ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Σχολή Θετικών Επιστημών
Τμήμα Μαθηματικών

Ανάλυση και ανάπτυξη κρυπτογραφικών αλγορίθμων υλοποιήσιμων σε φορητές τηλεπικοινωνιακές συσκευές με περιορισμένους υπολογιστικούς πόρους

Analysis and development of cryptographic algorithms implementable in portable telecommunication devices with limited computational resources

Ξηρουχάκης Παντελεήμων

Διπλωματική Εργασία για την απόκτηση του Μεταπτυχιακού Τίτλου Σπουδών στα Εφαρμοσμένα Μαθηματικά

Επιβλέπων καθηγητής: Νικόλαος Μπάρδης

ΑΘΗΝΑ
2019

Περίληψη

Η παρούσα διπλωματική εργασία μελετά τις ειδικές απαιτήσεις για την κρυπτογραφική προστασία των φωνητικών επικοινωνιών που πραγματοποιούνται με τη χρήση φορητών συσκευών με περιορισμένους υπολογιστικούς πόρους και παρέχονται είτε από το τηλεπικοινωνιακό δίκτυο των παρόχων κινητής τηλεφωνίας, ή από το Διαδίκτυο μέσω εφαρμογών VOIP. Χαρακτηριστικό παράδειγμα τέτοιων συσκευών είναι τα έξυπνα τηλέφωνα.

Αρχικά, γίνεται εισαγωγή στην έννοια της «Κρυπτολογίας», μελετάται το μαθηματικό υπόβαθρο που απαιτείται για την θεμελίωση και περαιτέρω ανάλυση των κρυπτογραφικών αλγορίθμων και γίνεται ιστορική αναδρομή στα είδη τηλεπικοινωνιών με έμφαση στις σύγχρονες μεθόδους. Στη συνέχεια αναφέρονται τα βασικά στοιχεία των κρυπτογραφικών αλγορίθμων και αναλύονται οι κυριότερες κατηγορίες τους.

Ακολούθως μελετώνται οι βασικοί αλγόριθμοι κρυπτογράφησης φωνής στις τηλεπικοινωνίες και παρουσιάζεται η ισχύς τους ως προς τις γνωστές κρυπταναλυτικές τεχνικές που έχουν εφαρμοστεί εναντίον τους.

Απώτερος στόχος της παρούσας διπλωματικής εργασίας είναι η σχεδίαση νέων αλγορίθμων κρυπτογράφησης, των οποίων το υπολογιστικό κόστος θα ενθαρρύνει την ενσωμάτωσή τους σε φορητές συσκευές υψηλής ασφάλειας· ικανές να χρησιμοποιηθούν σε ευρεία κλίμακα από προσωπικό που σήμερα είτε δεν διαθέτει ασφαλή κανάλια επικοινωνιών, είτε βασίζεται μόνο σε μεθόδους ασφαλείας που ενσωματώνουν στα προϊόντα τους οι εξυπηρετητές δικτύου ή τρίτοι πάροχοι.

ΑΘΗΝΑ – ΝΟΕΜΒΡΙΟΣ 2019
ΞΗΡΟΥΧΑΚΗΣ ΠΑΝΤΕΛΕΗΜΩΝ

Abstract

In this thesis, we study the requirements for the cryptographic protection of voice communications that take place using portable telecommunication devices with limited computational resources, provided either by telecommunication providers or by Internet via VOIP applications. A typical example of such a device is smartphones.

Initially, we introduce the reader in the term of "Cryptology", we study the mathematical background required for the foundation and further analysis of the cryptographic algorithms and we quote the chronology of telecommunication, emphasising in the recent progress. Also, we refer to the rudiments of cryptographic algorithms, by analyzing their basic categories.

Following, we study the most important cryptographic algorithms concerning the protection of voice in mobile telecommunications, presenting their strength against known cryptanalytic techniques that have been used against them.

The ultimate goal of this thesis is the development of new cryptographic algorithms, whom computational cost will encourage the integration of them in high security portable devices, in order to be widely usable from people that now either not have secure communication channels or are based on security methods that are embedded to the products by the service or other providers.

ATHENS - NOVEMBER 2019
XIROUCHAKIS PANTELEIMON

Η παρούσα Διπλωματική Εργασία εκπονήθηκε για τις ανάγκες απόκτησης του Μεταπτυχιακού Τίτλου Σπουδών στα Μαθηματικά (Ειδίκευση Εφαρμοσμένων Μαθηματικών) του Προγράμματος Μεταπτυχιακών Σπουδών, του Τμήματος Μαθηματικών, της Σχολής Θετικών Επιστημών, του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών.

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ

ΕΠΙΒΛΕΠΩΝ

ΜΠΑΡΔΗΣ ΝΙΚΟΛΑΟΣ - Αναπληρωτής Καθηγητής ΣΣΕ

ΜΕΛΗ

ΔΡΑΚΟΠΟΥΛΟΣ ΜΙΧΑΗΛ - Επίκουρος Καθηγητής τμήματος Μαθηματικών ΕΚΠΑ
ΚΟΤΤΑ–ΑΘΑΝΑΣΙΑΔΟΥ ΕΥΑΓΓΕΛΙΑ - Επίκουρη Καθηγήτρια τμήματος Μαθηματικών ΕΚΠΑ

Ευχαριστίες

Όντας στην ευχάριστη θέση της ολοκλήρωσης των μεταπτυχιακών μου σπουδών, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κύριο Μπάρδη Νικόλαο για την εμπιστοσύνη που επέδειξε στο πρόσωπό μου αλλά και τη καθοδήγηση που μού παρείχε κατά τη διάρκεια της εκπόνησης της διπλωματικής εργασίας. Επίσης, θα ήθελα να ευχαριστήσω την κυρία Κόττα-Αθανασιάδου Ευαγγελία για τη συμμετοχή της στην τριμελή επιτροπή καθώς και τον κύριο Δρακόπουλο Μιχαήλ τόσο για τη συμμετοχή του στην τριμελή επιτροπή, όσο και για την ευχάριστη και εποικοδομητική συνεργασία που είχαμε καθ΄όλη τη διάρκεια των σπουδών μου· προπτυχιακών και μεταπτυχιακών.

Σε αυτό το ταξίδι πέραν των σπουδαίων γνώσεων που αποκομίσαμε, περάσαμε ωραίες στιγμές, μοιραστήκαμε προβληματισμούς και ήρθαμε κοντά με τους συμφοιτητές μου, τους οποίους θα ήθελα μαζικά να ευχαριστήσω. Επίσης, θα ήθελα να ευχαριστήσω τους φίλους μου - ιδιαιτέρως αυτούς από τα προπτυχιακά μου έτη - οι οποίοι με ενθάρρυναν συνεχώς μέχρι να φτάσω στο τέλος αυτής της απαιτητικής διαδρομής.

Για το τέλος άφησα την οικογένειά μου, την οποία ευχαριστώ απεριόριστα για την αμέριστη συμπαράσταση και κατανόηση που επέδειξε όλα αυτά τα χρόνια· από το πρώτο έτος στο Πανεπιστήμιο μέχρι και την τελευταία πρόταση του παρόντος πονήματος.

Λίγο πριν πέσουν οι τίτλοι τέλους του δεύτερου ταξιδιού μου στην τριτοβάθμια εκπαίδευση, θα ήθελα να εκφράσω την ιδιαίτερη χαρά μου που ολοκλήρωσα και τους δύο κύκλους σπουδών στο Τμήμα Μαθηματικών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών, του αρχαιότερου Ανώτατου Εκπαιδευτικού Ιδρύματος της χώρας.

Στην οικογένειά μου.

Κατάλογος εικόνων και σχημάτων

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή στην Κρυπτογραφία

Η *κρυπτολογία*, ως ο κλάδος που ασχολείται με ζητήματα ασφάλειας των επικοινωνιών, έχει μία πλούσια ιστορία χιλιάδων ετών, όσων δηλαδή και οι διάφοροι τρόποι επικοινωνίας: Ξεκινάει με την εμφάνιση πρώιμων μορφών κρυπτογράφησης, που βασίζονται σε απλές αντικαταστάσεις των συμβόλων του μεταδιδόμενου μηνύματος, και συνεχίζεται μέχρι σήμερα που έχει αναπτυχθεί μια πληθώρα πολύπλοκων αλγορίθμων κρυπτογράφησης αλλά και σύνθετων πρωτοκόλλων που στηρίζονται στην απόκρυψη πληροφορίας. Ιδιαίτερα κατά τις τελευταίες δεκαετίες, η ραγδαία άνθηση των τηλεπικοινωνιών και του Διαδικτύου έχει καταστήσει τη διασφάλιση του απορρήτου των επικοινωνιών απόλυτη ανάγκη, φέρνοντας την κρυπτολογία στο επίκεντρο των τεχνολογικών εξελίξεων. Η κρυπτολογία αποτελεί πλέον αντικείμενο έντονης ερευνητικής δραστηριότητας, η οποία την έχει μετατρέψει σε επιστήμη, με αυστηρούς ορισμούς και αποδείξεις.

Τυπικά με τον όρο *κρυπτολογία* αναφερόμαστε τόσο στην κρυπτογραφία όσο και στην κρυπτανάλυση: η *κρυπτογραφία* ασχολείται με τον σχεδιασμό κρυπτοσυστημάτων, ενώ η *κρυπτανάλυση* μελετά τις αδυναμίες τους. Καταχρηστικά, ο όρος *κρυπτολογία* έχει «απορροφηθεί» από τον όρο *κρυπτογραφία*.

1.1 Ιστορική Αναδρομή

Από την Αρχαιότητα έως την Αναγέννηση

Στην αρχαία Ελλάδα, οι έφοροι της Σπάρτης επικοινωνούσαν με τους στρατηγούς χρησιμοποιώντας μακριές και στενές κορδέλες τις οποίες τύλιγαν γύρω από μια σκυτάλη και έπειτα έγραφαν το μήνυμα κατά μήκος της. Για να διαβάσει κάποιος το μήνυμα, έπρεπε να έχει μια παρόμοια σκυτάλη με αυτή που είχε χρησιμοποιηθεί από τον δημιουργό του ώστε να τυλίξει την κορδέλα γύρω από αυτήν με τον ίδιο τρόπο. Έτσι, το πρόβλημα της ασφάλειας των πληροφοριών μεταφερόταν από το ίδιο το μήνυμα, στο κλειδί με το οποίο κωδικοποιούταν το μήνυμα και το οποίο είχε μέγεθος κατά πολύ μικρότερο του μηνύματος. Αυτό το σύστημα αμφίδρομης κρυπτογράφησης είναι ένα κλασικό σύστημα με ένα κλειδί (τη σκυτάλη) και αποτελεί το πρώτο γνωστό κλασικό σύστημα κρυπτογράφησης της ανθρωπότητας.

Ο Ιούλιος Καίσαρας επικοινωνούσε με τους συνεργάτες του αντικαθιστώντας κάθε γράμμα με ένα άλλο, το οποίο προέκυπτε με ολίσθηση κατά k βήματα στο αλφάβητο. Αυτό είναι ένα από τα πιο απλά και εύκολα, αλλά ταυτοχρόνως ανασφαλή κρυπτοσυστήματα που έχουν προταθεί.

Οι Βενετοί ήταν οι πρώτοι που χρησιμοποίησαν κρυπτογραφία συστηματικά, από το 13ο αιώνα, για διπλωματική αλληλογραφία. Οι πρώτες γνωστές δημοσιεύσεις (στα λατινικά) περί κρυπτογραφίας ανάγονται στο 1500 μ.Χ. και το 1518 μ.Χ. από τον αβά Ιωάννη Τριθέμιο, ενώ αργότερα δημοσιεύτηκε το «Περί κρυπτικών συμβόλων και γραμμάτων» από τον J. B. Porta (1538 - 1615), Ιταλό φυσικό και μαθηματικό. Έκτοτε η κρυπτογραφία έγινε αντικείμενο ιδιαίτερου ενδιαφέροντος. Οι αλχημιστές χρησιμοποιούσαν σύμβολα για να κρυπτογραφήσουν τους τύπους τους, ενώ και πολλοί φιλόσοφοι ενδιαφέρθηκαν για την κρυπτογραφία: Ο Sir Francis Bacon (1561 - 1626) επινόησε ένα σύστημα κρυπτογράφησης όπου κάθε γράμμα αντικαθίσταται με μια λέξη πέντε γραμμάτων, ενώ ο Leonardo Da Vinci (1452 - 1519) χρησιμοποιούσε μια μέθοδο κρυπτογράφησης με καθρέπτη.

19ος και 20ος αιώνας

Ο Edgar Allan Poe (1809 - 1849) στο διήγημά του «Ο χρυσός σκαραβαίος» που δημοσιεύθηκε το 1843, εξηγεί τις βασικές αρχές παραβίασης των κρυπταλγορίθμων και υποστηρίζει την άποψη ότι ο ανθρώπινος νους μπορεί να σπάσει οποιοδήποτε κρυπτογραφημένο κείμενο μπορεί να επινοήσει η ανθρώπινη ευρηματικότητα. Ακόμη περιγράφει ένα σύστημα με το οποίο κάθε κρυπτογραφημένο κείμενο που προέρχεται από μια ευρωπαϊκή γλώσσα μπορεί να αποκρυπτογραφηθεί - αν έχει κρυπτογραφηθεί με αντικατάσταση - μετρώντας τη συχνότητα των γραμμάτων της γλώσσας, τεχνική που πρώτοι χρησιμοποίησαν οι Άραβες.

Ίσως ένα από τα διασημότερα κρυπτογραφήματα ήταν το σημείωμα του Zimmerman (The Zimmerman Note), το οποίο ώθησε τις ΗΠΑ στον πρώτο παγκόσμιο πόλεμο. Όταν το κρυπτογράφημα αποκρυπτογραφήθηκε το 1917, οι Αμερικανοί έμαθαν ότι η Γερμανία είχε προσπαθήσει να πείσει το Μεξικό να μπει στον πόλεμο με το μέρος της, υποσχόμενη παραχωρήσεις εδαφών των ΗΠΑ στο Μεξικό.

Τον ίδιο περίπου καιρό, ο Gilbert S. Vernam της AT&T ανέπτυξε τον πρώτο πραγματικά άθραυστο κώδικα που ονομάστηκε προς τιμήν του κρυπτόγραμμα Vernam (The Vernam Cipher). Μια ξεχωριστή ιδιότητα αυτού του κώδικα ήταν η απαίτηση για ένα κλειδί με μήκος τουλάχιστον όσο και το μήκος του μηνύματος που έπρεπε να μεταδοθεί και το οποίο δεν θα επαναχρησιμοποιείτο για την αποστολή άλλου μηνύματος. Η κρυπτογράφηση Vernam είναι γνωστή επίσης και ως κρυπτογράφηση με μπλοκάκι μιας χρήσης ή one-time-pad από την πρακτική της προμήθειας κατασκόπων με το κείμενο-κλειδί γραμμένο σε ένα μπλοκάκι του οποίου κάθε κομμάτι χρησιμοποιείται μια φορά και έπειτα καταστρέφεται. Ωστόσο, η ανακάλυψη του συστήματος αυτού δεν εκτιμήθηκε ιδιαίτερα εκείνη την εποχή, κυρίως επειδή δεν είχε αποδειχτεί ακόμη ότι ο κρυπταλγόριθμος είναι άθραυστος, αλλά και επειδή η απαίτηση για πολλά και μεγάλα κλειδιά το καθιστούσαν μη πρακτικό για γενική χρήση. Αξίζει πάντως να αναφερθεί ότι το 1967, όταν ο στρατός της Βολιβίας συνέλαβε και εκτέλεσε τον Che Guevara, λέγεται ότι βρήκαν στην κατοχή του ένα χαρτί που έδειχνε πως προετοίμαζε ένα μήνυμα για αποστολή στον Κουβανό πρόεδρο Fidel Castro. Σε αυτό, ο Che Guevara χρησιμοποιούσε τον άσπαστο κώδικα του Vernam!

Εξαιτίας των μη πρακτικών απαιτήσεων της κρυπτογράφησης Vernam , άλλες (πιο αδύναμες) μέθοδοι συνέχισαν να χρησιμοποιούνται ευρέως. Έτσι, κατά το δεύτερο παγκόσμιο πόλεμο, οι Σύμμαχοι ήταν σε θέση να αποκρυπτογραφούν τα περισσότερα από τα μυστικά μηνύματα που στέλνονταν από τους Γερμανούς. Η εγγενής δυσκολία του σπασίματος των ολοένα και πιο περίπλοκων κρυπτογραφικών μεθόδων ήταν μάλιστα ένας από τους παράγοντες που ώθησε την ανάπτυξη των ηλεκτρονικών υπολογιστών.

Μηχανή Enigma και Alan Turing

Η περίφημη Μηχανή - Αίνιγμα (Enigma) που χρησιμοποιήθηκε από τους Γερμανούς κατά το δεύτερο παγκόσμιο πόλεμο για την κρυπτογράφηση των ραδιοτηλεπικοινωνιών ήταν ίσως το πλέον εξελιγμένο κρυπτοσύστημα της εποχής και πυροδότησε μια από τις πιο έντονες προσπάθειες αποκρυπτογράφησης στην ιστορία. Ο κώδικας Αίνιγμα θύμιζε έναν παλιότερο κώδικα τύπου πολυαλφαβητικής υποκατάστασης αλλά ήταν πολύ πιο πολύπλοκος. Μια βασική ιδιότητα της μηχανής αυτής ήταν η αυτό-αντιστροφή: εάν το κωδικοποιημένο κείμενο δινόταν ως είσοδος στη μηχανή, τότε η έξοδος θα ήταν το αρχικό μήνυμα (αν φυσικά η μηχανή είχε την ίδια αρχική κατάσταση με τη μηχανή που είχε κάνει την κωδικοποίηση). Παρόλο που αυτό αποτελούσε τρομερή ευκολία για τους χειριστές της μηχανής, αποδείχτηκε ότι ήταν και μεγάλη αδυναμία της Μηχανής - Αίνιγμα. Πριν τον πόλεμο, η γαλλική αντικατασκοπεία είχε αποκτήσει αντίγραφα των εντολών της Μηχανής - Αίνιγμα και έδωσε την πληροφορία αυτή στους Πολωνούς που υπέκλεπταν και ανέλυαν τις γερμανικές ράδιοεπικοινωνίες. Με τη βοήθεια των εντολών αυτών, οι Πολωνοί κρυπταναλυτές μπόρεσαν να συμπεράνουν τη συνδεσμολογία - καλωδίωση της μηχανής, οπότε κατέστη εφικτό να διαβάζονται τα κρυπτογραφημένα κείμενα, με την προϋπόθεση να είναι γνωστή η αρχική κατάσταση της μηχανής. Παρόλο που οι Βρετανοί πήραν τις πληροφορίες αυτές από τους Πολωνούς, είχαν μικρή αξία γι' αυτούς επειδή οι Γερμανοί έκαναν κάποιες τροποποιήσεις στη μηχανή πριν τον πόλεμο. Έτσι, οι Βρετανοί συγκέντρωσαν μια ομάδα κρυπταναλυτών και μαθηματικών, με επικεφαλής τον Alan Turing, και αξιοποιώντας τις πληροφορίες των Πολωνών βάσισαν τις προσπάθειές τους στη λεγόμενη μέθοδο πιθανής λέξης. Η μέθοδος αυτή βασίζεται στο γεγονός ότι σε κάποιες περιπτώσεις μια συγκεκριμένη ακολουθία συμβόλων σχεδόν σίγουρα αντιπροσωπεύει μια γνωστή λέξη. Μαντεύοντας σωστά μερικές από τις κρυπτογραφημένες λέξεις του κρυπτοκειμένου, μπορούσαν να καθορίζουν τη συνδεσμολογία της μηχανής, δοκιμάζοντας όλες τις πιθανές συνδεσμολογίες και προσδιορίζοντας ποια είχε ως αποτέλεσμα τα υποτιθέμενα ζευγάρια κρυπτογραφημένων-αποκρυπτογραφημένων λέξεων. Ο Turing αντιλήφθηκε ότι μόνο μια αυτόματη και σχετικά γρήγορη μηχανή θα μπορούσε να τα βγάλει πέρα με τις δοκιμές, όποτε και οδηγήθηκε στην κατασκευή ενός εξομοιωτή της Μηχανής - Αίνιγμα με το όνομα Bombe.

Σύγχρονη Κρυπτογραφία

Η σημερινή μορφή των κρυπτογραφικών συστημάτων έχει καθοριστεί σε πολύ μεγάλο βαθμό από δύο, κεφαλαιώδους σημασίας για την κρυπτογραφία - και τις επικοινωνίες γενικότερα - επιστημονικές εργασίες των Kerchoffs και Shannon, που δημοσιεύτηκαν στα 1883 και 1949 αντίστοιχα. Στην πρώτη, ο Kerchoffs έθεσε τη βασική σχεδιαστική αρχή που έκτοτε διέπει κάθε κρυπτογραφικό σύστημα, σύμφωνα με την οποία η ασφάλεια ενός συστήματος πρέπει να έγκειται μόνο στη μυστικότητα του κλειδιού και να μην εξαρτάται από τη μυστικότητα του αλγορίθμου κρυπτογράφησης. Η δεύτερη εργασία ανήκει στο θεμελιωτή της Θεωρίας Πληροφορίας Claude Shannon. Στην εργασία αυτή η κρυπτογραφία μετατρέπεται σε αυστηρό επιστημονικό πεδίο, όπου ορίζεται η έννοια του κρυπτοσυστήματος και η απόλυτη ασφάλεια.

Η εργασία αυτή του Shannon αποτέλεσε ισχυρή κινητήρια δύναμη για την ταχεία εξέλιξη της έρευνας στο χώρο της κρυπτογραφίας, η οποία έλαβε χώρα στο δεύτερο μισό του εικοστού αιώνα και συνεχίζεται μέχρι σήμερα. Έτσι, όλοι οι σύγχρονοι κρυπτογραφικοί αλγόριθμοι σχεδιάζονται υπό το πρίσμα των εννοιών που εισήγαγε.

Σημαντική τομή ωστόσο στο χώρο της κρυπτογραφίας αποτέλεσε και η εργασία των Diffie - Hellman το 1976, όπου προτάθηκε μία επαναστατική τεχνική η οποία επιλύει το πρόβλημα της ανταλλαγής κλειδιού από απόσταση, χωρίς να απαιτείται άμεση επαφή, θέτοντας έτσι τις βάσεις για την κρυπτογραφία δημοσίου κλειδιού. Πράγματι, το 1977 οι Ronald L. Rivest, Adi Shamir και Leonard M. Adleman πρότειναν ένα ιδιαίτερα επιτυχημένο (έως και σήμερα) κρυπτοσύστημα δημοσίου κλειδιού, γνωστό ως RSA.

Εκτός από την υιοθέτηση των παραπάνω αρχών, η σύγχρονη κρυπτογραφία έχει επεκτείνει σημαντικά το πεδίο εφαρμογής της πέρα από την ιδιωτική επικοινωνία. Οι μέθοδοί της επιτρέπουν τον έλεγχο της ακεραιότητας μηνυμάτων, την μη αποποίηση αποστολής ή λήψης τους, την χρονική τους σήμανση, και πλήθος άλλων ιδιοτήτων. Επιπλέον μπορεί να χρησιμοποιηθεί για την απόδειξη της ταυτότητας κάποιου χρήστη (αυθεντικοποίηση) και την παροχή εξουσιοδότησης για την πραγματοποίηση συγκεκριμένων λειτουργιών (ψηφιακή υπογραφή).

1.2 Η σημασία της Κρυπτογραφίας στον 21ο αιώνα

Ιδιαίτερα κατά τον 21ο αιώνα, η ραγδαία άνθηση των τηλεπικοινωνιών (με κυριότερο εκφραστή αυτής το διαδίκτυο) έχει καταστήσει την κρυπτογραφία αναπόσπαστο κομμάτι των τεχνολογικών εξελίξεων και αντικείμενο έντονης ερευνητικής δραστηριότητας.

