**
1	0.7	0.95	0.85
4	3	4	3.5
100	61	84	74
388	236	324	281
716	439	607	490
1370	845	1108	959

Σχήμα 6.1: Χρόνος ολοκλήρωσης διαδικασίας ανά μέγεθος αρχείου.

Επίσης αυτό που παρατηρούμε είναι ότι κατά τη διαδικασία της αποκρυπτογράφησης εάν βάλουμε λάθος κλειδί θα εκτελεστεί κανονικά, αλλά το αρχείο δεν θα έχει την αρχική του μορφή και θα πετάξει μήνυμα σφάλματος.
Τέλος η επιλογή του προγράμματος Visual Basic για να δημιουργηθεί το γραφικό περιβάλλον του αλγορίθμου ήταν απόλυτα επιτυχής και επαρκής, επειδή σαν γλώσσα είναι εύκολη καθώς η εκμάθηση της έγινε από την αρχή, λόγο του ότι δεν ήταν γνωστή. Σαν αναπτυξιακό περιβάλλον είναι πλήρης και έτσι μας βοήθησε να δημιουργήσουμε ένα interface καλά οργανωμένο και προσιτό στο χρηστή κατά την λειτουργία του. Επίσης το interface δεν επηρέασε καθόλου τον αλγόριθμο στη C ούτε στον χρόνο ολοκλήρωσης της διαδικασίας ούτε και στις μεταβλητές εισόδου και εξόδου. Έτσι μπορούμε να πούμε ότι δημιουργήσαμε ένα πλήρης δέσιμο μεταξύ των δυο γλωσσών προγραμματισμού.

Βιβλιογραφία:

Βιβλία:

- "Cryptography and Network Security" Fourth Edition , William Stallings, εκδόσεις Pearson Prentice Hall, 2006.
- "Applied Cryptography", Bruce Schneier, εκδόσεις John Wiley & Sons, 1996.
- "Ασφάλεια Δικτύων Υπολογιστών", Πομπόρτσης Α., Παπαδημητρίου Γ., εκδόσεις Τζιόλα, 2003.

Ιστότοποι:

- *http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf*
- *http://students.ceid.upatras.gr/~mprokala/techarticles/catindex.htm*
  Δημοσίευμα από Μπροκαλάκη Ανδρέα.
- *http://el.wikipedia.org/wiki*

Περιεχόμενα

Περίληψη

Η επιθυμία προστασίας του περιεχομένου μηνυμάτων οδήγησε στην επινόηση και χρήση κρυπτογραφικών τεχνικών και συστημάτων τα οποία επιτρέπουν το μετασχηματισμό μηνυμάτων ή δεδομένων κατά τέτοιον τρόπο ώστε να είναι αδύνατη η υποκλοπή του περιεχομένου τους κατά τη μετάδοσή ή αποθήκευσή τους και, βεβαίως, την αντιστροφή του μετασχηματισμού. Η διαδικασία μετασχηματισμού καλείται κρυπτογράφηση και η αντίστροφή της αποκρυπτογράφηση. Η εργασία αυτή αναφέρεται στην ηλεκτρονική κρυπτογράφηση/αποκρυπτογράφηση όπου είναι μια διαδικασία βασισμένη στη λογική της κωδικοποίησης ομάδων δεδομένων με κάποιο μυστικό κλειδί χρησιμοποιώντας τον πρότυπο αλγόριθμο AES. Πέρα από την ιστορική ανάλυση βασικών εννοιών και την σύντομη ιστορική ανάδρομη, η υλοποίηση του αλγορίθμου AES μέσω λογισμικού αποτελεί τον βασικό στόχο αυτής της εργασίας. Ακολουθεί θεωρητική μελέτη και κατανόηση της λειτουργίας του AES μελετώντας όλη του τη δομή. Στη συνέχεια θα ταξιδέψουμε στο χώρο του προγραμματισμού, όπου θα υλοποιήσουμε τον αλγόριθμο χρησιμοποιώντας την γλώσσα προγραμματισμού C και την διεπαφή χρησιμοποιώντας Visual Basic. Τέλος, θα εξαχθούν συμπεράσματα σχετικά με την συγκεκριμένη υλοποίηση του AES και θα πραγματοποιηθούν χρονομετρήσεις για διάφορα μεγέθη αρχείων.

Abstract

The wish of protection of content of messages led to the invention and use of cryptographic techniques and systems which allow the transformation of messages or data of so that it is impossible to intercept the content at the transmission or their storage and, of course, the inverse transformation. The transformation process is called encryption and its opposite decryption. This work is reported in the electronic encryption / decryption where it is a process based on the logic of coding blocks of data with some secret key using the algorithm AES. Beyond the historical analysis of basic significances and historical retrograde, the primary target of this work was the implementation of the AES algorithm via software. Theoretical studding and comprehension of AES structure was fulfilled. Then we will travel in the space of planning, where we will implement the algorithm using the programming languages: C for the code and Visual Basic for the interface. In the end, there will be conclusions about this implementation of AES and some timing measurements for several file sizes.