Οι κρυπτογραφικές εφαρμογές, πρωτόκολλα και τεχνικές παίζουν κομβικό ρόλο, ειδικά στους τομείς της ασφαλούς επικοινωνίας, της ασφαλούς πρόσβασης σε συστήματα και υπηρεσίες, των ηλεκτρονικών ψηφοφοριών, των στρατιωτικών εφαρμογών, της ανάκτησης και διαχείρισης ευαίσθητων δεδομένων, του ηλεκτρονικού εμπορίου και των ηλεκτρονικών συναλλαγών, με πρόσφατη σημαντικότατη εξέλιξη την ανάπτυξη του κρυπτονομίσματος Bitcoin, και αρκετών διαδόχων του, με κυρίαρχο χαρακτηριστικό την απουσία κεντρικού ελέγχου.

Οι παραπάνω εφαρμογές, και πολλές άλλες που δεν αναφέρθηκαν, μπόρεσαν να πραγματοποιηθούν χάρη στην αλματώδη ανάπτυξη επαναστατικών ιδεών και αλγορίθμων, όπως η τέλεια μυστικότητα, η κρυπτογραφία δημοσίου κλειδιού, η ασφαλής ανταλλαγή κλειδιού από απόσταση, οι ψηφιακές υπογραφές, τα διαλογικά συστήματα αποδείξεων, οι αποδείξεις μηδενικής γνώσης, οι γεννήτριες ψευδοτυχαιότητας και οι συναρτήσεις σύνοψης χωρίς συγκρούσεις. Κοινό χαρακτηριστικό των παραπάνω μεθόδων είναι ότι η ασφάλειά τους εδράζεται, όλο και περισσότερο, σε αυστηρές μαθηματικές αποδείξεις.

Έτσι, τα μαθηματικά παραμένουν στην επιφάνεια, βοηθώντας στην εμπέδωση εμπιστοσύνης σε κρίσιμες λειτουργίες, και μέσω αυτής στο άνοιγμα της κρυπτογραφίας στο ευρύ κοινό. Τα περισσότερα κρυπτοσυστήματα είναι πλέον εντελώς ανοιχτά (ο τρόπος λειτουργίας είναι γνωστός), και η ασφάλειά τους βασίζεται αποκλειστικά σε αλγορίθμους που μας επιτρέπουν να εκτελούμε αποδοτικά πράξεις, οι αντίστροφες των οποίων έχουν απαγορευτική υπολογιστική πολυπλοκότητα. Η κρυπτογραφία έχει φύγει οριστικά από τα στεγανά των μυστικών υπηρεσιών και τη στρατιωτική χρήση και προσφέρει τις υπηρεσίες της στο σύνολο της ανθρωπότητας σε πλειάδα εκφάνσεων του καθημερινού της βίου.

Κεφάλαιο 2

Μαθηματικό & Υπολογιστικό Υπόβαθρο

Στο κεφάλαιο αυτό θα παρουσιάσουμε συνοπτικά το μαθηματικό υπόβαθρο στο οποίο έχει βασιστεί η σύγχρονη κρυπτογραφία. Θα αναφέρουμε βασικές έννοιες και θα παρουσιάσουμε θέματα τα οποία αποτελούν το θεμέλιο λίθο για τον ορισμό, την υλοποίηση και την ασφάλεια των σύγχρονων κρυπτοσυστημάτων.

2.1 Θεωρία Αριθμών

Σε αυτήν την ενότητα θα παρουσιάσουμε ορισμένα βασικά στοιχεία από τη Θεωρία Αριθμών.

Διαιρετότητα

Ορισμός 2.1.1 Έστω $a, b \in \mathbb{Z}$. Θα λέμε ότι ο a **διαιρεί** τον b και θα γράφουμε $a|b$, αν το b είναι ακέραιο πολλαπλάσιο του a, δηλαδή αν $\exists\ c \in \mathbb{Z}$ ώστε $b = ac$.

Παράδειγμα 2.1.2 Με βάση τον παραπάνω ορισμό προκύπτει ότι $7 \mid 28$ καθώς $28 = 7 \cdot 4$, ενώ $7 \nmid 28$ καθώς $\nexists\ c \in \mathbb{Z}$ ώστε $30 = 7 \cdot c$.

Πρόταση 2.1.3 Έστω $a, b \in \mathbb{Z}$. Τότε ισχύουν τα παρακάτω:

1. Αν $a \mid b$ και $a \mid c \Rightarrow a \mid bx + cy, \forall x, y \in \mathbb{Z}$.
2. Αν $a \mid b$ και $b \mid a \Rightarrow b = \pm a$.
3. Αν $a \mid b$ και $b \mid c \Rightarrow a \mid c$.
4. Αν $a \mid b$ και a, b θετικοί αριθμοί, τότε ισχύει ότι $a \leq b$.

Μέγιστος Κοινός Διαιρέτης

Ορισμός 2.1.4 Έστω $a, b \in \mathbb{Z}$ με τουλάχιστον έναν διάφορο του μηδενός. Ένας **μέγιστος κοινός διαιρέτης** των a και b (συμβολίζεται με $MK\Delta(a, b)$ ή (a, b)) είναι ένας θετικός ακέραιος d που έχει τις εξής ιδιότητες:

1. $d \mid a$ και $d \mid b$.
2. Αν $c \in \mathbb{Z}$ με $c \mid a$ και $c \mid b$, τότε $c \mid d$.

Ευκλείδεια Διαίρεση

Θεώρημα 2.1.5 Έστω $a, b \in \mathbb{Z}$ με $a \neq 0$. Τότε υπάρχουν μοναδικοί $q, r \in \mathbb{Z}$ ώστε $b = qa + r$, με $0 \leq r < |a|$.

Θεώρημα 2.1.6 Έστω $a, b \in \mathbb{Z}$ με τουλάχιστον έναν διάφορο του μηδενός. Τότε υπάρχει μοναδικός μέγιστος κοινός διαιρέτης των a και b (δηλαδή μοναδικός d ώστε $d = MK\Delta(a, b)$. Επιπλέον υπάρχουν $x, y \in \mathbb{Z}$ τέτοιοι ώστε $MK\Delta(a, b) = ax + by$.

Πρόταση 2.1.7 Έστω $b > 1$ ένας ακέραιος αριθμός. Τότε κάθε θετικός ακέραιος n έχει μοναδική αναπαράσταση της μορφής

$$n = a_k b^k + a_{k-1} b^{k-1} + ... + a_1 b + a_0,$$

όπου $k, a_j \in \mathbb{N}$ με $0 \leq a_j \leq b - 1$ για $j = 0, 1, ..., k$ και $a_k \neq 0$.

Άσκηση 2.1.8 Να γράψετε τον ακέραιο αριθμό 110 από το δεκαδικό στο δυαδικό σύστημα αρίθμησης.

Λύση:
Αρχικά γράφουμε τις δυνάμεις του 2 μέχρι να ξεπεράσουμε το 110. Έχουμε:

$$2^0 = 1 < 110$$
$$2^1 = 2 < 110$$
$$2^2 = 4 < 110$$
$$2^3 = 8 < 110$$
$$2^4 = 16 < 110$$
$$2^5 = 32 < 110$$
$$2^6 = 64 < 110$$

$$2^7 = 128 > 110$$

Από τον Αλγόριθμο της Διαίρεσης έχουμε ότι:

$$110 = 1 \cdot 2^6 + 46, \;\; 46 < 2^6 = 64$$
$$46 = 1 \cdot 2^5 + 14, \;\; 14 < 2^5 = 32$$
$$14 = 0 \cdot 2^4 + 14, \;\; 14 < 2^4 = 16$$
$$14 = 1 \cdot 2^3 + 6, \;\; 6 < 2^3 = 8$$
$$6 = 1 \cdot 2^2 + 2, \;\; 2 < 2^2 = 4$$
$$2 = 1 \cdot 2^1 + 0, \;\; 0 < 2^1 = 2$$
$$0 = 0 \cdot 2^0 + 0$$

Άρα το $(110)_{10} = (1101110)_2$.

Πρώτοι αριθμοί

Ορισμός 2.1.9 Ένας θετικός ακέραιος $p \neq 1$ λέγεται **πρώτος** αν οι μόνοι διαιρέτες του είναι οι ± 1 και $\pm p$.

Θεώρημα 2.1.10 Κάθε θετικός ακέραιος διάφορος του 1 είναι πρώτος ή γινόμενο πρώτων αριθμών.

Θεώρημα 2.1.11 Υπάρχουν άπειροι το πλήθος πρώτοι αριθμοί.

Αριθμητική Υπολοίπων

Ορισμός 2.1.12 Έστω $m \in \mathbb{Z}$. Θα λέμε ότι δύο ακέραιοι a και b είναι ισότιμοι modulo m ή ισοϋπόλοιποι modulo m και θα συμβολίζουμε $a \equiv b \bmod m$, αν ο m διαιρεί τη διαφορά $a - b$. Δηλαδή $a \equiv b \bmod m \Leftrightarrow m \mid a - b$.

Παράδειγμα 2.1.13 Ισχύει $5001 \equiv 1 \bmod 5$, γιατί το $5 \mid 5001 - 1 \Rightarrow 5 \mid 5000$. Επίσης, $17 \equiv -3 \bmod 20$, γιατί το $20 \mid 17 - (-3) \Rightarrow 20 \mid 20$.

Παρατήρηση 2.1.14 $a \equiv b \bmod m \Leftrightarrow b \equiv a \bmod m$.

Παρατήρηση 2.1.15 Συμβολίζουμε με $\mathbb{Z}_m$, το δακτύλιο των ακεραίων modulo m και με $[a] \in \mathbb{Z}_m$ την κλάση ισοδυναμίας modulo m του $a \in \mathbb{Z}$.

Ορισμός 2.1.16 Η πρόσθεση στο $\mathbb{Z}_m$. Ορίζουμε την απεικόνιση:

$$+ : \mathbb{Z}_m \times \mathbb{Z}_m \to \mathbb{Z}_m : ([a], [b]) \to [a + b]$$

Ορισμός 2.1.17 Ο πολλαπλασιασμός στο $\mathbb{Z}_m$. Ορίζουμε την απεικόνιση:

$$\cdot : \mathbb{Z}_m \times \mathbb{Z}_m \to \mathbb{Z}_m : ([a], [b]) \to [a \cdot b]$$

Παρατήρηση 2.1.18 Οι παραπάνω πράξεις είναι καλά ορισμένες, δηλαδή δεν εξαρτώνται από την επιλογή των αντιπροσώπων.

Παράδειγμα 2.1.19 Έστω το $\mathbb{Z}_3 = [0],[1],[2]$.

- $[3] + [2] = [5] = [2]$
- $[3] \cdot [2] = [6] = [0]$

Παράδειγμα 2.1.20 Έστω το $\mathbb{Z}_{10} = [0], [1], [2], [3], [4], [5], [6], [7], [8], [9]$.

- $[3] + [2] = [5]$
- $[3] \cdot [5] = [15] = [5]$

Παρατήρηση 2.1.21 Ισχύουν οι παρακάτω ιδιότητες:

1. $([a] + [b]) + [c] = [a] + ([b] + [c]) = ([a] + [c]) + [b]$
2. $[a] + [0] = [0] + [a] = [a]$
3. $[a] + [-a] = [-a] + [a] = [0]$
4. $[a] + [b] = [b] + [a]$
5. $([a][b])[c] = [a]([b][c]) = ([a][c])[b]$
6. $[a]([b] + [c]) = [a][b] + [a][c]$
7. $([a] + [b])[c] = [a][c] + [b][c]$
8. $[a][b] = [b][a]$
9. $[a] + [1] = [1] + [a] = [a]$

Ορισμός 2.1.22 Έστω $m \in \mathbb{Z}^+$ και $a \in \mathbb{Z}$. Το $[a] \in \mathbb{Z}$ καλείται αντιστρέψιμο αν $\exists\ [b] \in \mathbb{Z}_m$ ώστε $[a][b] = [1]$. Το $[b]$, εάν υπάρχει, είναι μοναδικό, ονομάζεται αντίστροφο του $[a]$ και συμβολίζεται με $[a]^{-1}$.

Πρόταση 2.1.23 Έστω $m \in \mathbb{Z}^+$ και $a \in \mathbb{Z}$. Το $[a] \in \mathbb{Z}_m$ είναι αντιστρέψιμο $\Leftrightarrow MK\Delta(a, m) = 1$.

Το Μικρό Θεώρημα του Fermat

Θεώρημα 2.1.24 Έστω $a \in \mathbb{Z}$ και p πρώτος αριθμός, τότε ισχύει ότι $a^p \equiv a \bmod p \Leftrightarrow p \mid a^p - a$. Επιπλέον, αν ο p δεν διαιρεί τον a τότε ισχύει ότι $a^{p-1} \equiv 1 \bmod p \Leftrightarrow p \mid a^{p-1} - 1$.

Η Συνάρτηση του Euler

Ορισμός 2.1.25 Αν το $n \geq 1$, τότε με $\phi(n)$ συμβολίζουμε το πλήθος των k με $0 \leq k \leq n-1$ και $(k, n) = 1$. Η συνάρτηση ϕ ονομάζεται και συνάρτηση του Euler και είναι σπουδαίας σημασίας για τον κλάδο της Θεωρίας Αριθμών.

Πρόταση 2.1.26 Για κάθε p πρώτο, ισχύει $\phi(p^i) = p^i - p^{i-1}$. Επίσης, αν $\text{ΜΚΔ}(m, n) = 1$, τότε $\phi(m \cdot n) = \phi(m) \cdot \phi(n)$.

2.2 Θεωρία Ομάδων

Ορισμός 2.2.1 Έστω G ένα μη-κενό σύνολο και $\star$ μια διμελής πράξη στο G. Το ζεύγος $(G, \star)$ ονομάζεται **ομάδα**, αν:

1. Το σύνολο G είναι κλειστό ως προς την πράξη $\star$, αν δηλαδή ισχύει ότι για κάθε $a, b \in G$, το στοιχείο $c = a \star b \in G$.
2. Η πράξη $\star$ είναι προσεταιριστική, αν δηλαδή ισχύει ότι $(a \star b) \star c = a \star (b \star c)$, για κάθε $a, b, c \in G$.
3. Υπάρχει στοιχείο $e \in G$, για το οποίο ισχύει ότι $a \star e = e = a, \forall a \in G$. Το στοιχείο αυτό ονομάζεται ουδέτερο στοιχείο του G ως προς την πράξη $\star$.
4. Για κάθε $a \in G, \exists\ a' \in G$ για το οποίο ισχύει ότι $a' = a' = e$. Το στοιχείο αυτό ονομάζεται αντίστροφο στοιχείο του a και συμβολίζεται με a^{-1}.

Ορισμός 2.2.2 Μία ομάδα $(G, \star)$ λέγεται **μεταθετική** ή **αβελιανή** αν $x \star y = y \star x, \forall x, y \in G$.

Ορισμός 2.2.3 Έστω $m \in \mathbb{Z}^+$. Το σύνολο των αντιστρέψιμων στοιχείων του $\mathbb{Z}_m$ θα το συμβολίζουμε με $\cup(\mathbb{Z}_m)$, όπου $\cup(\mathbb{Z}_m) = \{[a] \in \mathbb{Z}_m : \text{ΜΚΔ}(a, m) = 1\}$.

Παράδειγμα 2.2.4 $\cup(\mathbb{Z}_5) = \{[1], [2], [3], [4]\}$.

Ορισμός 2.2.5 Προφανώς το παραπάνω σύνολο εφοδιασμένο με την πράξη του πολλαπλασιασμού, αποτελεί ομάδα. Ισχύει ότι $\phi(m) = |\cup(\mathbb{Z}_m)|$.

Ορισμός 2.2.6 Υποομάδα μιας ομάδας $(G, \star)$ λέγεται το ζεύγος $(S, \star)$ τέτοιο ώστε $S \subseteq G$ και $(S, \star)$ να είναι ομάδα.

Ορισμός 2.2.7 Μια ομάδα $(G, \star)$ λέγεται **κυκλική** αν υπάρχει στοιχείο $g \in (G, \star)$ με την ιδιότητα $\forall\ x \in G\ \exists\ y : x = g^y$. Το στοιχείο αυτό το ονομάζουμε και γεννήτορα της $(G, \star)$.

Ορισμός 2.2.8 Δακτύλιο ονομάζουμε μια τριάδα $(R, +, *)$ όπου το $(R, +)$ είναι μεταθετική ομάδα και ισχύει η προσεταιριστική ιδιότητα για την πράξη $*$ και η επιμεριστική ιδιότητα της $*$ ως προς την $+$:

$$\forall a, b, c \in R \begin{cases} a * (b + c) = a * b + a * c, \\ (b + c) * a = b * a + c * a \end{cases}$$

Ορισμός 2.2.9 Ο δακτύλιος λέγεται **μεταθετικός** αν η πράξη $*$ έχει την αντιμεταθετική ιδιότητα.

Ορισμός 2.2.10 Ο δακτύλιος ονομάζεται **δακτύλιος με μονάδα** αν υπάρχει μοναδιαίο στοιχείο ως προς την $*$.

Ορισμός 2.2.11 Σώμα λέγεται μια τριάδα $(F, +, *)$ όπου το $(F, +)$ και $(F - e_+, *)$ είναι μεταθετικές ομάδες και ισχ1ει η επιμεριστική ιδιότητα της $*$ ως προς την $+$. Με e_+ συμβολίζουμε το ουδέτερο ως προς την πράξη $+$.

Ορισμός 2.2.12 Το πλήθος των στοιχείων του συνόλου F ονομάζεται **τάξη του σώματος**.

Ορισμός 2.2.13 Ένα σώμα πεπερασμένης τάξης ονομάζεται **πεπερασμένο σώμα**.

Θεώρημα 2.2.14 Ένα σώμα τάξης m υπάρχει αν και μόνον αν το m είναι δύναμη πρώτου, δηλ. $m = p^n$ με $n \in N$ και p πρώτο.

Πόρισμα 2.2.15 Για κάθε δύναμη πρώτου υπάρχει ακριβώς ένα πεπερασμένο σώμα, το οποίο συμβολίζουμε $GF(p^n)$.

2.3 Δυαδικές Πράξεις

Παρακάτω θα παρουσιάσουμε τις βασικές δυαδικές πράξεις που χρησιμοποιούνται στην κατασκευή συμμετρικών κρυπτοσυστημάτων όπως θα δούμε σε επόμενο κεφάλαιο.

Οι πιο συνηθισμένες τέτοιες πράξεις είναι η (αποκλειστική) διάζευξη (ή bitwise XOR) και οι περιστροφές.

Ορισμός 2.3.1 (Αποκλειστική) Διάζευξη είναι η γνωστή συνάρτηση δύο μεταβλητών

$$\oplus : \mathbb{B}^l \times \mathbb{B}^l \to \mathbb{B}^l$$

για κάποιο $l \geq 1$, η οποία σε κάθε δύο $x, y \in \mathbb{B}^l$ επιστρέφει ως $x \oplus y$ την ακολουθία που έχει ως i-οστό ψηφίο ($i = 1, ..., l$) το άθροισμα (modulo 2) των i-οστών ψηφίων των x και y. Είναι εύκολο να επαληθεύσει κανείς ότι με αυτήν την πράξη το $\mathbb{B}^l$ γίνεται αβελιανή ομάδα με ουδέτερο στοιχείο την ακολουθία 0^l των l μηδενικών και με αντίστροφο κάθε ακολουθίας την ίδια την ακολουθία.
Αυτό πρακτικά σημαίνει ότι μπορούμε, π.χ., την εξίσωση

$$((x_3 \oplus x_4) \oplus (x_1 \oplus x_2)) \oplus x_5 = y_1 \oplus (y_2 \oplus y_3)$$

να την γράψουμε κατευθείαν και ισοδύναμα ως:

$$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 = y_1 \oplus y_2 \oplus y_3$$

(δηλαδή να διώξουμε όλες τις παρενθέσεις και να αλλάξουμε τη σειρά των μεταβλητών) αλλά και ως:

$$x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus y_1 \oplus y_2 \oplus y_3 = 0^l$$

(δηλαδή να φέρουμε όλες τις μεταβλητές στο ένα μέλος και να αφήσουμε στο άλλο μέλος το 0^l).

Συχνά, η πράξη εμφανίζεται με το ένα από τα ορίσματά της σταθερό, δηλαδή ως η συνάρτηση μιας μεταβλητής $\oplus_u : \mathbb{B}^l \to \mathbb{B}^l$, με $\oplus_u(x) = x \oplus u$, για κάποιο $u \in \mathbb{B}^l$, και τότε βέβαια είναι μια αντιστοιχία με αντίστροφη τον εαυτό της. Επιπλέον, τις περισσότερες φορές η σταθερά u είναι συνάρτηση του κλειδιού k (συνηθέστερα, μια επιλογή ψηφίων του k). Τότε λέμε ότι η $\oplus_u$ είναι μια συνάρτηση λεύκανσης (whitening)

Ορισμός 2.3.2 Περιστροφές είναι οι πράξεις $<<<$: $\mathbb{B}^l \times \{0, ..., l-1\} \to \mathbb{B}^l$ και $>>>$: $\mathbb{B}^l \times \{0, ..., l-1\} \to \mathbb{B}^l$ για κάποιο $l \geq 1$, οι οποίες για κάθε $x \in \mathbb{B}^l$ και κάθε $i = 0, ..., l-1$ επιστρέφουν ως $x <<< i$ και $x >>> i$ τις ακολουθίες που προκύπτουν από την κυκλική περιστροφή της x κατά i ψηφία προς τα αριστερά ή δεξιά, αντίστοιχα. Όταν το μήκος l των ακολουθιών δεν είναι σαφές, θα γράφουμε τις $<<<, >>>$ και ως $<<<_l, >>>_l$. Είναι προφανές ότι οι περιστροφές είναι αντιστοιχίες και μάλιστα $(<<<_l)^{-1} = >>>_l$.

Στις περισσότερες περιπτώσεις, οι περιστροφές χρησιμοποιούνται με το δεύτερο από τα ορίσματά τους σταθερό και μόνο πρόσφατα αυτό το όρισμα άρχισε να εξαρτάται από τις εισόδους των συναρτήσεων κρυπτογράφησης και αποκρυπτογράφησης. Πρόκειται για μια σημαντική καινοτομία που μπορεί να κάνει τους αλγόριθμους πολύ πιο απλούς, και μάλιστα χωρίς συνέπειες στην ασφάλεια και την ταχύτητα.

2.4 Διακριτός Λογάριθμος

Ορισμός 2.4.1 Για κάθε $a > 0$ και $b > 0$, με $a \neq 1$ ορίζουμε ως **λογάριθμο** το:

$$log_a b = x \Leftrightarrow a^x = b$$

Γενικά για τους λογαρίθμους ισχύουν οι παρακάτω ιδιότητες:

1. $log_a(x \cdot y) = log_a x + log_a y$
2. $log_a(\frac{x}{y}) = log_a x - log_a y$
3. $log_a x^y = y \cdot log_a x$

Παρατήρηση 2.4.2 Ο λογάριθμος $log_a b$ δεν είναι πάντα ακέραιος αριθμός.

Παρατήρηση 2.4.3 Έστω $log_2 3 = x \Leftrightarrow 2^x = 3$. Με χρήση υπολογιστή προκύπτει ότι $x approx 1.58496$, καθώς $2^{1.58496} approx 2.99999 approx 3$.

Το ενδιαφέρον μας θα επικεντρωθεί στο διακριτό αντίστοιχο του λογαρίθμου, το οποίο εμφανίζεται όταν πολλαπλασιάσουμε modulo έναν αριθμό.

Ορισμός 2.4.4 Έστω G μία πεπερασμένη κυκλική ομάδα τάξης n, a ένας γεννήτορας της G και $b \in G$. Ο **Διακριτός Λογάριθμος** του b στη βάση a συμβολίζεται με $log_a b$, είναι ο μοναδικός ακέραιος $x, 0 \leq x \leq n-1$, τέτοιος ώστε $b = a^x$.

Ορισμός 2.4.5 Για p πρώτο αριθμό, a ένα γεννήτορα του $\mathbb{Z}_p^*$ και ένα στοιχείο $b \in \mathbb{Z}_p^*$ ορίζουμε ως **Πρόβλημα του Διακριτού Λογαρίθμου στο** $\mathbb{Z}_p^*$, το πρόβλημα του υπολογισμού ενός ακέραιου αριθμού $x \in \{0, 1, ..., p-2\}$ ώστε να ισχύει $a^x \equiv b \bmod p$.

Υπάρχουν ομάδες για τις οποίες η λύση στο πρόβλημα του διακριτού λογαρίθμου είναι αρκετά δύσκολη. Μάλιστα, σε κάποιες περιπτώσεις (π.χ. υποομάδες του $(\mathbb{Z}_p, \cdot)$, όπου p πολύ μεγάλος πρώτος, ο αλγόριθμος για εύρεση λύσης τόσο στο μετριοπαθές όσο και στο χειρότερο σενάριο αποδεικνύεται εξίσου μη αποδοτικός.

Την ίδια στιγμή, το αντίστροφο πρόβλημα της εκθετοποίησης υπολοίπων είναι εύκολα επιλύσιμο (π.χ. μέσω επαναλαμβανόμενου τετραγωνισμού). Η παραπάνω ασυμμετρία είναι αντίστοιχη με αυτήν που παρατηρείται μεταξύ παραγοντοποίησης ακεραίων και πολλαπλασιασμού ακεραίων. Για το λόγο αυτό τα παραπάνω προβλήματα έχουν αξιοποιηθεί σε μεγάλο βαθμό στις σύγχρονες κρυπτογραφικές τεχνικές.

2.5 Ελλειπτικές Καμπύλες

Στη συνέχεια θα δούμε ορισμένα πράγματα για τις ελλειπτικές καμπύλες. Οι ελλειπτικές καμπύλες, ως αντικείμενο της θεωρίας αριθμών και της αλγεβρικής γεωμετρίας, έχουν μελετηθεί για περισσότερο από δύο αιώνες και η θεωρία που έχει αναπτυχθεί γύρω τους είναι ιδιαίτερα πλούσια σε αποτελέσματα. Ωστόσο την τελευταία εικοσαετία το ενδιαφέρον της ακαδημαϊκής κοινότητας για τις ελλειπτικές καμπύλες έχει αυξηθεί σημαντικά, τόσο λόγω των εφαρμογών τους στην κρυπτογραφία, όσο και της άμεσης σχέσης της θεωρίας ελλειπτικών καμπυλών με την απόδειξη του τελευταίου θεωρήματος του Fermat. Η σημασία των ελλειπτικών καμπυλών στην κρυπτογραφία συνοψίζεται στο γεγονός ότι τα σημεία τους μπορούν να αντικαταστήσουν τα σημεία του $\mathbb{Z}_p^*$ με πολύ μεγάλη επιτυχία καθώς:

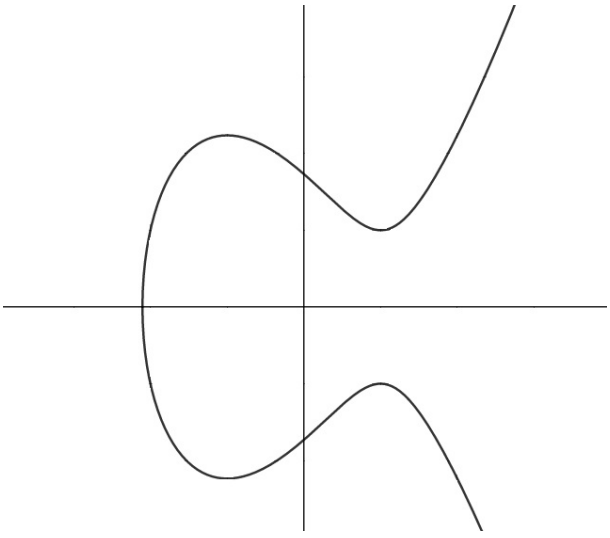
- Το πρόβλημα του διακριτού λογάριθμου στην ομάδα που σχηματίζουν αυτά δεν επιδέχεται υποεκθετικούς αλγορίθμους.
- Κατά συνέπεια μπορούμε να πετύχουμε ίδια και καλύτερα επίπεδα ασφάλειας χρησιμοποιώντας μικρότερες παραμέτρους ασφαλείας, δηλαδή λιγότερα bits κλειδιών και κατά συνέπεια πιο αποδοτικές λειτουργίες.

Τα μαθηματικά που απαιτούνται για τις ελλειπτικές καμπύλες είναι αρκετά πολύπλοκα και έχουν αναπτυχθεί σε βάθος αιώνων. Για τον λόγο αυτό η παρουσίαση τους ξεφεύγει από τους σκοπούς της παρούσας διπλωματικής εργασίας. Επιπλέον, η περιγραφή των εφαρμογών τους μπορεί να γίνει αφαιρώντας τις "εξειδικευμένες λεπτομέρειες", υποθέτοντας απλά μια κυκλική ομάδα όπου κάποιο πρόβλημα είναι δύσκολο.

Ορισμός 2.5.1 Η ελλειπτική καμπύλη στο $\mathbb{Z}_p$, $p \geq 3$, είναι το σύνολο όλων των σημείων $x, y \in \mathbb{Z}_p$, για τα οποία ισχύει ότι:

1. $y^2 \equiv (x^3 + \alpha x + \beta) \bmod p$, μαζί με ένα σημείο στο άπειρο το οποίο συμβολίζουμε $\mathscr{O}$, όπου τα $\alpha, \beta \in \mathbb{Z}_p$.
2. $4\alpha^3 + 27\beta^2 \not\equiv 0 \bmod p$.

Παράδειγμα 2.5.2 Έστω η εξίσωση $(E) : y^2 = x^3 - 3x + 3$. Θα τη μελετήσουμε στο καρτεσιανό σύστημα.



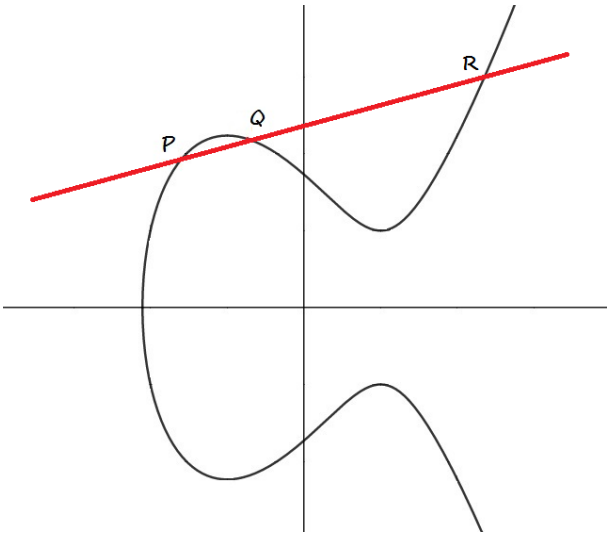
Αρχικά, παρατηρούμε ότι είναι συμμετρική ως προς τον άξονα των x. Αυτό συμβαίνει γιατί ισχύει ότι:

$$y^2 = x^3 + \alpha x + \beta \Rightarrow y = \pm\sqrt{x^3 + \alpha x + \beta}$$

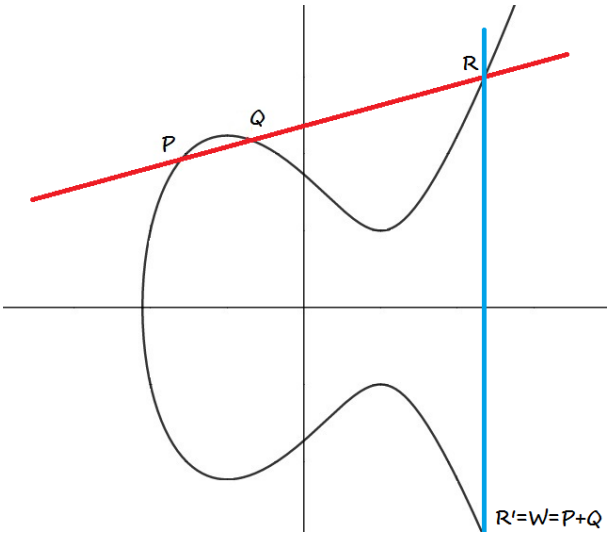
Επιστρέφοντας στο παράδειγμά μας, έστω $x = 2 \Rightarrow y^2 = 2^3 - 3 \cdot 2 + 3 = 5 \Rightarrow y = \pm\sqrt{5}$. Επομένως, προκύπτουν δύο σημεία τα $A_1(2, \sqrt{5})$ και $A_2(2, -\sqrt{5})$, τα οποία προφανώς είναι συμμετρικά ως προς τον άξονα των x.

Στην ουσία, αυτό που αναζητούμε είναι να επιλύσουμε ένα πρόβλημα διακριτού λογαρίθμου. Άρα χρειάζεται να ορίσουμε μια κυκλική ομάδα. Ως στοιχεία της θα επιλέξουμε τα στοιχεία της καμπύλης, των οποίων οι συντεταγμένες είναι ακέραιοι αριθμοί, καθώς στη συνέχεια θα εργαστούμε modulo p και θα ορίσουμε μια πράξη μεταξύ τους.

Αρχικά, ενώνουμε τα σημεία P, Q με μια ευθεία γραμμή. Έτσι, προκύπτει το εξής:



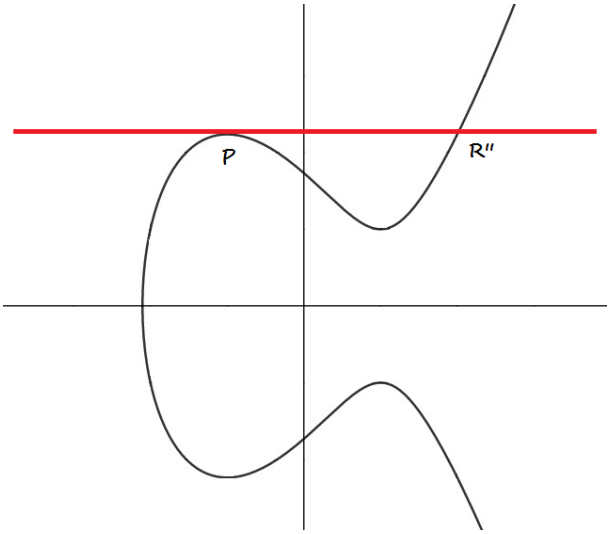
Παρατηρούμε ότι η ευθεία αυτή, τέμνει την ελλειπτική καμπύλη και σε ένα τρίτο σημείο, εκτός από τα σημεία P, Q. Ας ονομάσουμε αυτό το σημείο R. Από τον ορισμό της ελλειπτική καμπύλης, αποδεικνύεται ότι υπάρχει και τρίτο σημείο τομής. Αυτό που μένει να κάνουμε είναι να εντοπίσουμε το συμμετρικό του σημείου αυτού ως προς τον άξονα των x, φέρνοντας την κάθετη στον εν λόγω άξονα και επεκτείνοντάς την μέχρι να έχουμε την τομή με την ελλειπτική καμπύλη E.



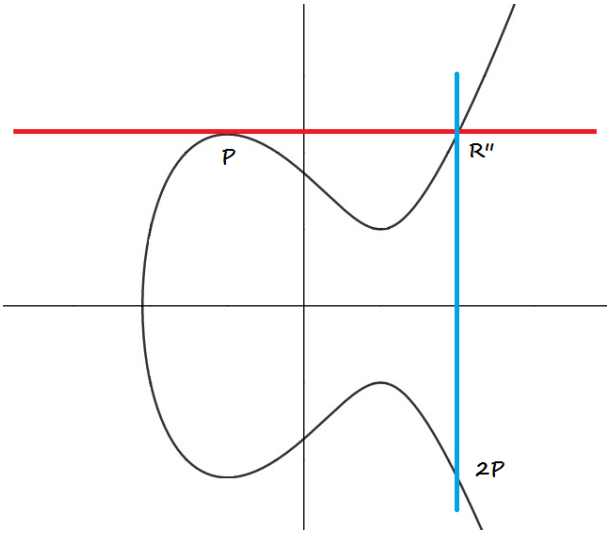
Εξ ορισμού το σημείο $R' = W$ είναι το άθροισμα των σημείων P, Q. Η πράξη αυτή ονομάζεται point addition. Υπάρχει όμως και μία περίπτωση στην οποία αυτός ο ορισμός δεν δουλεύει. Ας δούμε τι συμβαίνει όταν θέλουμε να προσθέσουμε το P στο P.

Στην περίπτωση αυτή, η κατασκευαστική μέθοδος που είδαμε σταματάει να δουλεύει. Προκειμένου

να ξεπεράσουμε το εμπόδιο αυτό, κάνουμε το εξής τέχνασμα. Στο σημείο P θα φέρουμε εφαπτομένη. Έτσι προκύπτει το εξής:



Βλέπουμε ότι η εφαπτομένη τέμνει την E στο σημείο R''. Από εκεί και πέρα με την ίδια κατασκευαστική τεχνική θα προκύψει το άθροισμα $P + P = 2P$.



Παρακάτω βλέπουμε την ίδια κατασκευή, αριθμητικά. Θέλουμε την αναλυτική έκφραση για την πράξη της κυκλικής ομάδας. Έστω η ελλειπτική καμπύλη:

$$E : y^2 = x^3 + \alpha x + \beta$$

μαζί με δύο σημεία, τα $P(x_1, y_1)$ και $Q(x_2, y_2)$.
Το πρώτο πράγμα που κάναμε στη γεωμετρική κατασκευή, ήταν να φέρουμε την ευθεία γραμμή, η οποία ενώνει τα σημεία P και Q. Έστω ότι αυτή η ευθεία έχει εξίσωση:

$$l : y = sx + m$$

όπου s είναι η κλίση της ευθείας l. Στη συνέχεια, θέλουμε τα σημεία τομής της ευθείας l με την καμπύλη E. Να σημειώσουμε, ότι μόνο τα δύο από τα τρία σημεία τομής είναι γνωστά. Είναι τα P και Q. Ψάχνουμε το $R = (x_3, y_3)$. Για να το βρούμε λύνουμε την εξίσωση $l = E$.

Έχουμε:

$$(sx+m)^2 = x^3+\alpha x+\beta \Leftrightarrow s^2x^2+2sxm+m^2 = x^3+\alpha x+\beta \Leftrightarrow x^3-s^2x^2+(\alpha-2sm)x+(\beta-m^2) = 0$$

Η εξίσωση είναι τρίτου βαθμού, επομένως γνωρίζουμε ότι αφού τα x_1, x_2 και x_3 είναι λύσεις της, έπεται ότι

$$(x-x_1)(x-x_2)(x-x_3) = x^3 - s^2x^2 + (\alpha - 2sm)x + (\beta - m^2) = 0$$

Κάνοντας τις πράξεις στο αριστερό μέλος έχουμε:
$(x-x_1)(x-x_2)(x-x_3) = (x^2 - xx_2 - xx_1 + x_1x_2)(x-x_3) = x^3 - x^2x_2 - x^2x_1 + xx_1x_2 - x^2x_3 + x_2x_3 + xx_1x_3 - x_1x_2x_3 = x^3 + x^2(-x_1 - x_2 - x_3) + x(x_1x_2 + x_1x_3 + x_2x_3) - x_1x_2x_3$.

Εξισώνοντας τους συντελεστές των παραπάνω εξισώσεων προκύπτει:

$$-x_1 - x_2 - x_3 = -s^2 \Rightarrow x_3 = s^2 - x_1 - x_2,$$

όπου ως γνωστόν $s = \dfrac{y_2 - y_1}{x_2 - x_1}$
Τέλος, ισχύει ότι:

$$y_3 = s(x_1 - x_3) - y_1$$

Όλα τα παραπάνω, ισχύουν στην περίπτωση στην οποία έχουμε δύο διαφορετικά σημεία, τα P και Q και θέλουμε να τα προσθέσουμε (point addition).

Στην περίπτωση που έχουμε μόνο ένα σημείο και θέλουμε να κάνουμε πρόσθεση με τον εαυτό του (point doubling), έχουμε τους τύπους:

$$x_3 = s^2 - 2x_1$$

$$y_3 = s(x_1 - x_3) - y_1$$

$$s = \frac{3x_1^2 + a}{2y_1}$$

Αποδεικνύεται ότι το σύνολο αυτό, με την πράξη που ορίσαμε, αποτελεί ομάδα με ουδέτερο στοιχείο το $\mathcal{O}$. Τέλος, αποδεικνύεται ότι είναι κυκλική.

2.6 Καταχωρητές ολίσθησης με ανάδραση (FSR)

Έστω το πεπερασμένο σώμα $\mathbb{F}_q$ που αποτελείται από q στοιχεία.

2.6.1. Ορισμός Μία ακολουθία $y = \{y_i\}_{i\geq 0}$ με στοιχεία στο σώμα $\mathbb{F}_q$ ονομάζεται τελικά περιοδική (ultimately periodic) εάν υπάρχουν ακέραιοι $T > 0$ και $t_0 \geq 0$ τέτοιοι ώστε $y_{i+T} = y_i$, $\forall i \geq t_0$. Ο μικρότερος ακέραιος T με την παραπάνω ιδιότητα ονομάζεται πρωταρχική περίοδος (fundamental period) της ακολουθίας y ή απλά περίοδος, ενώ ο ακέραιος t_0 εκφράζει την προ-περίοδο (preperiod) της y. Αν $t_0 = 0$, τότε η ακολουθία y καλείται περιοδική (periodic).

Αν μία ακολουθία παίρνει τιμές στο $\mathbb{F}_2 = \{0, 1\}$, τότε καλείται **δυαδική ακολουθία** - αυτή είναι και η συνηθέστερη περίπτωση για κρυπτογραφικές εφαρμογές.

Περιοδικές ή τελικά περιοδικές ακολουθίες παράγονται από καταχωρητές ολίσθησης με ανάδραση (FeedBack Shift Register (FSR)). Ένας καταχωρητής μήκους n στο σώμα $\mathbb{F}_q$ αποτελείται από n θέσεις μνήμης ή βαθμίδες, κάθε μία εκ των οποίων μπορεί να περιέχει ένα στοιχείο του σώματος $\mathbb{F}_q$. Σε κάθε κύκλο, το περιεχόμενο της κάθε θέσης μνήμης μετατοπίζεται κατά μία θέση δεξιά, ενώ η τιμή της αριστερότερης βαθμίδας καθορίζεται από την πράξη ανάδρασης h. Συνεπώς, κάθε ακολουθία που παράγεται από έναν τέτοιο καταχωρητή ικανοποιεί την ακόλουθη αναδρομική σχέση

$$y_{i+n} = h(y_{i+n-1}, ..., y_i), i \geq 0$$

όπου η συνάρτηση $h : \mathbb{F}_q^n \to \mathbb{F}_q$ είναι στη γενική περίπτωση μη γραμμική.

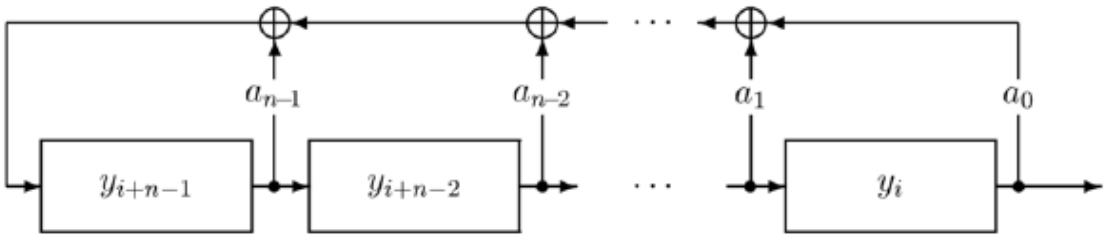
Για κάθε χρονική στιγμή, τα περιεχόμενα των θέσεων μνήμης ορίζουν την κατάσταση (state) του FSR. Με άλλα λόγια, αν y είναι η παραγόμενη ακολουθία από έναν FSR, τότε η κατάστασή του για κάθε χρονική στιγμή $i \geq 0$ δίνεται από το διάνυσμα $(y_{i+n-1}y_{i+n-2}...y_i)$. Προφανώς, ένας FSR μήκους n μπορεί να περάσει από q^n διαφορετικές καταστάσεις. Συνεπώς, η μέγιστη περίοδος που μπορεί να έχει μία ακολουθία που παράγεται από έναν FSR n βαθμίδων είναι q_n.

Αν περιοριστούμε στους FSR των οποίων η συνάρτηση ανάδρασης έχει μηδενικό σταθερό όρο, τότε η μέγιστη δυνατή περίοδος της ακολουθίας εξόδου είναι $q^n - 1$, λόγω του ότι αν ο FSR περάσει από τη μηδενική κατάσταση θα παραμείνει για πάντα σε αυτή.

Ορισμός 2.6.2 Γραμμικοί καταχωρητές ολίσθησης με ανάδραση (Linear FeedBack Shift Register (LFSR)) ονομάζονται εκείνοι οι καταχωρητές των οποίων η συνάρτηση ανάδρασης είναι γραμμική, δηλαδή της μορφής

$$y_{i+n} = a_{n-1}y_{i+n-1} + ... + a_1y_{i+1} + a_0y_i, a_j \in \mathbb{F}_q, \forall j = 1, 2, ..., n$$

Ένας LFSR με n βαθμίδες απεικονίζεται στο παρακάτω σχήμα.



Σχήμα 2.6: Η λειτουργία ενός γραμμικού καταχωρητή ολίσθησης με ανάδραση

Το χαρακτηριστικό πολυώνυμο (characteristic polynomial) ενός LFSR που δίνεται από το παραπάνω σχήμα είναι το πολυώνυμο

$$f(x) = x^n - a_{n-1}x^{n-1} - ... - a_0 \in \mathbb{F}_q[x]$$

Ένας LFSR παράγει περιοδική ακολουθία αν και μόνο αν ο σταθερός όρος a_0 του χαρακτηριστικού του πολυωνύμου είναι μη μηδενικός - δηλαδή, αν f(0) $\neq 0$.

Για οποιονδήποτε LFSR μήκους n με χαρακτηριστικό πολυώνυμο f, αν η αρχική του κατάσταση είναι η 100...0 τότε η παραγόμενη ακολουθία έχει τη μέγιστη δυνατή περίοδο που μπορεί να έχει

οποιαδήποτε ακολουθία παράγεται από αυτόν τον LFSR. Η συγκεκριμένη περιοδική ακολουθία που προκύπτει καλείται ακολουθία κρουστικής απόκρισης (impulse response sequence). Οποιαδήποτε άλλη ακολουθία παράγεται από αυτόν τον LFSR έχει περίοδο που είναι διαιρέτης της περιόδου της κρουστικής απόκρισης ή ίση με αυτή. Με τη σειρά της, η περίοδος της ακολουθίας κρουστικής απόκρισης του LFSR ισούται με την τάξη $ord(f)$ του χαρακτηριστικού πολυωνύμου f - δηλαδή, είναι ίση με τον μικρότερο ακέραιο k τέτοιον ώστε το f να διαιρεί το $x^k - 1$.

Ισχύει $ord(f) = q^n - 1$ αν και μόνο αν το f είναι πρωταρχικό πολυώνυμο (primitive polynomial) στο $\mathbb{F}_q$.

2.7 Πεπερασμένα Αυτόματα

Τα πεπερασμένα αυτόματα ή finite-state machine (FSM) αποτελούν μια αναπαράσταση υπολογιστή με εξαιρετικά περιορισμένη μνήμη. Είναι μια μηχανή η οποία βρίσκεται σε ακριβώς μία κατάσταση κάθε φορά από ένα σύνολο πεπερασμένων καταστάσεων. Η αλλαγή της κατάστασης ονομάζεται *μετάβαση* και είναι αποτέλεσμα εξωτερικής επίδρασης. Το πεπερασμένο αυτόματο ορίζεται από το σύνολο των δυνατών καταστάσεων, την αρχική του κατάσταση και τις συνθήκες για την πραγματοποίηση κάθε μετάβασης. Η μνήμη του περιορίζεται από τον αριθμό των καταστάσεών του.

2.7.1 Ορισμός. Πεπερασμένο Αυτόματο είναι μία πεντάδα $(Q, \Sigma, \delta, q_0, F)$, όπου:

1. Q είναι ένα πεπερασμένο σύνολο, τα στοιχεία του οποίου ονομάζονται **καταστάσεις**.
2. Σ είναι ένα πεπερασμένο σύνολο που ονομάζεται **αλφάβητο**.
3. $\delta : Q \times \Sigma \to Q$ είναι η **συνάρτηση μεταβάσεων**.
4. $q_0 \in Q$ είναι η **εναρκτήρια κατάσταση**.
5. $F \subseteq Q$ είναι το **σύνολο των καταστάσεων αποδοχής**.

Υπάρχουν δύο είδη πεπερασμένων αυτομάτων, τα αιτιοκρατικά (ντετερμινιστικά) και τα ανταιτιοκρατικά (ή πιθανοκρατικά ή μη-ντετερμινιστικά).

2.7.2 Θεώρημα. Για κάθε ανταιτιοκρατικό πεπερασμένο αυτόματο υπάρχει ένα ισοδύναμο αιτιοκρατικό.

Βασικά στοιχεία των Πεπερασμένων Αυτομάτων

1. **Λέξη εισόδου:** Το αυτόματο διαβάζει μία πεπερασμένη σειρά από σύμβολα $a_1, a_2, ..., a_n$, $a_i \in \Sigma$, η οποία ονομάζεται λέξη εισόδου. Το σύνολο όλων των λέξεων εισόδου αποτελεί το Σ.
2. **Τρόπος λειτουργίας:** Αρχικά το αυτόματο βρίσκεται στην εναρκτήρια κατάσταση q_0 και αρχίζει να διαβάζει κατά σειρά τα σύμβολα της λέξης εισόδου. Όταν διαβάσει το σύμβολο a_i, το αυτόματο μεταβαίνει στην κατάσταση $q_i = \delta(q_{i-1}, a_i)$. Η q_n ονομάζεται τελική κατάσταση.
3. **Αποδεκτή λέξη:** Μία λέξη $w \in \Sigma$ είναι αποδεκτή από το αυτόματο αν $q_n \in F$.
4. **Αναγνωριζόμενη γλώσσα:** Το αυτόματο μπορεί να αναγνωρίζει μία τυπική γλώσσα. Η γλώσσα $L \subseteq \Sigma$ που αναγνωρίζεται από το αυτόματο είναι το σύνολο όλων των λέξεων που είναι αποδεκτές από αυτό.

Κεφάλαιο 3

Εξέλιξη Τηλεπικοινωνιών

Αν και η ανάγκη για εξ αποστάσεως επικοινωνία με αξιόπιστο και εύχρηστο τρόπο έκανε την εμφάνισή της από τα πρώτα βήματα της ανθρωπότητας, η δυνατότητα αυτή αποκτήθηκε μόλις τα τελευταία 200 περίπου χρόνια. Σε αυτό το κεφάλαιο, παρουσιάζεται συνοπτικά η εξέλιξη της τηλεφωνίας από την εφεύρεση του τηλεφώνου από τον Μπελ μέχρι και τα σημερινά έξυπνα κινητά τηλέφωνα που επικοινωνούν με χρήση του Διαδικτυακού Πρωτοκόλλου.

3.1 Τεχνολογία Μεταγωγής

Είναι σημαντικό πριν ξεκινήσουμε να αναφέρουμε τις κύριες κατηγορίες στις οποίες διαχωρίζονται οι τηλεπικοινωνίες, ανάλογα με τον τρόπο υλοποίησής τους και οι οποίες είναι:

- Η *μεταγωγή κυκλώματος*, στην οποία οι δύο κόμβοι (χρήστες) επικοινωνούν μέσω ενός καναλιού επικοινωνίας που χρησιμοποιείται αποκλειστικά από αυτούς, για όση ώρα διαρκεί η επικοινωνία. Έτσι, το κύκλωμα λειτουργεί σαν οι κόμβοι-χρήστες να ήταν πραγματικά συνδεδεμένοι.

- Η *μεταγωγή πακέτου*, στην οποία τα προς αποστολή δεδομένα ομαδοποιούνται για να μεταφερθούν μέσω ενός ψηφιακού δικτύου. Τα πακέτα πέραν του περιεχομένου τους (payload), έχουν και μια επικεφαλίδα (header), τα δεδομένα της οποίας αξιοποιούνται από το δίκτυο για να δρομολογήσουν το πακέτο στον προορισμό του. Αυτός ο τρόπος υλοποίησης χρησιμοποιείται κατά κόρον στα δίκτυα υπολογιστών παγκοσμίως.

- Υπάρχει, τέλος, και μια τρίτη κατηγορία η *εικονική μεταγωγή κυκλώματος*, στην οποία χρησιμοποιείται και πάλι η τεχνολογία της μεταγωγής πακέτου. Ωστόσο, προσομοιάζει στην μεταγωγή κυκλώματος αφού πριν τη μεταφορά των πακέτων εγκαθιδρύεται μια συγκεκριμένη σύνδεση και τα πακέτα αποστέλλονται σε σειρά.

3.2 Η Εφεύρεση του τηλεφώνου και η εξέλιξη των δικτύων

Το 1876 ο Αλεξάντερ Γκράχαμ Μπελ κατασκεύασε το πρώτο τηλέφωνο στον κόσμο, το οποίο αποτελούνταν από μία ελαστική μεμβράνη η οποία βρισκόταν μπροστά από σιδηρομαγνητικό πυρήνα ο οποίος ήταν περιτυλιγμένος με μονωμένο αγωγό. Τη συσκευή αυτή τη χρησιμοποιούσαν για ομιλίες σε κοντινές αποστάσεις. Μετά τη εφεύρεση του μικροφώνου από τον Ντέιβιντ Χίουζ το 1877, το τηλέφωνο εξελίχθηκε και χρησιμοποιούνταν για ομιλίες σε μακρινές αποστάσεις, ενώ μεσολάβησαν αρκετά στάδια μέχρις ότου λάβει τη μορφή που γνωρίζουμε σήμερα.

Για πολλά χρόνια από την εφεύρεση του, οι χρήστες έπρεπε να είναι άμεσα συνδεδεμένοι μεταξύ τους, γεγονός που αποδείχτηκε μη πρακτικό. Έτσι, σταδιακά άρχισε η λειτουργία τηλεφωνικών κέντρων με ένα από τα οποία ήταν συνδεδεμένος κάθε χρήστης. Με αυτόν τον τρόπο είχε τη δυνατότητα να συνομιλήσει με οποιονδήποτε ήταν συνδεδεμένος σε εκείνο το τηλεφωνικό κέντρο. Τα τηλεφωνικά κέντρα στη συνέχεια συνδέθηκαν μεταξύ τους, κάτι που είχε ως αποτέλεσμα το γνωστό μας δίκτυο PSTN (Public Switched Telephone Network).

Στην πρώτη εκδοχή επομένως της σταθερής τηλεφωνίας, τα σήματα κατευθυνόντουσαν μέσω ενός συνεστραμμένου ζεύγους καλωδίων από τον ένα χρήστη στον άλλο, περνώντας στο ενδιάμεσο από τα τηλεφωνικά κέντρα. Μάλιστα, πριν την εξέλιξη της τεχνολογίας, στην οποία θα αναφερθούμε παρακάτω, για είναι σε θέση να συνομιλήσουν 2 άτομα σήμαινε ότι «ενοικίαζαν» για όσο χρόνο μιλούσαν ένα συνεστραμμένο ζεύγος καλωδίων από τους παρόχους τηλεπικοινωνιών και το οποίο εκτεινόταν από την τοποθεσία του ενός μέχρι την τοποθεσία του άλλου. Αυτός ήταν και ο λόγος που παλαιότερα το κόστος των κλήσεων ήταν ιδιαίτερα αυξημένο. Όπως εύκολα συμπεραίνουμε, πρόκειται για χαρακτηριστικό δείγμα τεχνολογίας μεταγωγής κυκλώματος.

Αργότερα έκανε την εμφάνισή του το Διαδίκτυο, αλλάζοντας δραματικά τις συνήθειες αλλά και τις ανάγκες μας σε σχέση με την επικοινωνία και την αξιοποίηση του δικτύου τηλεφωνίας. Χάρη στην εξέλιξη της τεχνολογίας κατέστη εφικτή η μεγαλύτερη χωρητικότητα και εκμετάλλευση των δικτύων. Οι ταχύτητες μετάδοσης των δεδομένων αυξήθηκαν, σε αντίθεση με το κόστος των επικοινωνιών που έχει πέσει και όλα αυτά κυρίως χάρη στα δίκτυα οπτικών ινών, και τη δυνατότητα ασύρματης / δορυφορικής επικοινωνίας.

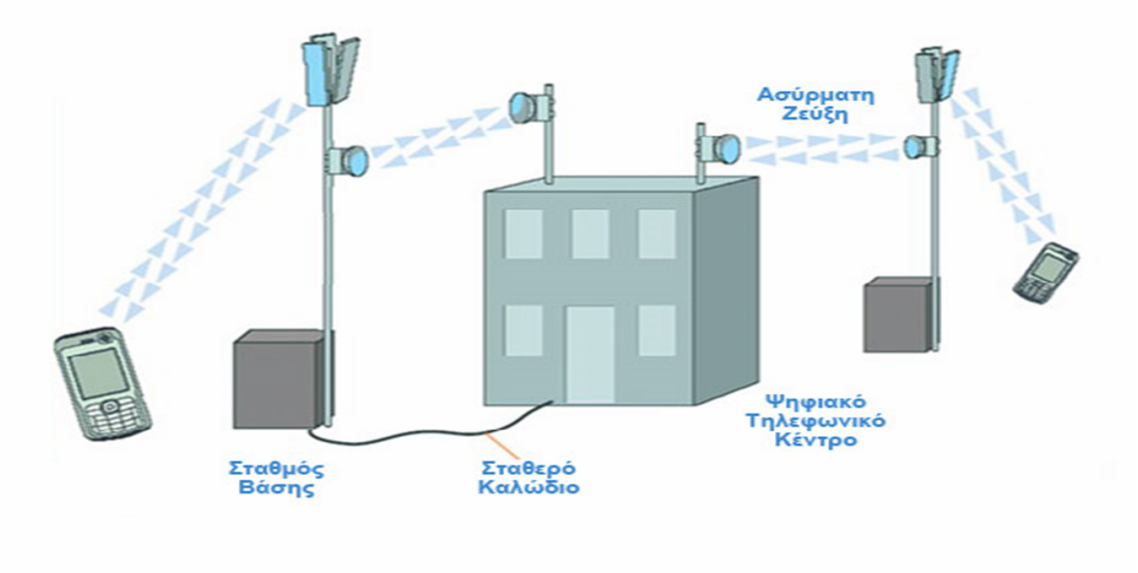
3.3 Εξέλιξη Κινητής Τηλεφωνίας

Η περιπέτεια της κινητής τηλεφωνίας ξεκίνησε αμέσως μετά τον Β' παγκόσμιο πόλεμο με τις πρώτες προσπάθειες Σουηδών ,Φινλανδών και Αμερικανών να δημιουργήσουν μια συσκευή που δεν θα εξαρτιόταν από καλωδιακή σύνδεση με δίκτυο παροχής τηλεφωνίας, ούτε από κάποια τοπική ασύρματη συσκευή εκπομπής ραδιοφωνικού σήματος χαμηλής συχνότητας. Η πρώτη συσκευή κινητού τηλεφώνου, εμφανίστηκε το 1973 και στη συνέχεια ακολούθησαν χιλιάδες μοντέλα που βοήθησαν στην εξέλιξη της μορφής αλλά και των λειτουργιών της, καθιστώντας την πλέον αναπόσπαστο μέρος της καθημερινότητάς μας.

Εικόνα 3.1: Μία από τις πρώτες συσκευές κινητής τηλεφωνίας

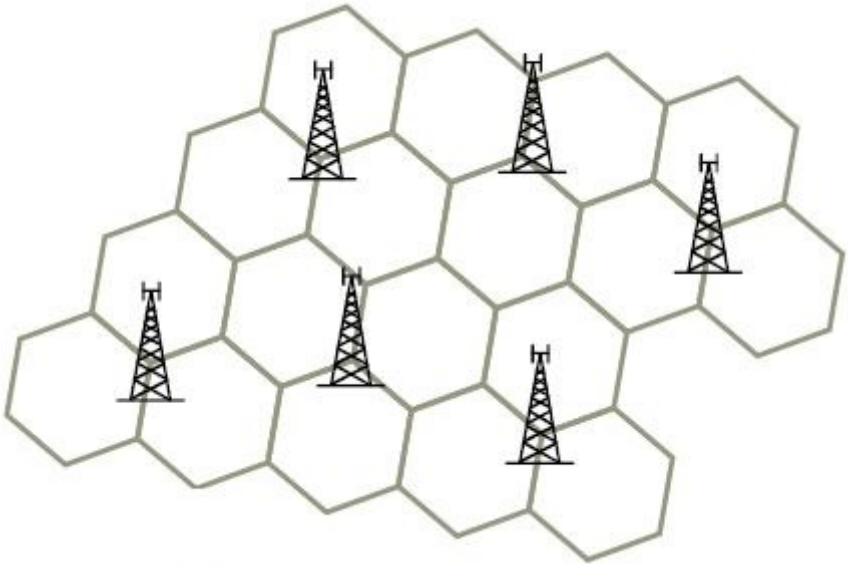
Αρχή λειτουργίας κινητού τηλεφώνου

Τα κινητά τηλέφωνα είναι πομποδέκτες ραδιοκυμάτων οι οποίοι με μια ενσωματωμένη κεραία και ηλεκτρονικό εξοπλισμό μετατρέπουν τη φωνή σε ψηφιακά (πλέον) δεδομένα και αντίστροφα. Για την αποστολή και λήψη των ραδιοκυμάτων χρησιμοποιούνται οι σταθμοί βάσης των παρόχων υπηρεσιών κινητής τηλεφωνίας. Έτσι, όταν κάποιος καλεί από το κινητό του τηλέφωνο αυτό εκπέμπει ραδιοκύματα τα οποία συλλαμβάνονται από το δέκτη στο πλησιέστερο σταθμό βάσης. Έπειτα, με ασύρματες ή ενσύρματες τεχνικές η κλήση δρομολογείται στο κατάλληλο ψηφιακό τηλεφωνικό κέντρο, από όπου δρομολογείται κατάλληλα προς το δέκτη της κλήσης ανάλογα αν πρόκειται για χρήστη κινητής ή σταθερής τηλεφωνίας.



Εικόνα 3.2: Απλοποιημένη αρχιτεκτονική κινητών τηλεπικοινωνιών

Πολύ συχνά οι σταθμοί βάσης αναφέρονται και ως «κυψέλες», κάτι που σχετίζεται άμεσα με τον τρόπο λειτουργίας του κινητού τηλεφώνου, το οποίο όπως προαναφέρθηκε εντοπίζει και συνδέεται με τον σταθμό βάσης για την επικοινωνία με τον οποίο υπάρχουν οι ευμενέστερες συνθήκες.



Εικόνα 3.3: Θεωρητική μοντελοποίηση ενός δικτύου

3.3.1 Γενιές Δικτύων Κινητής Τηλεφωνίας

Ο σκοπός των ασύρματων επικοινωνιών είναι να παρέχουν υψηλή ποιότητα και αξιοπιστία, όπως ακριβώς οι ενσύρματες επικοινωνίες. Κάθε νέα γενιά δικτύων αναπαριστά ένα μεγάλο βήμα – ένα άλμα θα λέγαμε – προς αυτήν την κατεύθυνση. Αυτή η επαναστατική διαδρομή ξεκίνησε το 1979 με την 1η γενιά δικτύων και συνεχίζεται ακόμη και σήμερα που βρισκόμαστε ένα βήμα πριν την εμπορική ανάπτυξη της 5ης γενιάς δικτύων. Κάθε γενιά δικτύων κινητής τεχνολογίας αποτελείται από ένα σύνολο προδιαγραφών που καθορίζουν λεπτομέρειες όπως την ταχύτητα μεταφοράς δεδομένων, το εύρος συχνοτήτων λειτουργίας, τη μέγιστη αποδεκτή χρονική υστέρηση, κλπ. Η τήρηση των προδιαγραφών αυτών είναι υποχρεωτική προκειμένου μια τεχνολογία να συμπεριληφθεί στις τεχνολογίες υλοποίησης μιας συγκεκριμένης γενιάς δικτύων. Για την εκπόνηση των παραπάνω

προδιαγραφών είναι επιφορτισμένα ινστιτούτα, τα οποία λαμβάνουν υπόψη την έρευνα και την τεχνολογική πρόοδο που έχει μεσολαβήσει από την ανάπτυξη της προηγούμενης γενιάς δικτύων.

Για να πραγματοποιηθούν οι εξελίξεις στα δίκτυα κινητής τηλεφωνίας, χρειάστηκε ορισμένες φορές να απελευθερωθούν περιοχές του ηλεκτρομαγνητικού φάσματος από άλλες λειτουργίες, (π.χ. τηλεόραση) και να αποδοθούν στην κινητή τηλεφωνία, καθώς συγκεκριμένες ζώνες του φάσματος είναι περισσότερο κατάλληλες για κάποιες από τις επιθυμητές λειτουργίες.

Ένα ακόμα σημαντικό στοιχείο που χαρακτηρίζει τις διάφορες γενιές δικτύων είναι ο τρόπος υλοποίησης της τεχνολογίας μεταγωγής. Εδώ, χαρακτηριστική είναι η μετεξέλιξη της τεχνολογίας που χρησιμοποιήθηκε: Από μεταγωγή κυκλώματος στην πρώτη γενιά και συνδυαστική λύση στις επόμενες, η τέταρτη γενιά πλέον υποστηρίζει αποκλειστικά τη μεταγωγή πακέτου.

1G - Πρώτη Γενιά

Η ιδέα της εμπορικής λειτουργίας των κινητών τηλεφώνων έγινε πραγματικότητα το 1979 στην Ιαπωνία. Αρχικά η τεχνολογία αυτή δεν προσδιοριζόταν ως «δίκτυο πρώτης γενιάς», ωστόσο αυτό έγινε αργότερα με την εμφάνιση των δικτύων δεύτερης γενιάς. Αυτή η γενιά δικτύων (πρώτη) ήταν βασισμένη σε αναλογικές τεχνολογίες. Η ταυτόχρονη πρόσβαση σε πολλούς χρήστες στην ίδια κυψέλη καθίστατο εφικτή με την τεχνική πολύπλεξης Frequency Division Multiple Access (FDMA). Η μετάβαση της επικοινωνίας από κυψέλη σε κυψέλη ήταν ήδη εφικτή, ωστόσο δεν υπήρχε δυνατότητα περιαγωγής καθώς δεν υπήρχε κάποιο συγκεκριμένο διεθνές πρότυπο για την ανάπτυξη αυτής της τεχνολογίας. Ο τελευταίος λόγος είναι και αυτός που οδήγησε σε διαφορές στην τεχνολογική υλοποίηση των δικτύων από χώρα σε χώρα. Όσον αφορά στις επιδόσεις των δικτύων αυτών, η ποιότητα της φωνής ήταν περιορισμένη, και χωρίς εξασφάλιση της προστασίας έναντι κακόβουλων τρίτων μερών, ενώ για την επικοινωνία χρησιμοποιούταν η μεταγωγή κυκλώματος. Η μόνη υπηρεσία που υποστηριζόταν από αυτά τα δίκτυα ήταν η μετάδοση φωνής. Η πρώτη γενιά δικτύων συνέχισε τη λειτουργία της μέχρι την αντικατάστασή της από τις ψηφιακές τηλεπικοινωνίες 2ης γενιάς.

2G - Δεύτερη Γενιά

Η δεύτερη γενιά δικτύων κινητής τηλεφωνίας εισήχθη στις αρχές της δεκαετίας του 1990 και έφερε μαζί της την ψηφιακή μετάδοση μεταξύ κινητής συσκευής και σταθμού βάσης καθώς και ένα σύνολο παροχών όπως αυξημένη χωρητικότητα του δικτύου, καλύτερη ποιότητα ομιλίας και δυνατότητα μεταφοράς δεδομένων. Η πρόοδος στην τεχνολογία μεταξύ 1ης και 2ης γενιάς μας προσέφερε νέες δυνατότητες όπως αποστολή γραπτών μηνυμάτων, μηνυμάτων πολυμέσων, αναμονή κλήσης καθώς και υπηρεσίες ασφάλειας στις επικοινωνίες. Η κυρίαρχη τεχνολογία 2G ήταν η Global System for Mobile (GSM) communications, ενώ η τεχνική πολύπλεξης για την ταυτόχρονη πρόσβαση πολλών χρηστών στην ίδια κυψέλη είναι η Time Division Multiple Access (TDMA). Πριν τη μετάβαση στα δίκτυα 3ης γενιάς χρησιμοποιήθηκαν τα δίκτυα General Packet Radio Service (GPRS) και Enhanced Data Rates for GSM Evolution (EDGE). Τα δίκτυα αυτά αποτελούν εκδόσεις που βασίστηκαν στο GSM και είναι επίσης γνωστά με τα ονόματα 2.5G και 2.75G αντίστοιχα. Αναπτύχθηκαν για να προσφέρουν υψηλότερη ταχύτητα μεταφοράς δεδομένων σε σύγκριση με το πρότυπο 2G, με τις μέγιστες ταχύτητες μετάδοσης δεδομένων να φτάνουν τα 40 Kbps για το πρώτο και τα 384 Kbps για το δεύτερο.

3G - Τρίτη Γενιά

Αυτή η γενιά έθεσε τις βάσεις για ό,τι αναγνωρίζουμε σήμερα ως αιχμή των κινητών τηλεπικοινωνιών. Περιήγηση στον Ιστό, υπηρεσίες ηλεκτρονικού ταχυδρομείου και μεταφόρτωσης βίντεο καθώς και πλήθος ακόμα άλλων υπηρεσιών που προσφέρουν τα έξυπνα κινητά τηλέφωνα, εισήχθησαν με την 3η γενιά δικτύων. Εμπορικά διαθέσιμη από το 2001, είχε ως στόχο την αύξηση της χωρητικότητας μεταφοράς ομιλίας και δεδομένων, την υποστήριξη μεγαλύτερου εύρους εφαρμογών και την αύξηση της μεταφοράς δεδομένων με ταυτόχρονη μείωση του κόστους. Μία από τις βασικές ιδέες για τα δίκτυα 3ης γενιάς ήταν η εξασφάλιση της δυνατότητας για παγκόσμια περιαγωγή: να είναι δηλαδή εφικτό ένας χρήστης να χρησιμοποιήσει τις υπηρεσίες κινητής τηλεφωνίας οπουδήποτε στον κόσμο. Έτσι, χάρη στη συνεργασία φορέων από την Ευρώπη, την Ασία και την Βόρεια Αμερική αναπτύχθηκε η πρώτη παγκόσμια κυψελωτή τεχνολογία, υπό την αιγίδα της 3rd Generation Partnership Project (3GPP). Η αρχιτεκτονική των δικτύων 3ης γενιάς αξιοποιεί μία νέα τεχνολογία τη UMTS (Univeral Mobile Telecommunications System). Σε αυτή συνδυάζονται ορισμένες από τις αρχές των δικτύων 2ης γενιάς με νέα πρωτόκολλα προκειμένου να επιτευχθεί σημαντική αύξηση στο ταχύτητα μετάδοσης των δεδομένων, πληρώντας τις προδιαγραφές International Mobile

Telecommunications-2000 (IMT-2000) για τα δίκτυα 3ης γενιάς που είχαν τεθεί από τη Διεθνή Ένωση Τηλεπικοινωνιών (International Telecommunication Union). Όντως, τα δίκτυα 3ης γενιάς αύξησαν την αποτελεσματικότητα του φάσματος συχνοτήτων βελτιώνοντας τη συμπίεση του ήχου κατά τη διάρκεια της κλήσης, ώστε πιο πολλές κλήσεις να μπορούν να πραγματοποιηθούν ταυτόχρονα στο ίδιο εύρος συχνοτήτων. Σύμφωνα με το πρότυπο, η μέγιστη θεωρητική ταχύτητα σε στάση είναι 2Mbps, ενώ σε κίνηση είναι 384Kbps. Όπως και στα δίκτυα 2ης γενιάς, έτσι και εδώ υπήρξαν μετεξελίξεις στην 3η γενιά των δικτύων που αποκαλούνται 3.5G και 3.75G με την μέγιστη θεωρητική ταχύτητα της τελευταίας να είναι 21.6Mbps. Η τεχνολογία που χρησιμοποιείται για την πολύπλεξη είναι η Wideband Code Division Multiple Access (WCDMA). Στην περίπτωση των δικτύων 3ης γενιάς, όπως και στην τεχνολογία GSM το σύστημα είναι χωρισμένο σε 2 τομείς ανάλογα με την τεχνολογία μεταγωγής. Ο τομέας μεταγωγής κυκλώματος προορίζεται για τη μεταφορά φωνής και σύντομων μηνυμάτων, ενώ ο τομέας μεταγωγής πακέτου χρησιμοποιείται για τη μεταφορά των δεδομένων.

4G - Τέταρτη Γενιά

Η 4η γενιά δικτύων διαφέρει κατά πολύ σε σύγκριση με την προηγούμενη. Ο σκοπός της είναι να παρέχει υψηλών προδιαγραφών ταχύτητα, ποιότητα και χωρητικότητα στους χρήστες συνδυάζοντας τα παραπάνω με τη βελτίωση της ασφάλειας των επικοινωνιών και τη μείωση του κόστους των υπηρεσιών φωνής, δεδομένων και πολυμέσων. Οι τωρινές αλλά και οι δυνητικές εφαρμογές περιλαμβάνουν μεταξύ άλλων την πρόσβαση στον Ιστό, την τηλεφωνία μέσω του Διαδικτυακού πρωτοκόλλου, τη μετάδοση υψηλής ευκρίνειας βίντεο καθώς και το υπολογιστικό νέφος. Οι τεχνολογίες – κλειδιά που καθιστούν τα παραπάνω εφικτά είναι η MIMO (Multiple Input Multiple Output), μέθοδος πολλαπλασιασμού της χωρητικότητας ενός ραδιοσυνδέσμου όπου με τη χρησιμοποίηση πολλαπλών κεραιών μετάδοσης και λήψης αξιοποιείται η πολυκατευθυντική διάδοση και η OFDM (Orthogonal Frequency Division Multiplexing), τύπος ψηφιακής διαμόρφωσης στην οποία το σήμα διαχωρίζεται σε πολλά περαιτέρω κανάλια μικρού εύρους σε διαφορετικές συχνότητες. Τα δύο πρότυπα δικτύων 4ης γενιάς είναι το WiMax και το LTE, με το τελευταίο να έχει γνωρίσει σημαντικότατη εξάπλωση σε όλο τον κόσμο. Θα πρέπει ωστόσο να επισημανθεί ότι το 4G LTE δεν πληροί τα κριτήρια για να ονομάζεται επίσημα 4G, καθώς υστερεί ως προς τη μέγιστη υποστηριζόμενη ταχύτητα μετάδοσης δεδομένων. Σύμφωνα με τις προδιαγραφές για το 4G, όταν η συσκευή βρίσκεται σε στάση θα πρέπει να υποστηρίζεται ταχύτητα έως 1Gbps, ενώ όταν βρίσκεται σε κίνηση έως 100 Mbps, ενώ η χρονική υστέρηση μετάδοσης (latency) έχει ως όριο τα 100ms. Σε αντίθεση με τις προηγούμενες γενιές δικτύων, τα δίκτυα 4ης γενιάς χρησιμοποιούν εξ ολοκλήρου την τεχνολογία μεταγωγής πακέτου[1]. Για το λόγο αυτό χρειάστηκε να πραγματοποιηθούν σημαντικές αλλαγές στην υποδομή των παρόχων τηλεπικοινωνιών οι οποίοι μέχρι πρότινος χρησιμοποιούσαν αποκλειστικά τη μεταγωγή κυκλώματος για να παρέχουν υπηρεσίες φωνής στους χρήστες. Βελτιωμένες εκδόσεις βασισμένες στα δίκτυα 4ης γενιάς είναι οι 4.5G και 4.9G (LTE-Advanced και LTE-Advanced Pro αντίστοιχα), οι οποίες προετοιμάζουν τον ερχομό των δικτύων 5ης γενιάς.

5G - Πέμπτη Γενιά

Η 5η γενιά δικτύων βρίσκεται υπό ανάπτυξη. Ο ερχομός της θα σημάνει μεταξύ άλλων ακόμα ταχύτερη μεταφορά δεδομένων, βελτιωμένη κάλυψη δικτύου και πολύ μικρότερη χρονική υστέρηση μετάδοσης. Ωστόσο αυτό που αναμένεται να προκαλέσει επανάσταση είναι η δυνατότητα διασύνδεσης των συσκευών μεταξύ τους (device-to-device communications), το αποκαλούμενο και Διαδίκτυο των Πραγμάτων ή δημοφιλέστερα IoT. Η μέγιστη ταχύτητα των δικτύων 5ης γενιάς θα είναι 35.46Gbps, δηλαδή 35 φορές ταχύτερη από την αυτήν των δικτύων της αμέσως προηγούμενης γενιάς. Για την επίτευξη των παραπάνω θα αξιοποιηθούν τεχνολογίες που αναπτύχθηκαν κατά την προηγούμενη δεκαετία όπως οι Massive MIMO , Millimeter Wave Mobile Communications, Li-Fi, καθώς και η τεχνολογία μικροκυψελών. Εκτιμάται ότι τουλάχιστον 100 δισεκατομμύρια συσκευές θα αξιοποιήσουν τα δίκτυα αυτά, τα οποία αναμένεται να κάνουν την επίσημη εμπορική εμφάνισή τους εντός του 2020. Η τεχνολογία στην οποία θα βασιστεί η ανάπτυξη αυτών των δικτύων έχει καθοριστεί από την 3GPP και φέρει την ονομασία 5G NR (New Radio).

[1]Για την ακρίβεια υπάρχουν 2 τεχνικές για τη μετάδοση ομιλίας στο 4G. Η μία χρησιμοποιεί όντως το 4G δίκτυο και άρα την τεχνολογία μεταγωγής πακέτου. Η άλλη μεταφέρει την επικοινωνία σε προγενέστερο δίκτυο, 2ης ή 3ης γενιάς και επομένως χρησιμοποιεί την τεχνολογία μεταγωγής κυκλώματος για τη μετάδοση ομιλίας.

3.4 Τηλεφωνία VOIP

Η τηλεφωνία μέσω Διαδικτυακού Πρωτόκολλου, ευρύτερα γνωστή ως VOIP (Voice over Internet Protocol), είναι ένα σύνολο από τεχνολογίες για την επίτευξη φωνητικών επικοινωνιών καθώς και επικοινωνιών μετάδοσης δεδομένων και πολυμέσων, με αξιοποίηση του Διαδικτυακού Πρωτοκόλλου, π.χ. του Ίντερνετ, έναντι των κλασικών μεθόδων μετάδοσης μέσω του PSTN δικτύου που είδαμε νωρίτερα.

Η πρώτη του εμπορική εμφάνιση πραγματοποιήθηκε τη δεκαετία του 1990, και οι πρώτοι πάροχοι υπηρεσιών VOIP προσέφεραν λύσεις αντίστοιχες με αυτές των συμβατικών τηλεφωνικών παρόχων. Ωστόσο στη συνέχεια υπήρξαν πάροχοι που υλοποίησαν κλειστά δίκτυα για ιδιωτικές ομάδες χρηστών και τέλος, άλλοι, που επέτρεψαν τη διασύνδεση μεταξύ χρηστών από διαφορετικές πλατφόρμες παροχής υπηρεσιών VOIP.

Η τεχνολογία αυτή παρουσιάζει μεγάλη δυναμική και πλέον είναι διαθέσιμη πέραν της σταθερής τηλεφωνίας, στους υπολογιστές καθώς και στην κινητή τηλεφωνία μέσω ειδικών εφαρμογών, ενώ υποστηρίζεται πλήρως και από το πρότυπο 4G. Παρέχει δυνατότητες πραγματοποίησης κλήσεων, αποστολής πολυμέσων και άλλων δεδομένων μέσω ενός ενοποιημένου τηλεπικοινωνιακού συστήματος ενώ είναι και αρκετά οικονομική έως δωρεάν υπό συγκεκριμένες συνθήκες.

Πέραν των παραπάνω όμως, έχει διεισδύσει και στον πυρήνα των τηλεπικοινωνιών καθώς σε πολλούς παρόχους τηλεπικοινωνιακών υπηρεσιών, ένα τμήμα της μετάδοσης της πληροφορίας που περνάει από το κεντρικό της δίκτυο (backhaul σύστημα) στηρίζεται στην VOIP τεχνολογία.

Τι είναι όμως το IP;

Το IP είναι πρωτόκολλο που χρησιμοποιείται για διασύνδεση ηλεκτρονικών υπολογιστών που μπορούν να ανήκουν στο ίδιο ή σε διαφορετικά δίκτυα. Η μετάδοση στο IP γίνεται με την τεχνική της μεταγωγής πακέτων. Το κάθε πακέτο του IP φθάνει στον παραλήπτη διασχίζοντας ένα ή περισσότερα διασυνδεδεμένα δίκτυα IP, χωρίς να εξαρτάται από άλλα προηγούμενα ή επόμενα πακέτα διατηρώντας έτσι την αυτονομία του μέσα στο δίκτυο. Λόγω του ότι δεν απαιτεί την εγκαθίδρυση σύνδεσης μεταξύ δύο σημείων πριν την ανταλλαγή δεδομένων, χαρακτηρίζεται και ως επικοινωνία χωρίς σύνδεση (connectionless communication).

Το IP επομένως ασχολείται με την διευθυνσιοδότηση, τον κατακερματισμό (fragmentation) μεγάλων πακέτων και την επανασυγκόλληση τους και λειτουργεί ως εξής:

- Παραλαμβάνει τα δεδομένα σε πακέτα με μέγιστο μέγεθος 64 Kbyte.
- Διαιρεί το κάθε πακέτο σε περισσότερα τμήματα (fragments) και τα μεταδίδει μέσω του δικτύου.
- Επανασυγκολλεί τα IP πακέτα μόλις αυτά φτάσουν στον τελικό παραλήπτη.

Στο χειρισμό του πρωτοκόλλου IP συμμετέχουν μόνο οι δύο ακραίοι υπολογιστικοί σταθμοί και οι ενδιάμεσοι δρομολογητές. Την δρομολόγηση του IP πρωτοκόλλου αναλαμβάνουν οι δρομολογητές οι οποίοι γνωρίζουν την τοπολογία του δικτύου και διαθέτουν κατάλληλους πίνακες δρομολόγησης. Έτσι οι χρήστες αρκεί να γνωρίζουν μόνο την τελική διεύθυνση του αποδέκτη ώστε να δρομολογηθεί κατάλληλα το μήνυμά τους.

Κεφάλαιο 4

Σύγχρονη Κρυπτογραφία

Στο κεφάλαιο αυτό θα γίνει μια σύντομη αναφορά στα βασικά χαρακτηριστικά των ασφαλών επικοινωνιών ώστε να κατανοήσουμε τις ανάγκες που καλείται να καλύψει η κρυπτογραφία. Παρουσιάζουμε βασικές έννοιες της κρυπτογραφίας και τον τρόπο που αυτή λειτουργεί «διαισθητικά». Όλα αυτά θα μας φανούν χρήσιμα καθώς στη συνέχεια του κεφαλαίου παρουσιάζουμε τις ευρύτερες κατηγορίες στις οποίες διαχωρίζονται οι αλγόριθμοι κρυπτογράφησης και δίνουμε ορισμένα χαρακτηριστικά παραδείγματα αυτών.

4.1 Εισαγωγή στην Ασφάλεια και την Κρυπτογραφία

Βασικά χαρακτηριστικά των ασφαλών επικοινωνιών είναι η εμπιστευτικότητα, η αξιοπιστία, η αυθεντικοποίηση και η μη απάρνηση.

1. Εμπιστευτικότητα είναι η διατήρηση της πληροφορίας μακριά από μη εξουσιοδοτημένους χρήστες.
2. Αξιοπιστία είναι η προστασία από μη εξουσιοδοτημένη αλλαγή των δεδομένων.
3. Αυθεντικοποίηση είναι η εξασφάλιση της ακρίβειας της πληροφορίας και των εχόντων πρόσβαση σε αυτή.
4. Μη απάρνηση είναι η αδυναμία να ισχυριστεί κάποιος ότι δεν πραγματοποίησε μια ενέργεια που όντως πραγματοποίησε.

Σκοπός της κρυπτογραφίας και των αλγορίθμων που εφαρμόζονται είναι να εξασφαλίσουν -μεταξύ άλλων- τα παραπάνω χαρακτηριστικά στο βαθμό που αυτό είναι εφικτό.

Κρυπτογράφηση είναι η διαδικασία κωδικοποίησης ενός μηνύματος μέσω ενός κρυπτογραφικού αλγορίθμου, με τέτοιον τρόπο ώστε να είναι αδύνατη η πρόσβαση σε αυτό από μη εξουσιοδοτημένα μέρη. Με αυτόν τον τρόπο δεν διασφαλίζεται ότι δεν θα υπάρξουν διαρροές, ωστόσο αν αυτές συμβούν, το περιεχόμενο του μηνύματος δεν θα είναι κατανοητό.

Αποκρυπτογράφηση είναι η διαδικασία ανάκτησης του αρχικού μηνύματος (plaintext) από μια μη αναγνώσιμη μορφή που ήταν προϊόν της κρυπτογράφησης.

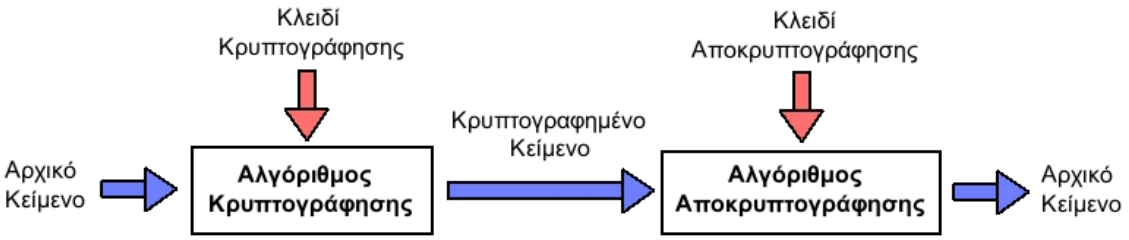
Κρυπτογραφικός αλγόριθμος είναι μία μέθοδος (μαθηματική συνάρτηση) μετασχηματισμού των δεδομένων σε μία μορφή που δεν επιτρέπει σε μη εξουσιοδοτημένα μέρη να κατανοήσουν το περιεχόμενό τους.

Κρυπτοσύστημα ονομάζεται ένα σύνολο αλγορίθμων που είναι αναγκαίοι προκειμένου να υλοποιήσουμε μια συγκεκριμένη απαίτηση ασφαλείας π.χ. την εμπιστευτικότητα. Τυπικά αποτελείται από 3 αλγόριθμους, τη γεννήτρια κλειδιού, τον αλγόριθμο κρυπτογράφησης και τον αλγόριθμο αποκρυπτογράφησης. Αυστηρώς διατυπωμένο (και περιορισμένου στην περίπτωση συμμετρικού κλειδιού όπως θα αναλύσουμε παρακάτω), το κρυπτοσύστημα είναι μια διατεταγμένη 5-άδα (P, C, K, E, D) με τις ακόλουθες ιδιότητες:

1. R είναι ο χώρος όλων των δυνατών μηνυμάτων
2. C είναι ο χώρος όλων των δυνατών κρυπτοκειμένων
3. K είναι ο χώρος όλων των δυνατών κλειδιών
4. $E = \{E_k : k \in K\}$ είναι το σύνολο των συναρτήσεων κρυπτογράφησης $\{E_k : P \to C\}$
5. $D = \{D_k : k \in K\}$ είναι το σύνολο των συναρτήσεων αποκρυπτογράφησης $\{D_k : C \to P\}$

Σε ένα κρυπτοσύστημα, το αρχικό μήνυμα (plaintext) κρυπτογραφείται χρησιμοποιώντας έναν αλγόριθμο κρυπτογράφησης (cipher) και ένα κλειδί κρυπτογράφησης (encryption key). Έτσι, παράγεται το κρυπτόγραμμα (ciphertext), το οποίο μπορεί να διαβαστεί μόνο εφόσον αποκρυπτογραφηθεί. Ένας εξουσιοδοτημένος παραλήπτης μπορεί εύκολα να αποκρυπτογραφήσει το μήνυμα με τη χρήση του κλειδιού αποκρυπτογράφησης (decryption key).

Στη γλώσσα της κρυπτογραφίας έχουν καθιερωθεί κάποιοι όροι που παρουσιάζουν "σχηματικά" το πρόβλημα που καλούμαστε να αντιμετωπίσουμε. Έτσι, ο αποστολέας ενός μηνύματος έχει καθιερωθεί να φέρει τον όρο Alice, ενώ ο παραλήπτης τον όρο Bob. Οι δυο τους επικοινωνούν μέσω ενός ανασφαλούς καναλιού επικοινωνίας. Το κακόβουλο μέρος καλείται Eve και θεωρούμε ότι έχει πρόσβαση στο παραπάνω κανάλι επικοινωνίας. Χρησιμοποιώντας αυτούς τους όρους, κρυπτογράφηση είναι η διαδικασία με την οποία η Alice με τη βοήθεια του κλειδιού κρυπτογράφησης (encryption key) μετατρέπει το αρχικό μήνυμα (plaintext) που έχει ως παραλήπτη τον Bob, σε τέτοια μορφή (ciphertext), ώστε αν η Eve αποκτήσει πρόσβαση σε αυτό να μη μπορέσει να το κατανοήσει. Αποκρυπτογράφηση είναι η διαδικασία με την οποία ο Bob μετατρέπει το κρυπτομήνυμα (ciphertext) που παρέλαβε, στο αρχικό μήνυμα (plaintext) με τη βοήθεια του κλειδιού αποκρυπτογράφησης (decryption key).



Εικόνα 4.1: Προστασία της εμπιστευτικότητας ενός κειμένου

Κρυπτανάλυση είναι η διαδικασία αποκρυπτογράφησης ενός μηνύματος από μη εξουσιοδοτημένα μέρη, τους κρυπταναλυτές. Η διαδικασία αυτή στηρίζεται στην προσπάθεια εύρεσης αδυναμιών στο κρυπτοσύστημα που χρησιμοποιείται, ώστε να καταστεί εφικτή η αποκρυπτογράφηση του μηνύματος χωρίς να είναι εξαρχής γνωστό το κλειδί. Η εύρεση τέτοιων αδυναμιών πάντως δεν θα πρέπει να θεωρείται δεδομένη, καθώς όπως ήδη αναφέρθηκε υπάρχει τουλάχιστον ένα κρυπτοσύστημα που παρέχει τέλεια ασφάλεια (το one–time–pad, κάτω από συγκεκριμένες προϋποθέσεις). Ακόμα όμως και αν υπάρχουν αδυναμίες στο κρυπτοσύστημα, κάθε αξιόπιστος αλγόριθμος κρυπτογράφησης οφείλει να απαιτεί εξαιρετικά σημαντική υπολογιστική ισχύ για να κρυπταναλυθεί. Έτσι, προκύπτει η έννοια της πρακτικής ασφάλειας που παρέχουν οι παραπάνω αλγόριθμοι έναντι της τέλειας ασφάλειας του one-time-pad. Με τον όρο αυτό περιγράφουμε τους αλγορίθμους για τους οποίους ισχύει ότι παράγοντας εργασίας (work factor κατά τον Shannon) για την επιτυχή κρυπτανάλυσή τους είναι πέρα από τις δυνάμεις των αντιπάλων.

Η κρυπτανάλυση μπορεί να κατηγοριοποιηθεί με κριτήριο την ενεργητική ή παθητική συμπεριφορά του κρυπταναλυτή σε:

- Παθητική, όταν ο επιτιθέμενος μόνο παρακολουθεί την επικοινωνία και προσπαθεί να διασπάσει την εμπιστευτικότητα.
- Ενεργητική, όταν ο επιτιθέμενος μπορεί επίσης να προσθαφαιρέσει/ τροποποιήσει τα μηνύματα. Σε αυτήν την περίπτωση προσπαθεί επίσης να διασπάσει και άλλα χαρακτηριστικά ασφαλείας εκτός της εμπιστευτικότητας.

Όπως έχουμε ήδη συνειδητοποιήσει, το κλειδί κρυπτογράφησης αποτελεί βασικό στοιχείο στη διεργασία της κρυπτογράφησης. Η απλούστερη επίθεση για την ανάκτηση του κλειδιού είναι η αποκαλούμενη επίθεση εκτενούς αναζήτησης (exhaustive search attack), γνωστή και ως επίθεση ωμής βίας (brute force attack). Στο συγκεκριμένο τύπο επίθεσης εάν ο κρυπταναλυτής καταφέρει να αποκτήσει ένα ζεύγος αρχικού και κρυπτογραφημένου κειμένου μπορεί με διαδοχική δοκιμή κλειδιών να ανακτήσει το κλειδί που χρησιμοποιήθηκε. Για την αντιμετώπιση των συγκεκριμένων επιθέσεων το κλειδί πρέπει να είναι αρκετά μεγάλο. Τα κλειδιά είναι ουσιαστικά σειρές από bits και ως εκ τούτου η απαίτηση για μεγάλο κλειδί αφορά την χρήση πολλών bits. Γενικώς, όσο αυξάνεται η υπολογιστική δυνατότητα των νέων ηλεκτρονικών υπολογιστών τόσο πιο εξελιγμένοι κρυπτογραφικοί αλγόριθμοι αναπτύσσονται που μπορούν να χρησιμοποιήσουν κλειδιά μεγαλύτερου μήκους.

Λόγω της μη πρακτικής φύσης της επίθεσης εκτενούς αναζήτησης, οι κρυπταναλυτές χρησιμοποιούν κυρίως ένα πλήθος από άλλου τύπου επιθέσεις οι οποίες κατηγοριοποιούνται ανάλογα με το τι δυνατότητες και γνώσεις θεωρούμε ότι έχει στη διάθεσή του ο επιτιθέμενος. Οι βασικότεροι τύποι επιθέσεων, οι οποίοι μπορούν να χρησιμοποιηθούν ενάντια στην αυθεντικότητα, την αυθεντικοποίηση και την αξιοπιστία των συμμετρικών κρυπταλγορίθμων είναι οι εξής:

i) Ciphertext only, όπου ο επιτιθέμενος έχοντας πρόσβαση μόνο στο κρυπτοκείμενο προσπαθεί να ανακαλύψει το κλειδί αποκρυπτογράφησης ή τουλάχιστον το αντίστοιχο αρχικό κείμενο.

ii) Known plaintext, όπου έχοντας πρόσβαση σε ένα ή περισσότερα κείμενα και στα αντίστοιχα κρυπτοκείμενά τους, προσπαθεί να ανακαλύψει το κλειδί αποκρυπτογράφησης.

iii) Chosen-plaintext , όπου ο επιτιθέμενος μπορεί να επιλέξει το αρχικό κείμενο και να δει το αντίστοιχο κρυπτοκείμενο. (και πάλι προσπαθεί να ανακαλύψει το κλειδί αποκρυπτογράφησης).

iv) Chosen-ciphertext, όπου ο επιτιθέμενος επιλέγει το κρυπτοκείμενο και βλέπει το αντίστοιχο

αρχικό κείμενο.

v) Man in the middle, όπου ο επιτιθέμενος παρεμβάλλεται, έχει πρόσβαση, και τροποποιεί τα δεδομένα που αποστέλλονται.

Από τους παραπάνω τύπους επιθέσεων μόνο οι δύο πρώτοι είναι διαθέσιμοι για παθητική κρυπτανάλυση. Ενδεικτικά παραθέτουμε ορισμένες δημοφιλείς μεθόδους κρυπτανάλυσης σύγχρονων κρυπτοσυστημάτων:

Διαφορική κρυπτανάλυση. Αποτελεί παράδειγμα επίθεσης τύπου chosen-plaintext. Υλοποιείται επιλέγοντας ένα μεγάλο αριθμό από ζευγάρια αρχικών κειμένων με προκαθορισμένες διαφορές. Η ανάλυση γίνεται μελετώντας τις αντίστοιχες διαφορές στα κρυπτοκείμενα.

Γραμμική κρυπτανάλυση. Η επίθεση αυτή, τύπου known-plaintext, έχει εφαρμοστεί επιτυχημένα σε πολλούς αλγορίθμους. Βασίζεται στην ανάλυση της συσχέτισης μεταξύ αρχικών και κρυπτογραφημένων bits.

Κλειδοσχεσιακή κρυπτανάλυση (related-key attack). Είναι μία μέθοδος επίθεσης όπου ο επιτιθέμενος μπορεί να καταφέρει να αλλάξει το κλειδί με τέτοιο τρόπο ώστε η σχέση μεταξύ παλιού και νέου κλειδιού να είναι προκαθορισμένη και επιλεγμένη από τον ίδιο. Η συγκεκριμένη μέθοδος επίθεσης είναι υπερβολικά αισιόδοξη προς όφελος του επιτιθέμενου, και συχνά χρησιμοποιείται σαν θεωρητικό εργαλείο για την ανάλυση των κρυπταλγορίθμων και της δομής τους.

Για το τέλος αφήσαμε μία μέθοδο επίθεσης που ξεφεύγει από τη στενή έννοια των προηγούμενων επιθέσεων καθώς είναι σχετική με πρακτικές εφαρμογές της κρυπτογραφίας, την **επίθεση παράπλευρου καναλιού (side channel attack)**. Σε αυτήν, ο επιτιθέμενος μπορεί να αποκομίσει πληροφορίες σχετικές με τη φυσική υλοποίηση του κρυπτοσυστήματος. Για παράδειγμα, ο επιτιθέμενος θα μπορούσε να μετρήσει το χρόνο και την κατανάλωση ενέργειας κατά την εκτέλεση του κρυπτογραφικού αλγορίθμου και να αποκομίσει χρήσιμες πληροφορίες για το κλειδί, πιθανώς χρησιμοποιώντας στατιστικές μεθόδους.

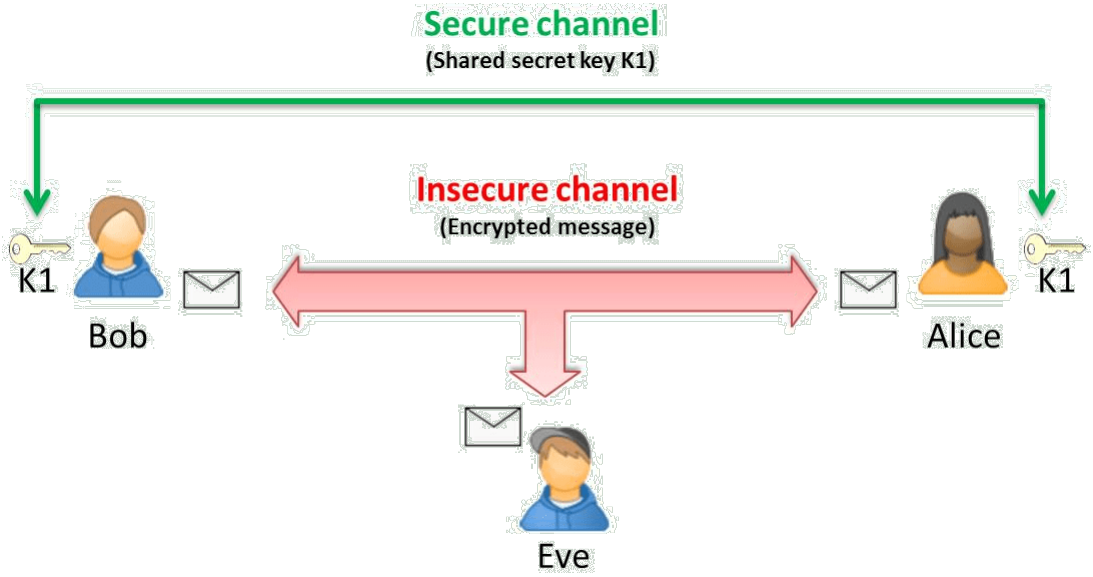
Οι αλγόριθμοι κρυπτογράφησης που βασίζουν την αποτελεσματικότητά τους στη διατήρηση της μυστικότητάς τους ονομάζονται **περιορισμένοι (restricted)** αλγόριθμοι κρυπτογράφησης. Παρέχουν χαμηλό επίπεδο ασφάλειας και σε περίπτωση που ο κώδικας διαρρεύσει θα πρέπει να υπάρξουν τροποποιήσεις σε αυτόν. Στις σύγχρονες εφαρμογές δεν χρησιμοποιούνται τέτοιου είδους αλγόριθμοι κρυπτογράφησης, καθώς όπως αναφέρθηκε στην εισαγωγή η ασφάλεια της κρυπτογράφησης έγκειται πλέον μόνο στην ικανότητά μας να διατηρήσουμε το κλειδί μυστικό· ο αλγόριθμος είναι δημοσιευμένος και δοκιμασμένος για την αντοχή του στις διάφορες γνωστές μεθόδους κρυπτανάλυσης.

Οι σύγχρονοι αλγόριθμοι κρυπτογράφησης κατατάσσονται σε δύο κατηγορίες:
i) Συμμετρικοί αλγόριθμοι
ii) Ασύμμετροι Αλγόριθμοι

Η διαφορά τους έγκειται στον τρόπο με τον οποίο ορίζουν και χρησιμοποιούν τα κλειδιά κρυπτογράφησης και αποκρυπτογράφησης. Περισσότερες πληροφορίες καθώς και υποκατηγορίες αυτών των αλγορίθμων παρουσιάζονται στις ενότητες που ακολουθούν.

4.2 Συμμετρική Κρυπτογραφία

Συναντάται και με το όνομα κρυπτογραφία ιδιωτικού κλειδιού ή ενός κλειδιού. Ήταν η μόνη ευρέως γνωστή μέθοδος κρυπτογραφίας μέχρι και το 1976. Σε αυτήν την κατηγορία εντάσσονται οι αλγόριθμοι στους οποίους το κλειδί αποκρυπτογράφησης ταυτίζεται με το κλειδί κρυπτογράφησης. Η Alice και ο Bob θα πρέπει επομένως να έχουν συμφωνήσει το κλειδί που θα χρησιμοποιήσουν για την ασφαλή μετάδοση του μηνύματος. Η Alice κρυπτογραφεί το αρχικό μήνυμα με χρήση του κλειδιού αυτού και στέλνει το κρυπτομήνυμα (ciphertext) στον Bob. Ο Bob αποκρυπτογραφεί το κρυπτομήνυμα με χρήση του ίδιου κλειδιού και λαμβάνει το αρχικό μήνυμα (plaintext).



Εικόνα 4.2: Συμμετρικό κρυπτοσύστημα

Η αναγκαιότητα για εκ των προτέρων συμφωνία του κλειδιού που θα χρησιμοποιηθεί αποτελεί αδιαμφισβήτητα μία εγγενή αδυναμία σε αυτού του τύπου τους αλγορίθμους καθώς απαιτείται η ύπαρξη ενός διαφορετικού και ασφαλούς καναλιού επικοινωνίας. Ωστόσο, το εμπόδιο αυτό έχει εν μέρει αρθεί χάρη σε τεχνικές που εμπλέκουν το πρωτόκολλο δημοσίου κλειδιού, όπως θα δούμε παρακάτω. Ένα άλλο αρνητικό σημείο των συμμετρικών αλγορίθμων είναι το πλήθος των κλειδιών που απαιτούνται για διμερή επικοινωνία σε μια ομάδα N ατόμων, το οποίο είναι της τάξης $O(N^2)$. Στα θετικά, αντίθετα, μπορούμε να εντάξουμε την πολύ ταχύτερη ολοκλήρωση της διαδικασίας σε σχέση με τους ασύμμετρους αλγορίθμους κρυπτογράφησης, όπως θα δούμε παρακάτω.

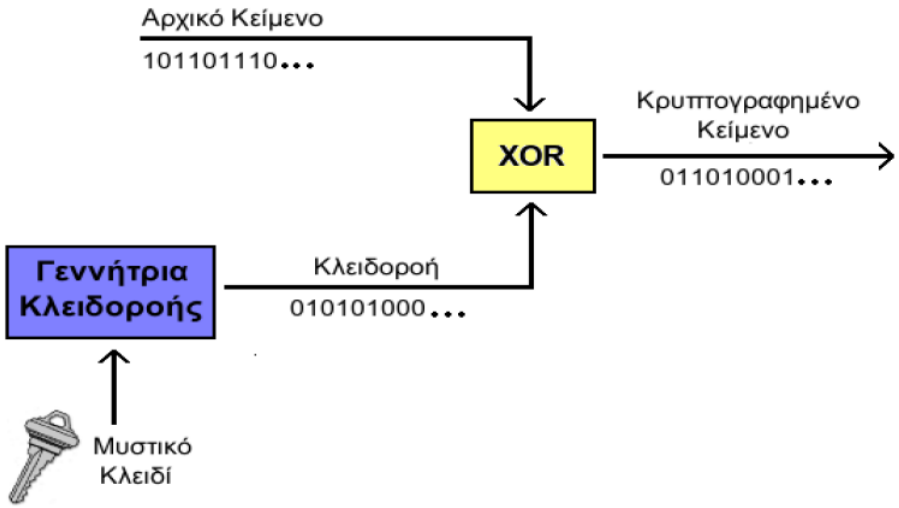
Οι συμμετρικοί κρυπτογραφικοί αλγόριθμοι συνήθως εδράζονται στη δυσκολία επίλυσης μη γραμμικών Boolean εξισώσεων. Ανάλογα με την ποσότητα της πληροφορίας που κρυπτογραφείται και αποστέλλεται κάθε φορά, οι συμμετρικοί αλγόριθμοι κρυπτογράφησης διακρίνονται περαιτέρω σε δύο κατηγορίες, τους αλγόριθμους ροής και τους αλγόριθμους δέσμης ή πακέτου.

Υπάρχει μια ακόμη κατηγορία συμμετρικών αλγορίθμων που χρησιμοποιείται ευρέως στις μέρες μας, που περιλαμβάνει τις λεγόμενες συναρτήσεις σύνοψης (hash functions). Σε αντίθεση με τις δύο προηγούμενες κατηγορίες, η συγκεκριμένη είναι μονόδρομη, δηλαδή δεν υπάρχει δυνατότητα λήψης του αρχικού μηνύματος από το κρυπτοκείμενο. Αυτού του είδους οι συναρτήσεις χρησιμοποιούνται για να επιβεβαιώσουν την αξιοπιστία/αυθεντικότητα των δεδομένων που λήφθηκαν από μια αναξιόπιστη πηγή ή για να προσθέσουν ένα επίπεδο ελέγχου και ασφαλείας.

Οι συμμετρικοί αλγόριθμοι κρυπτογράφησης είναι επιρρεπείς στις τεχνικές κρυπτανάλυσης που προαναφέρθηκαν. Για το λόγο αυτό θα πρέπει η κατασκευή τους να γίνεται με προσοχή ώστε κατά το δυνατόν να προλαμβάνουν αυτές τις κρυπταναλυτικές τεχνικές.

4.2.1 Αλγόριθμοι Ροής

Οι κρυπτογραφικοί αλγόριθμοι ροής (Stream ciphers) χρησιμοποιούνται για την κρυπτογράφηση μίας συνεχούς ροής δεδομένων (data stream). Για την κρυπτογράφηση επιλέγεται αρχικά μία γεννήτρια κλειδοροής (keystream generator), η οποία δέχεται ως είσοδο το μυστικό κλειδί και παράγει στην έξοδό της μία ψευδοτυχαία ακολουθία bits, η οποία ονομάζεται κλειδοροή (keystream). Στην συνέχεια εφαρμόζεται η συνάρτηση XOR ανάμεσα στο αρχικό κείμενο και στην κλειδοροή και το αποτέλεσμα της συνάρτησης είναι η τελική κρυπτογραφημένη ροή δεδομένων. Η παραπάνω διαδικασία φαίνεται στο σχήμα που ακολουθεί.



Εικόνα 4.3: Η λειτουργία του συμμετρικού αλγορίθμου ροής

Πρακτικά, η γεννήτρια κλειδοροής έχει στην βάση της μία Boolean γραμμική συνάρτηση η οποία παρέχει μέγιστη περίοδο. Αυτό εξασφαλίζεται με την συμμετοχή πρώτων πολυωνύμων στην δομή της κλειδοροής. Το γεγονός ότι τα πρώτα πολυώνυμα είναι γνωστά αφού έχουν υπολογιστεί από πριν, προκαλεί σημαντικό μειονέκτημα ως προς την κρυπτανάλυση. Ωστόσο, έχουν αναπτυχθεί νέες τεχνολογίες οι οποίες αποφεύγουν την χρήση πρώτων πολυωνύμων και βασίζονται σε μη γραμμικές Boolean συναρτήσεις.

Η αποκρυπτογράφηση γίνεται με την αντίστροφη ακριβώς διαδικασία. Εάν χρησιμοποιηθεί το ίδιο κλειδί ως είσοδος στην γεννήτρια κλειδοροής, τότε αυτή θα παραγάγει ακριβώς την ίδια ακολουθία bits (κλειδοροή) σε σχέση με τη διαδικασία της κρυπτογράφησης. Εφαρμόζοντας τη συνάρτηση XOR ανάμεσα στην κρυπτογραφημένη ακολουθία δεδομένων και την κλειδοροή προκύπτει το αρχικό κείμενο.

Για να είναι ασφαλής ο κρυπτογραφικός αλγόριθμος ροής, θα πρέπει να πληρούνται ορισμένες προϋποθέσεις όσον αφορά την γεννήτρια κλειδοροής και την ψευδοτυχαία ακολουθία bits που αυτή παράγει. Συγκεκριμένα η ασφάλεια του αλγορίθμου εξαρτάται από τις εξής κρυπτογραφικές ιδιότητες:

1. Η ψευδοτυχαία ακολουθία bits (κλειδοροή) που παράγεται από την γεννήτρια κλειδοροής θα πρέπει να έχει αρκετά μεγάλη περίοδο επανάληψης. Επειδή ουσιαστικά η γεννήτρια κλειδοροής είναι μία μαθηματική συνάρτηση που δέχεται ως είσοδο το μυστικό κλειδί και παράγει στην έξοδο την κλειδοροή, είναι βέβαιο πως η κλειδοροή που θα παραχθεί θα είναι από ένα σημείο και μετά περιοδική. Αυτό σημαίνει πως μετά από κάποιον αριθμό bits της κλειδοροής, αυτή θα επαναλαμβάνεται ξεκινώντας από την αρχή. Αν η περίοδος επανάληψης είναι πολύ μικρή, τότε το γεγονός αυτό καθιστά τον αλγόριθμο κρυπτογράφησης ιδιαίτερα ευάλωτο σε προσπάθειες κρυπτανάλυσης.

2. Η ακολουθία bits της κλειδοροής θα πρέπει να μοιάζει πολύ με τυχαία. Αυτό σημαίνει ότι η μαθηματική συνάρτηση που χρησιμοποιείται στην γεννήτρια κλειδοροής θα πρέπει να επιλεγεί κατάλληλα ούτως ώστε το αποτέλεσμά της να πλησιάζει όσο το δυνατόν περισσότερο το τυχαίο. Υπάρχουν ειδικές μέθοδοι δοκιμής της καταλληλότητας της γεννήτριας κλειδοροής, οι οποίοι

εκπονούν ελέγχους τυχαιότητας (randomness tests) σε αυτήν. Ένας έλεγχος τυχαιότητας είναι για παράδειγμα ο εξής: Για μία κλειδοροή μήκους εκατομμυρίων bits, θα πρέπει ο αριθμός των 1 να ισούται με τον αριθμό των 0. Επίσης θα πρέπει κάθε 0 να ακολουθείται από 1 τόσο συχνά όσο και το αντίστροφο. Σε κάθε περίπτωση όμως η κλειδοροή που θα παραχθεί δεν μπορεί να είναι εντελώς τυχαία, γι' αυτό και ονομάζεται ψευδοτυχαία ακολουθία bits.

3. Η κλειδοροή θα πρέπει να έχει μεγάλη γραμμική ισοδυναμία (linear equivalence). Οποιαδήποτε ακολουθία δυαδικών ψηφίων μπορεί να παραχθεί με χρήση γραμμικών μεθόδων, για παράδειγμα με υπολογισμό της επόμενης τιμής βάσει των προηγούμενων τιμών της ακολουθίας. Αν στον υπολογισμό αυτό χρησιμοποιείται ένας μικρός σχετικά αριθμός προηγούμενων τιμών, τότε λέμε πως η ακολουθία έχει μικρή γραμμική ισοδυναμία. Αντίθετα εάν στο υπολογισμό χρησιμοποιείται ένας μεγάλος αριθμός προηγούμενων τιμών, τότε η ακολουθία έχει μεγάλη γραμμική ισοδυναμία. Μία κλειδοροή με μεγάλη γραμμική ισοδυναμία εγγυάται μεγαλύτερη ασφάλεια των κρυπτογραφημένων δεδομένων απέναντι σε προσπάθειες κρυπτανάλυσης.

Οι συνθήκες που παρουσιάστηκαν παραπάνω είναι αναγκαίες για να εξασφαλίσουν έναν αξιόπιστο αλγόριθμο ροής, όχι όμως επαρκείς. Γενικά, για να είναι ένας κρυπτογραφικός αλγόριθμος ροής αξιόπιστος θα πρέπει να εξασφαλίζει ότι ακόμη και εάν κάποιος αποκτήσει οποιαδήποτε πληροφορία για κάποιο κομμάτι της ακολουθίας κλειδοροής, είναι υπολογιστικά αδύνατο να συνάγει άλλα κομμάτια της ακολουθίας.

Η ταχύτητα κρυπτογράφησης με αλγόριθμους ροής, οι μικρές απαιτήσεις μνήμης αλλά και η μηδενική διάδοση σφαλμάτων, καθιστούν τους συγκεκριμένους αλγόριθμους ιδανικούς για τηλεπικοινωνιακές εφαρμογές.

Παρά τα όσα πλεονεκτήματα αναφέρθηκαν, οι αλγόριθμοι ροής αποδεικνύονται ιδιαίτερα ευάλωτοι αν το ίδιο κλειδί χρησιμοποιηθεί δυο φορές, καθώς θα δώσει την ίδια κλειδοροή. Αυτό αποτελεί κίνδυνο για την ασφάλεια και πρέπει να αποφεύγεται όσο γίνεται. Η συνηθισμένη λύση σε αυτό το πρόβλημα παρέχεται όταν αυτός που κάνει την κρυπτογράφηση (encryptor):

- Παράγει ένα τυχαίο κλειδί μηνύματος ή συνόδου πριν να κρυπτογραφήσει το αρχικό κείμενο,
- Με αυτό το κλειδί μετατρέπει το μυστικό κλειδί που εισάγετε στη γεννήτρια κλειδοροής,
- Κατόπιν, στέλνει το κλειδί μηνύματος ως πρόθεμα (prefix) του κρυπτογραφημένου κειμένου.

Ένας από τους πιο δημοφιλείς κρυπταλγορίθμους ροής ήταν ο RC4. Σχεδιάστηκε από τον Ron Rivest και κυκλοφόρησε πρώτη φορά το 1994. Ο αλγόριθμος χρησιμοποιεί κλειδιά μήκους από 40 μέχρι 2048 bits και η λειτουργία του βασίζεται σε μεταθέσεις και πράξεις αποκλειστικής διάζευξης (XOR). Η απλή δομή του καθώς και οι μικρές απαιτήσεις σε μνήμη, τον έκαναν ευρέως διαδεδομένο και έτσι χρησιμοποιήθηκε σε πρωτόκολλα όπως το WEP και το TLS. Παρόλα αυτά, σήμερα δεν προτιμάται καθώς δεν είναι πλέον ασφαλής. Επιπλέον χαρακτηριστικά παραδείγματα αλγορίθμων αυτής της υποκατηγορίας αποτελούν οι αλγόριθμοι ORYX και SEAL. Βασική πηγή έμπνευσης για όλους τους αλγορίθμους αυτής της κατηγορίας αποτελεί ο one-time-pad, ο πρώτος (και μοναδικός μέχρι σήμερα) αλγόριθμος κρυπτογράφησης που παρέχει τέλεια μυστικότητα, υπό την προϋπόθεση τήρησης των αυστηρών προϋποθέσεων που αφορούν στο κλειδί κρυπτογράφησης.

4.2.2 Αλγόριθμοι Δέσμης / Πακέτου

Οι κρυπτογραφικοί αλγόριθμοι δέσμης ή πακέτου είναι αλγόριθμοι που τεμαχίζουν σε πακέτα (blocks) το αρχικό κείμενο που πρόκειται να κρυπτογραφηθεί και κρυπτογραφούν κάθε πακέτο ξεχωριστά. Για να επιτύχουν το παραπάνω, χρησιμοποιούν μία συνάρτηση που δέχεται σαν όρισμα το συμμετρικό κλειδί. Το αποτέλεσμα της διαδικασίας κρυπτογράφησης είναι η παραγωγή ενός κρυπτογραφημένου τμήματος το οποίο στην πλειοψηφία των περιπτώσεων έχει το ίδιο μήκος με το αντίστοιχο τμήμα του αρχικού κειμένου. Η διαδικασία της αποκρυπτογράφησης είναι η αντίστροφη της διαδικασίας κρυπτογράφησης, μόνο που στην περίπτωση αυτή χρησιμοποιείται η συνάρτηση αποκρυπτογράφησης αντί της κρυπτογραφικής συνάρτησης.

Το μέγεθος του κάθε τμήματος θα πρέπει να είναι αρκετά μεγάλο ούτως ώστε να προλαμβάνονται διάφορες επιθέσεις λεξικού (dictionary attacks). Διαφορετικά, μπορεί ο κρυπταναλυτής να εξετάσει ένα ζεύγος καθαρού και αντίστοιχου κρυπτογραφημένου κειμένου και να κατασκευάσει ένα λεξικό που θα αντιστοιχεί κάθε τμήμα κρυπτογραφημένου κειμένου με το αντίστοιχο τμήμα του καθαρού κειμένου. Με βάση το λεξικό αυτό, θα μπορεί στην συνέχεια να αποκρυπτογραφεί κάθε κείμενο που κρυπτογραφείται χρησιμοποιώντας το συγκεκριμένο κλειδί.

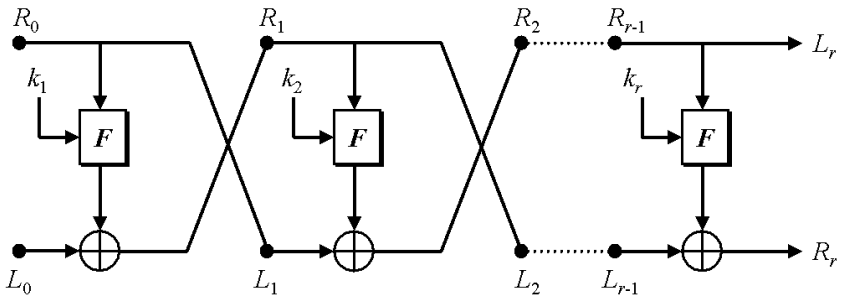
Οι συγκεκριμένοι αλγόριθμοι χρησιμοποιούνται ως δομικό στοιχείο σε πολλά κρυπτογραφικά πρωτόκολλα. Ο σχεδιασμός τους βασίστηκε στις ιδέες των επαναληπτικών αλγορίθμων κρυπτογράφησης δέσμης product, όπως τις παρουσίασε ο Claude Shannon το 1949, όπου τους ανέλυσε και πρότεινε έναν αποτελεσματικό τρόπο συνδυασμού απλών πράξεων όπως αντικαταστάσεις και μεταθέσεις, προκειμένου να επιτευχθούν τα ζητούμενα της σύγχυσης (confusion) και διάχυσης (diffusion) της πληροφορίας.

Οι επαναληπτικοί κρυπτογραφικοί αλγόριθμοι δέσμης product εφαρμόζουν επαναληπτικά μια κυκλική συνάρτηση, η οποία αντιστοιχίζει μια δέσμη n-bit σε μία άλλη δέσμη n-bit. Κάθε εφαρμογή της κυκλικής συνάρτησης ονομάζεται ανακύκλωση (round) και ο αριθμός των ανακυκλώσεων συμβολίζεται με r.

Σε κάθε εφαρμογή της κυκλικής συνάρτησης χρησιμοποιείται ένα υποκλειδί (subkey) $k_i, 1 \leq i \leq r$, το οποίο εξάγεται από το κλειδί k. Για αυτό το λόγο πρέπει να υπάρχει μία προκαθορισμένη μέθοδος που να υπολογίζει όλα τα κυκλικά κλειδιά από το κλειδί. Η μέθοδος αυτή ονομάζεται Προσδιορισμός Κλειδιού (Key Schedule).

Για να γίνει δυνατή η αποκρυπτογράφηση, η κυκλική συνάρτηση πρέπει να παρέχει μία αντίστροφη λειτουργία.

Μία διαδεδομένη υλοποίηση αυτών των αλγορίθμων είναι το δίκτυο Feistel, με χαρακτηριστική υλοποίηση στον κρυπταλγόριθμο DES, που αποτελούσε μέχρι το 2001 το πρότυπο για την ασφάλεια των ηλεκτρονικών δεδομένων στις ΗΠΑ. Σε αυτό (δίκτυο Feistel), το κρυπτογραφούμενο τμήμα χωρίζεται σε δύο μέρη και η κυκλική συνάρτηση εφαρμόζεται μόνο στο ένα. Το αποτέλεσμα γίνεται XOR με το άλλο μέρος και ακολουθεί εναλλαγή των δύο μερών. Ως αποτέλεσμα η κρυπτογράφηση και η αποκρυπτογράφηση παρουσιάζουν μεγάλες ομοιότητες, απαιτώντας (ορισμένες φορές) απλά την αντίστροφη λήψη των κλειδιών, απλοποιώντας έτσι την υλοποίησή του τόσο σε αλγοριθμικό επίπεδο όσο και σε επίπεδο υλισμικού (hardware).



Σχήμα 4.4: Η λειτουργία του δικτύου Feistel

Μία ακόμα διαδεδομένη υλοποίηση επαναληπτικών αλγορίθμων δέσμης είναι τα δίκτυα αντικατάστασης μετάθεσης, με χαρακτηριστικό παράδειγμα τον AES, τον αλγόριθμο κρυπτογράφησης που διαδέχθηκε των DES και τον οποίο θα αναλύσουμε στη συνέχεια.

Η δημοσιοποίηση του κρυπταλγορίθμου DES το 1977 ήταν καθοριστική για την κατανόηση από το ευρύ κοινό των μοντέρνων κρυπτογραφικών αλγορίθμων δέσμης. Επίσης, επηρέασε την ανάπτυξη των κρυπταναλυτικών τεχνικών, καθώς οι τεχνικές της διαφορικής, όσο και της γραμμικής κρυπτανάλυσης

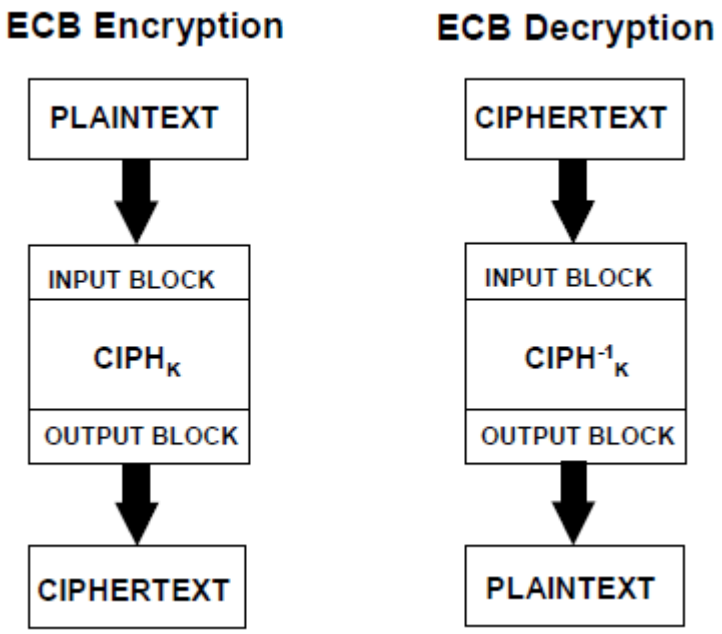
προέκυψαν από μελέτες επί του συγκεκριμένου κρυπταλγορίθμου. Πλέον, υπάρχει μία ευρεία γκάμα από κρυπταναλυτικές τεχνικές έναντι των οποίων θα πρέπει να είναι ασφαλείς οι αλγόριθμοι πακέτου, πέραν της προφανούς ανάγκης να είναι αποτελεσματικοί έναντι επιθέσεων εκτενών αναζητήσεων.

Πάντως, ακόμα και ένας ασφαλής κρυπταλγόριθμος δέσμης είναι κατάλληλος για την κρυπτογράφηση ενός και μόνο μπλοκ, συχνά 64 ή 128 bits, με χρήση ενός σταθερού κλειδιού κάθε φορά. Καθώς όμως τα μηνύματα μπορούν να είναι οποιουδήποτε μήκους, έχουν αναπτυχθεί διάφορες τεχνικές που επιτρέπουν την επαναλαμβανόμενη χρήση του αλγορίθμου κατά ασφαλή τρόπο για να επιτύχουν την εμπιστευτικότητα και την αυθεντικότητα.

Οι κυριότερες από τις τεχνικές αυτές, υπό το γενικότερο όνομα τύποι λειτουργίας αλγορίθμων δέσμης (block cipher mode of operation), παρουσιάζονται εν συντομία παρακάτω. Βασικό χαρακτηριστικό σε όλες τις περιπτώσεις είνα η ομοιότητα μεταξύ του ευθύ και του αντίστροφου μετασχηματισμού.

i) **Electronic Codebook (ECB) / Ηλεκτρονικό Κωδικοβιβλίο**

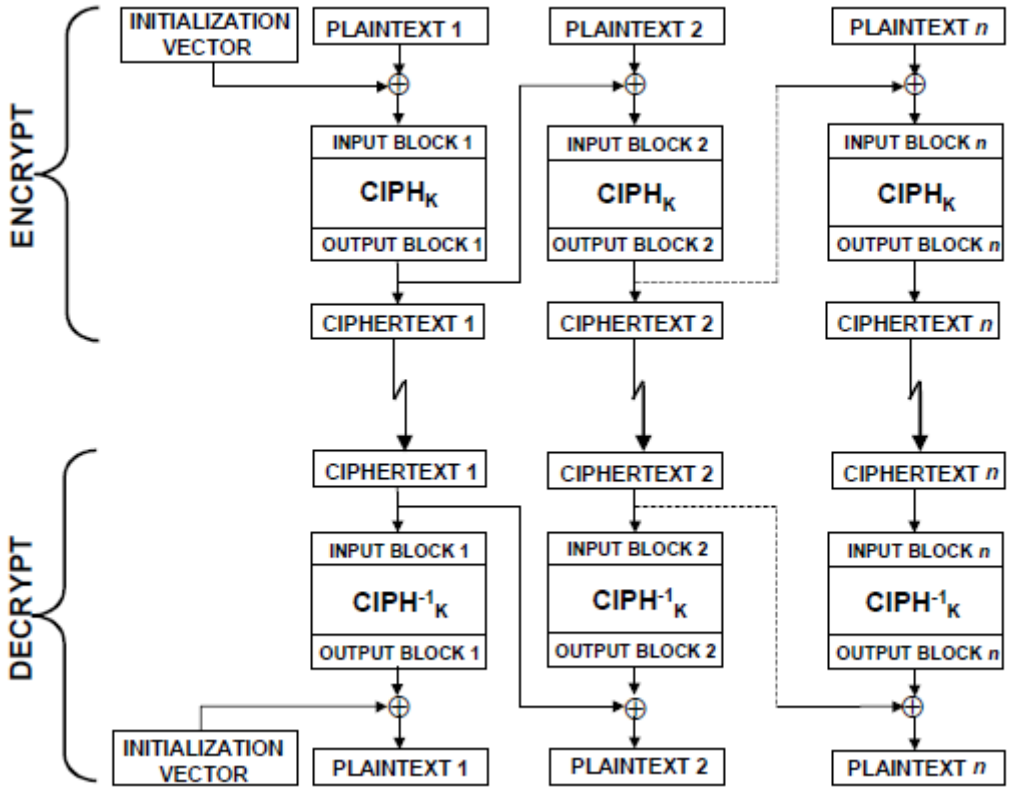
Το κλειδί κρυπτογραφεί το κάθε πακέτο στο αντίστοιχο κρυπτόγραμμα. Δύο ίδια μηνύματα κρυπτογραφημένα με το ίδιο κλειδί, οδηγούν πάντα στα ίδιο κρυπτόγραμμα. Για το λόγο αυτό, δεν συνίσταται η χρήση σε εφαρμογές όπου υπάρχουν επαναλαμβανόμενα μοτίβα δεδομένων στο αρχικό μήνυμα. Από την άλλη, στα θετικά συμπεριλαμβάνεται το γεγονός ότι ένα λάθος στη λήψη επηρεάζει το συγκεκριμένο μπλοκ και μόνον αυτό.



Σχήμα 4.5: Ο τύπος λειτουργίας "Ηλεκτρονικό Κωδικοβιβλίο (ECB)"

ii) **Cipher block chaining (CBC) / Κρυπταλγόριθμος Αλυσιδωτού Τμήματος**

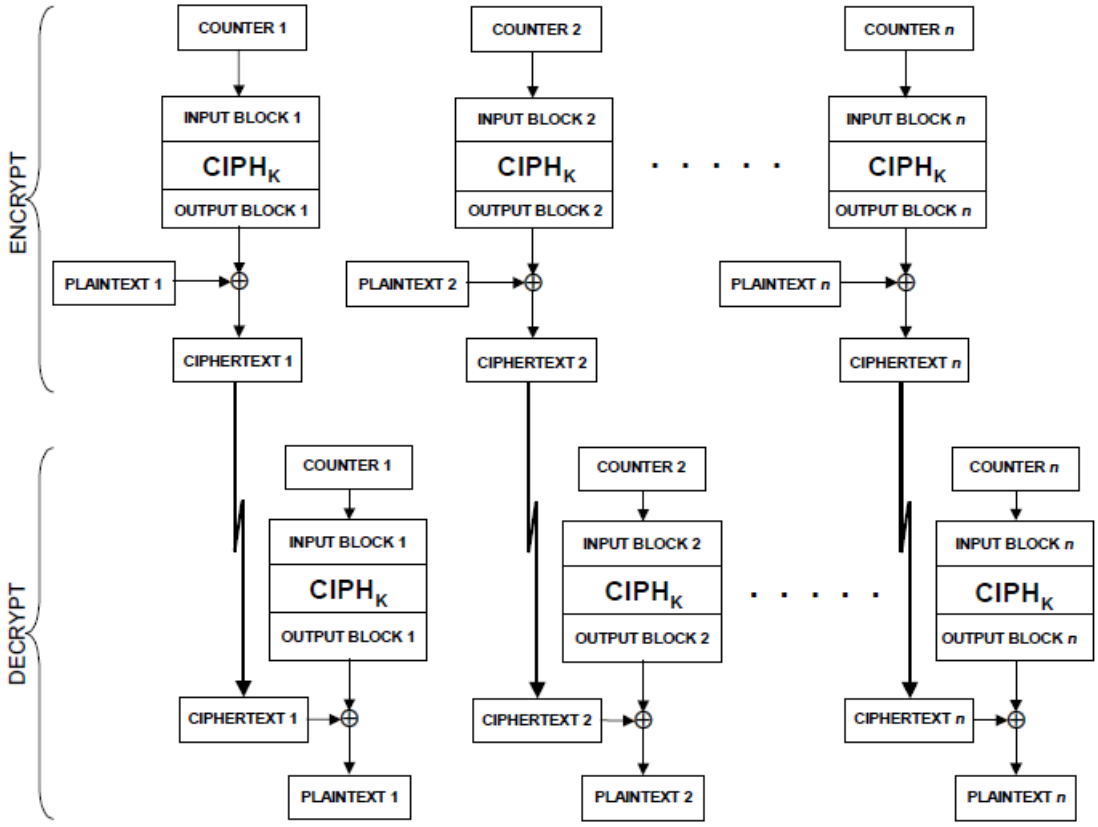
Τα bits από κάθε πακέτο του αρχικού μηνύματος υποβάλλονται σε πράξη αποκλειστικής διάζευξης με τα αντίστοιχα bits του προηγούμενου πακέτου κρυπτογράμματων. Το πακέτο που προκύπτει ως αποτέλεσμα της προηγούμενης πράξης είναι αυτό που εισάγεται στον αλγόριθμο προς κρυπτογράφηση. Συνεπώς το κάθε πακέτο κρυπτογράμματος εξαρτάται από όλα τα προηγούμενα κρυπτογράμματα και επομένως δύο ίδια μοτίβα δεν οδηγούν σε ίδια κρυπτογράμματα. Λάθος κατά τη μετάδοση σε ένα απλό bit του κρυπτογράμματος (είτε από σφάλμα είτε από παρεμβολή επιτιθέμενου), προκαλεί λάθος σε όλα τα επόμενα μπλοκ του αποκρυπτογραφημένου κειμένου. Προκειμένου να είναι δυνατή η αρχικοποίηση του αλγορίθμου χρησιμοποιείται ένα διάνυσμα αρχικοποίησης.



Σχήμα 4.6: Ο τύπος λειτουργίας "Κρυπταλγόριθμος Αλυσιδωτού Τμήματος (CBC)"

iii) **Counter (CTR) / Μετρητής**

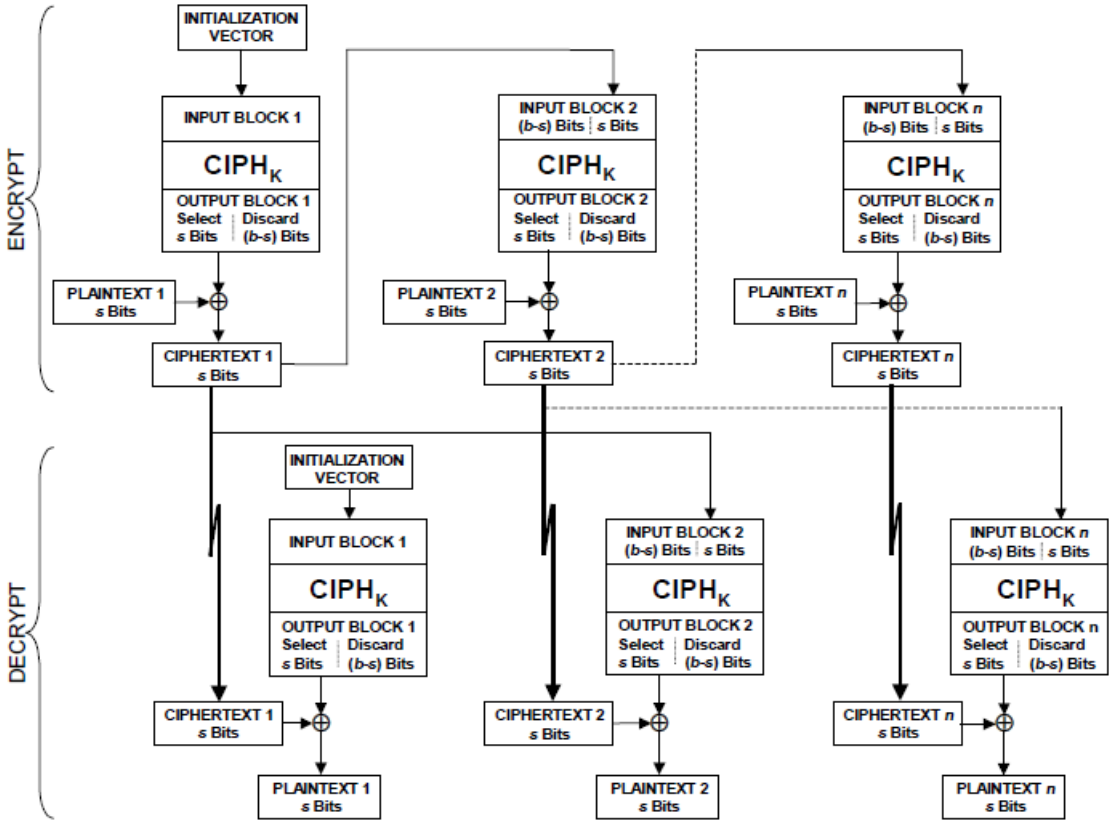
Για την κρυπτογράφηση με αυτόν τον τύπο λειτουργίας είναι απαραίτητη η ύπαρξη ενός συνόλου από πακέτα που ονομάζονται μετρητές. Κάθε μετρητής χρησιμοποιείται σαν είσοδος στον αλγόριθμο κρυπτογράφησης και η έξοδος που προκύπτει υποβάλλεται σε πράξη αποκλειστικής διάζευξης με το αντίστοιχο πακέτο αρχικού μηνύματος. Το αποτέλεσμα της παραπάνω πράξης είναι το αντίστοιχο πακέτο κρυπτογράμματος. Βασική προϋπόθεση για την αποτελεσματική λειτουργία αυτού του τύπου κρυπτογράφησης πακέτου είναι ότι -με δεδομένο το κλειδί κρυπτογράφησης- θα πρέπει όλοι οι μετρητές που χρησιμοποιούνται να είναι διαφορετικοί ανά δύο. Στα θετικά συγκαταλέγεται το γεγονός ότι λάθος κατά τη μετάδοση ενός bit κρυπτογράμματος επηρεάζει μόνο το πακέτο στο οποίο ανήκει.



Σχήμα 4.7: Ο τύπος λειτουργίας "Μετρητής (CTR)"

iv) **Cipher Feedback (CFB), Κρυπταλγόριθμος Ανάδρασης**

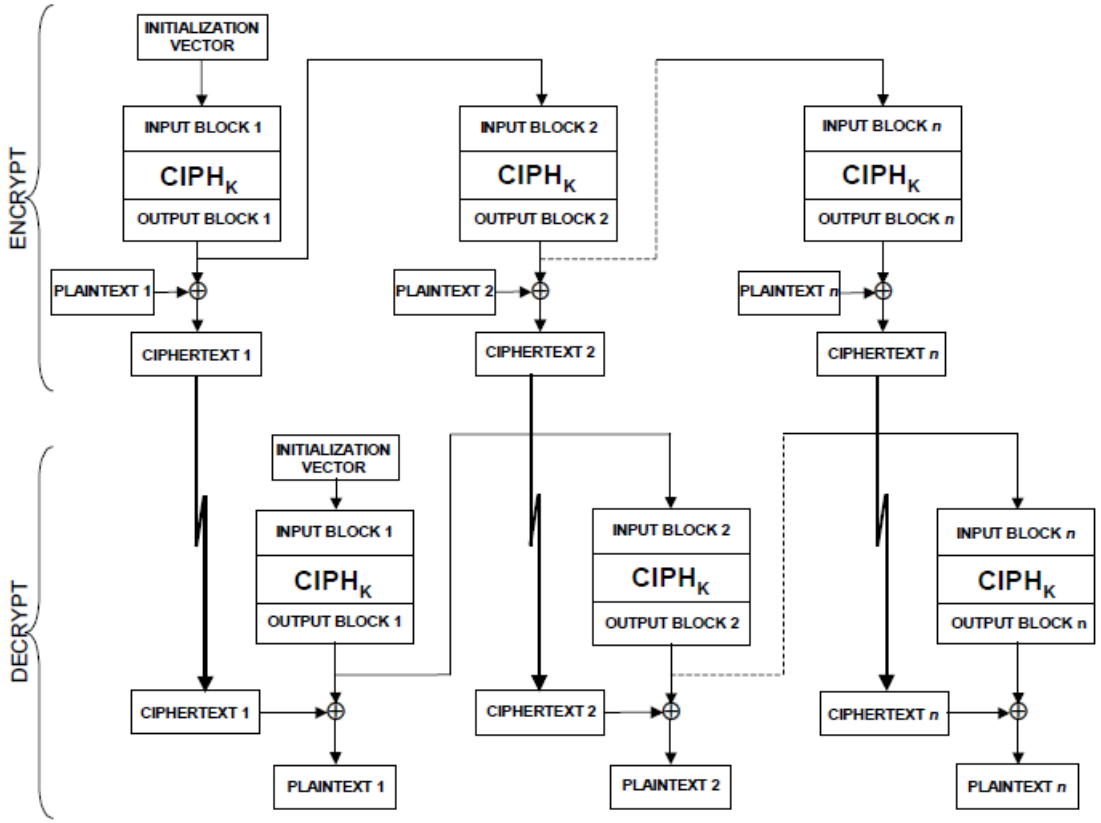
Σε αυτόν τον τύπο λειτουργίας κρυπτογραφούνται s bits τη φορά με τη βοήθεια ενός καταχωρητή ολίσθησης μεγέθους b, όσο δηλαδή και το μέγεθος του πακέτου. Σε κάθε βήμα, τα περιεχόμενα του καταχωρητή ολίσθησης κρυπτογραφούνται και από το αποτέλεσμα που προκύπτει τα s πιο σημαντικά bits γίνονται XOR με τα s bits του αρχικού μηνύματος. Από την παραπάνω πράξη προκύπτουν τα s bits του κρυπτογράμματος. Με τη σειρά του το κρυπτόγραμμα πηγαίνει στον καταχωρητή ολίσθησης που προκειμένου να χωρέσει τα s bits ολισθαίνει κατά s θέσεις αριστερά ώστε τα bits του κρυπτογράμματος να αποθηκευτούν στις s λιγότερο σημαντικές θέσεις. Λάθος σε ένα bit κρυπτογράμματος επηρεάζει το πολύ τις επόμενες [b/s] επαναλήψεις. Για την αρχικοποίηση του αλγορίθμου χρησιμοποιείται και σε αυτήν την περίπτωση ένα διάνυσμα αρχικοποίησης.



Σχήμα 4.8: Ο τύπος λειτουργίας "Κρυπταλγόριθμος Ανάδρασης (CFB)"

v) **Output Feedback (OFB) / Ανάδραση Εξόδου**

Μοιάζει πολύ με αλγόριθμο ροής και είναι σχεδόν ίδιος με τον Κρυπταλγόριθμο Ανάδρασης, αλλά με έναν μηχανισμό για τη μείωση των λαθών . Η έξοδος της συνάρτησης κρυπτογράφησης (και όχι το κρυπτόγραμμα) χρησιμοποιείται στην ανάδραση. Έτσι, ένα σφάλμα στο κρυπτόγραμμα επηρεάζει μόνο μια αποκρυπτογράφηση. Κατάλληλο για εφαρμογές με μεγάλο αριθμό σφαλμάτων λόγω υψηλού βαθμού θορύβου (π.χ. δορυφορικές επικοινωνίες).



Σχήμα 4.9: Ο τύπος λειτουργίας "Ανάδραση Εξόδου (OFB)"

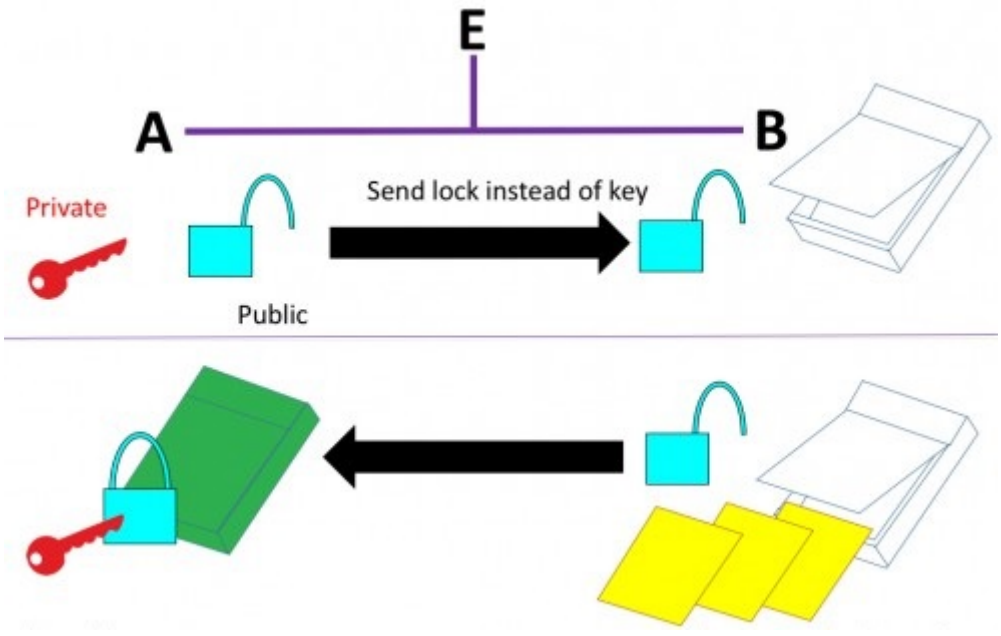
4.3 Ασύμμετρη Κρυπτογραφία

Η Ασύμμετρη Κρυπτογραφία, είναι γνωστή και ως «Κρυπτογραφία Δημοσίου Κλειδιού». Σε αντίθεση με τη συμμετρική κρυπτογραφία, η οποία χρησιμοποιεί μόνο ένα κλειδί, η ασύμμετρη χρησιμοποιεί δύο.

Οι Whitfield Diffie και Martin Hellman, ερευνητές στο Πανεπιστήμιο του Stanford, πρότειναν στη δημοσίευσή τους το 1977, New Directions in Cryptography, τη χρήση της ασύμμετρης κρυπτογραφίας. Ωστόσο, η όλη ιδέα είχε προταθεί από τους James Henry Ellis, Clifford Cocks και Graham Williamson ήδη από το 1972, όσο αυτοί εργάζονταν στον Βρετανικό Οργανισμό Πληροφοριών και Ασφαλείας Government Communications Headquarters, CGHQ). Ο ασύμμετρος αλγόριθμος όπως περιγράφεται στο paper των Diffie - Hellman , χρησιμοποιεί αριθμούς υψωμένους σε συγκεκριμένες δυνάμεις p, ώστε να παραχθούν τα κλειδιά αποκρυπτογράφησης και είναι η πρώτη δημοσιευμένη μέθοδος για την εγκαθίδρυση ενός κοινού μυστικού χωρίς τη χρήση ενός προϋφιστάμενου, μέσω ενός καναλιού επικοινωνίας που πληροί την αρχή της αυθεντικοποίησης αλλά όχι της εμπιστευτικότητας. Η παραπάνω διαδικασία είναι γνωστή και ως αλγόριθμος ανταλλαγής κλειδιού Diffie - Helllman.

Όπως προκύπτει και από την εναλλακτική ονομασία τους, σε αυτούς τους αλγορίθμους υπάρχει ένα δημόσιο κλειδί, υπό την έννοια ότι όλοι μπορούν να έχουν πρόσβαση σε αυτό. Αυτό το κλειδί είναι το κλειδί κρυπτογράφησης (encryption key). Το δεύτερο κλειδί που αποκαλείται ιδιωτικό κλειδί είναι γνωστό μόνο στον παραλήπτη του μηνύματος και παίζει το ρόλο του κλειδιού αποκρυπτογράφησης (decryption key). Έτσι, μόνο ο παραλήπτης μπορεί να αποκρυπτογραφήσει ένα μήνυμα που έχει κρυπτογραφηθεί με το δικό του δημόσιο κλειδί. Τα δύο αυτά κλειδιά είναι απλά πολύ μεγάλοι αριθμοί, του οποίους χρησιμοποιούμε ως ζεύγος με την προφανή προϋπόθεση να είναι διαφορετικοί μεταξύ τους. Παρά το γεγονός ότι τα κλειδιά σχετίζονται μαθηματικά, δεν υπάρχει (αποδοτικός) τρόπος να υπολογιστεί το ιδιωτικό με βάση το δημόσιο.

Τα στάδια της επικοινωνίας μέσω του ασύμμετρου μοντέλου είναι τα ακόλουθα: Ο Bob θα πρέπει να χρησιμοποιήσει τη γεννήτρια κλειδιών του προκειμένου να παραγάγει ένα ζεύγος κλειδιών και έπειτα να ενημερώσει την Alice για το δημόσιο κλειδί του. Οταν η Alice θελήσει να στείλει ένα μήνυμα στον Bob τότε δεν έχει παρά να κρυπτογραφήσει το μήνυμα με το δημόσιο κλειδί του Bob και να του αποστείλει το κρυπτοκείμενο. Ο Bob, χρησιμοποιώντας το ιδιωτικό του κλειδί θα αποκρυπτογραφήσει το κρυπτόγραμμα και θα διαβάσει το αρχικό κείμενο. Για να γίνει ακόμα πιο κατανοητή η όλη διαδικασία, θα μπορούσαμε να παρομοιάσουμε το δημόσιο κλειδί με μία ανοιχτή κλειδαριά που ο ιδιοκτήτης της διαθέτει σε όποιον του τη ζητήσει. Το ιδιωτικό κλειδί είναι το μόνο που μπορεί να ξεκλειδώσει αυτήν την κλειδαριά και προφανώς παραμένει αποκλειστικά στην κατοχή του ιδιοκτήτη του.



Εικόνα 4.10: Κρυπτοσύστημα δημοσίου κλειδιού

Όπως είναι εύκολα κατανοητό, η συγκεκριμένη μέθοδος κρυπτογράφησης αντιμετωπίζει και τα δύο αρνητικά των συμμετρικών αλγορίθμων αφού πρώτον, δεν χρειάζεται να έχει προηγηθεί ανταλλαγή κλειδιών μέσω ασφαλούς διαύλου και δεύτερον το πλήθος των κλειδιών σε μια ομάδα N ατόμων είναι $O(N)$ έναντι $O(N^2)$ στους συμμετρικούς αλγορίθμους. Από την άλλη πλευρά, υπάρχει σημαντική υστέρηση όσον αφορά στο χρόνο ολοκλήρωσης της διαδικασίας έναντι των συμμετρικών αλγορίθμων.

Για να είναι δυνατόν η ασύμμετρη κρυπτογράφηση να αποδώσει σε θέματα που αφορούν την εμπιστευτικότητα, την εγκυρότητα, την αυθεντικότητα και τη μη απάρνηση, οι χρήστες και τα συστήματα πρέπει να είναι σίγουροι ότι ένα δημόσιο κλειδί είναι αυθεντικό και ότι ανήκει στο πρόσωπο ή την οντότητα η οποία ισχυρίζεται ότι της ανήκει. Επίσης πρέπει να είναι σίγουροι ότι το κλειδί αυτό δεν έχει πειραχτεί με οποιονδήποτε τρόπο, ή έχει αντικατασταθεί από κακόβουλους τρίτους. Έτσι, προέκυψε η ανάγκη πιστοποίησης του χρήστη.

Γενικά, η ασύμμετρη κρυπτογραφία συνετέλεσε καθοριστικά στη δημιουργία ενός φάσματος νέων εφαρμογών όπως οι ψηφιακές υπογραφές, η ανάπτυξη της Υποδομής Δημόσιου Κλειδιού (Public Key Infrastructure) και τα Ψηφιακά πιστοποιητικά. Οι αλγόριθμοι δημοσίου κλειδιού αποτελούν πλέον αναπόσπαστο συστατικό όλων των σύγχρονων κρυπτοσυστημάτων εγγυώμενοι τα χαρακτηριστικά της εμπιστευτικότητας, της αυθεντικότητας και της μη απάρνησης των ηλεκτρονικών επικοινωνιών και της αποθήκευσης δεδομένων.

Τρεις είναι οι κυριότερες κατηγορίες ασύμμετρων κρυπταλγορίθμων οι οποίες έχουν πρακτική εφαρμογή. Τις ξεχωρίζουμε με βάση το μαθηματικό πρόβλημα πάνω στο οποίο βασίζονται:

1. Αρχικά έχουμε την οικογένεια των αλγορίθμων που βασίζονται στην Παραγοντοποίηση Ακεραίων. Ανάμεσα σε αυτές τις μεθόδους κυριαρχεί ο RSA στον οποίο θα αναφερθούμε εν συντομία παρακάτω.

2. Στη συνέχεια έχουμε την οικογένεια των αλγορίθμων που βασίζονται στον Διακριτό Λογάριθμο. Χαρακτηριστικά παραδείγματα αλγορίθμων αυτής της οικογένειας είναι η ανταλλαγή κλειδιού των Diffie - Hellman, η κρυπτογράφηση Elgamal και ο Αλγόριθμος Ψηφιακής Υπογραφής (DSA).

3. Τέλος, έχουμε την οικογένεια αλγορίθμων που βασίζονται στις Ελλειπτικές Καμπύλες, οι οποίες είναι γενικεύσεις του Διακριτού λογαρίθμου. Τα πιο χαρακτηριστικά παραδείγματα εδώ είναι οι εκδοχές ανταλλαγής κλειδιού Diffie - Hellman με ελλειπτικές καμπύλες και ο Αλγόριθμος Ψηφιακής Υπογραφής με ελλειπτικές καμπύλες.

Οι παραπάνω οικογένειες αλγορίθμων εδράζονται σε δυσεπίλυτα προβλήματα της θεωρίας αριθμών. Με αυτόν τον όρο χαρακτηρίζονται όλα τα προβλήματα τα οποία ναι μεν έχουν λύση, ωστόσο αυτή η λύση δεν προκύπτει μέσω μιας διαδικασίας που απαιτεί πολυωνυμικό χρόνο για την περάτωσή της, έχουν δηλαδή απαγορευτική υπολογιστική πολυπλοκότητα. Ένα χαρακτηριστικό στοιχείο των προβλημάτων στους ασύμμετρους αλγορίθμους είναι ότι απαιτούν αριθμητική με πάρα πολλά ψηφία. Οι δύο πρώτες οικογένειες προτάθηκαν στα μέσα της δεκαετίας του 1970, ενώ οι ελλειπτικές καμπύλες δύο δεκαετίες αργότερα. Οι προσπάθειες κρυπτανάλυσης των εν λόγω αλγορίθμων περιορίζονται κυρίως στην προσπάθεια σχεδίασης αλγορίθμων που θα λύνουν τα αντίστοιχα προβλήματα σε πολυωνυμικό χρόνο, ή στη χρησιμοποίηση άλλων τεχνολογιών για την επίλυσή τους, π.χ. με τη χρήση κβαντικών υπολογιστών. Ένα άλλο σημείο που θα πρέπει να επισημανθεί είναι ότι παρουσιάζονται διαφοροποιήσεις στην απόδοση της ασύμμετρης κρυπτογράφησης ανάλογα με το μαθηματικό πρόβλημα στο οποίο στοχεύει ο αλγόριθμος, Για παράδειγμα, οι βέλτιστοι γνωστοί αλγόριθμοι για την επίλυση προβλημάτων ελλειπτικών καμπυλών είναι γενικά πολύ πιο χρονοβόροι από τους αντίστοιχους για παραγοντοποίηση ακεραίων. Ως εκ τούτου, για την ίδια αντοχή έναντι επιθέσεων θα πρέπει να χρησιμοποιηθούν πολύ μεγαλύτερα κλειδιά στην παραγοντοποίηση έναντι των ελλειπτικών καμπυλών. Για το λόγο αυτό οι ελλειπτικές καμπύλες έχουν καταστεί ιδιαίτερα δημοφιλείς από την αρχή της εμφάνισής τους στα μέσα της δεκαετίας του 1990, καθώς αποδίδουν μεγαλύτερη ασφάλεια σε λιγότερο χρόνο και με λιγότερη υπολογιστική ισχύ.

Έχοντας πλέον δει τα διάφορα είδη κρυπταλγορίθμων, μπορούμε να μπούμε και στη διαδικασία σύγκρισής τους. Για να μπορέσουμε να συγκρίνουμε διαφορετικούς αλγορίθμους, χρησιμοποιούμε την έννοια του Επιπέδου Ασφαλείας.

Λέμε ότι **ένας αλγόριθμος έχει Επίπεδο Ασφαλείας n bit** όταν η καλύτερη γνωστή επίθεση χρειάζεται 2^n βήματα για να πετύχει.

Σχετικά, ισχύει ότι στη συμμετρική κρυπτογράφηση οι αλγόριθμοι με επίπεδο ασφαλείας n bits έχουν μήκος κλειδιού n bits. Στην ασύμμετρη κρυπτογράφηση τα πράγματα είναι εντελώς διαφορετικά. Στον παρακάτω πίνακα αποτυπώνονται τα προτεινόμενα bit length για τους διάφορους κρυπτογραφικούς αλγορίθμους.

Οικογένεια Αλγορίθμων	Κρυπτοσυστήματα	Ασφάλεια σε bit			
		80	128	192	256
Παραγοντοποίηση Ακεραίου	RSA	1024	3072	7680	15360
Διακριτός Λογάριθμος	DH, DSA, Elgamal	1024	3072	7680	15360
Ελλειπτικές Καμπύλες	ECDH, ECDSA	160	256	384	512
Συμμετρικοί Αλγόριθμοι	AES, 3DES	80	128	192	256

Σχήμα 4.11: Συγκριτικός Πίνακας Επιπέδου Ασφαλείας

Με απώτερο σκοπό την μακροπρόθεσμη ασφάλεια, κρίνεται προτιμότερη η επιλογή επιπέδου ασφαλείας τουλάχιστον 128 bit. Αυτό όμως απαιτεί γενικά πολύ μεγάλα κλειδιά για τους ασύμμετρους αλγορίθμους, το οποίο μεταφράζεται σε πάρα πολύ χρόνο λόγω των πολλών αριθμητικών πράξεων που απαιτούνται.

4.3.1 Κρυπταλγόριθμος RSA

Ο κρυπταλγόριθμος RSA αποτελεί την πιο γνωστή εφαρμογή της πραγματικά πρωτοποριακής ιδέας του δημοσίου κλειδιού. Εφευρέτες του αλγορίθμου αυτού ήταν οι Ron Rivest, Adi Shamir και Leonard Adleman, οι οποίοι δημοσίευσαν την εργασία τους το 1978 και ονοματοδότησαν τον αλγόριθμο με τα αρχικά τους.

Σήμερα είναι από τα πιο διαδεδομένα κρυπτοσυστήματα για εφαρμογές, αν και τα συστήματα του διακριτού λογαρίθμου και των ελλειπτικών καμπυλών κερδίζουν γρήγορα έδαφος. Στην πράξη χρησιμοποιείται για μεταφορά μικρών σε μέγεθος δεδομένων (κρυπτογράφηση και μεταφορά κλειδιών) και για ψηφιακές υπογραφές. Εξάλλου, η υπολογιστική πολυπλοκότητα που παρουσιάζει οδήγησε εξαρχής στο συμπέρασμα ότι δεν θα αντικαθιστούσε του συμμετρικούς αλγορίθμους καθώς είναι ιδιαίτερα αργός σε σύγκριση με αυτούς.

Η ασφάλειά του είναι συνδεδεμένη με τη δυσκολία στην παραγοντοποίηση μεγάλων ακεραίων, ένα πρόβλημα για το οποίο δεν είναι γνωστή καμία αποδοτική τεχνική από την άποψη των πόρων που απαιτεί η επίλυση αυτού του προβλήματος.

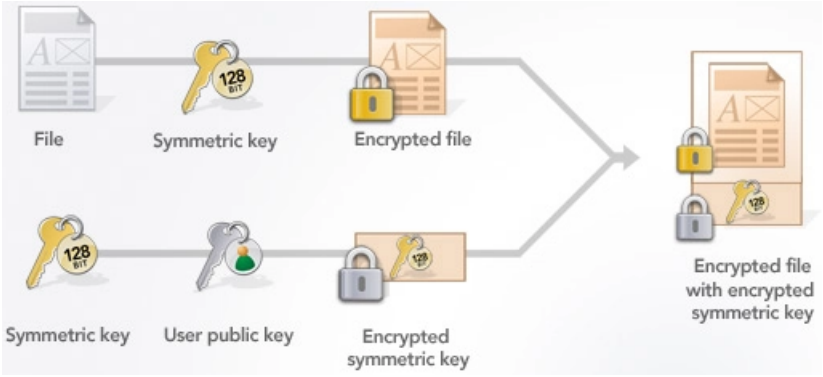
Η γενική ιδέα υλοποίησης του αλγορίθμου είναι η εξής:
Ένας χρήστης για να δημιουργήσει τα δυο κλειδιά του, επιλέγει δυο μεγάλους πρώτους αριθμούς τυχαία, έστω p και q, για τους οποίους πρέπει η διαφορά $p-q$ να είναι επίσης μεγάλη. Στη συνέχεια, υπολογίζει το γινόμενο τους $n = p \cdot q$, καθώς και την τιμή $\phi(n) = (p-1)(q-1)$. Ύστερα, επιλέγει ένα αριθμό e, ο οποίος πρέπει να είναι σχετικά πρώτος με το $\phi(n)$ και μεγαλύτερος του 1. Τέλος, υπολογίζει d τέτοιο ώστε $d \cdot e \equiv 1 \bmod \phi(n)$. Από τα παραπάνω, το ιδιωτικό του κλειδί είναι το d και το δημόσιο του, το ζευγάρι n, e. Προκειμένου να κρυπτογραφήσει κάποιος ένα μήνυμα m, υπολογίζει το $c = m \cdot e \bmod n$, ενώ η αποκρυπτογράφηση γίνεται υπολογίζοντας την ποσότητα $m = c \cdot m \bmod n$.

4.4 Υβριδική Κρυπτογράφηση

Η υβριδική κρυπτογράφηση χρησιμοποιεί συμμετρικές και ασύμμετρες μεθόδους για την κρυπτογράφηση των δεδομένων. Έτσι συνδυάζει τα πλεονεκτήματα και των δύο μεθόδων προκειμένου να ξεπεραστούν οι αναπόφευκτες δυσκολίες που συνεπάγεται η αποκλειστική χρήση μιας εκ των δύο μεθόδων κρυπτογράφησης.

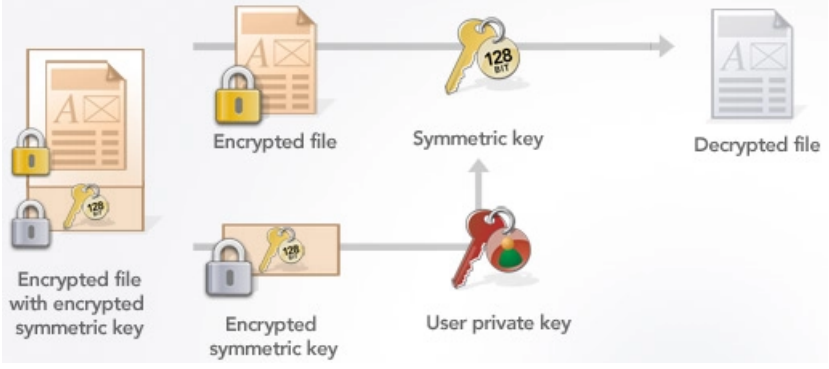
Πιο συγκεκριμένα, όπως είδαμε παραπάνω η συμμετρική κρυπτογράφηση απαιτεί το κλειδί κρυπτογράφησης να έχει συμφωνηθεί εκ των προτέρων μέσω ενός διαφορετικού και ασφαλούς καναλιού επικοινωνίας. Από την άλλη πλευρά, η ασύμμετρη κρυπτογράφηση ξεπερνάει πλήρως αυτό το θέμα, ωστόσο υστερεί σημαντικά ως προς τους υπολογιστικούς πόρους που απαιτούνται για την ολοκλήρωσή της έναντι της συμμετρικής κρυπτογράφησης.

Στην υβριδική κρυπτογράφηση τα δεδομένα είναι κρυπτογραφημένα συμμετρικά. Τα κλειδιά όμως, παραδίδονται με ασύμμετρη κρυπτογράφηση. Έτσι καθίσταται εφικτή η υπέρβαση των δυσκολιών που αναπτύξαμε παραπάνω. Η διαδικασία που υλοποιείται έχει ως εξής: Αρχικά, ο αποστολέας κρυπτογραφεί τα δεδομένα με ένα τυχαίο (συμμετρικό) κλειδί που έχει προηγουμένως δημιουργήσει. Τα δεδομένα μπορούν να αποκρυπτογραφηθούν από τον παραλήπτη μόνο με χρήση του ίδιου κλειδιού (συμμετρική κρυπτογράφηση). Για το λόγο αυτό, ο αποστολέας κρυπτογραφεί με το δημόσιο κλειδί του παραλήπτη το τυχαίο κλειδί που δημιούργησε και έπειτα στέλνει και τα δύο κρυπτομηνύματα όπως φαίνεται στο παρακάτω σχήμα.



Εικόνα 4.12: Η διαδικασία της κρυπτογράφησης σε ένα υβριδικό κρυπτοσύστημα

Ο παραλήπτης χρησιμοποιώντας το ιδιωτικό του κλειδί, αποκρυπτογραφεί το κρυπτοκείμενο που αφορούσε στο συμμετρικό κλειδί (το οποίο είχε κρυπτογραφηθεί ασύμμετρα). Με εκείνο το κλειδί πλέον, μπορεί να αποκρυπτογραφήσει τα δεδομένα που κρυπτογραφήθηκαν με τη μέθοδο της συμμετρικής κρυπτογράφησης.



Εικόνα 4.13: Η διαδικασία της αποκρυπτογράφησης σε ένα υβριδικό κρυπτοσύστημα

Κεφάλαιο 5

Αλγόριθμοι Κρυπτογράφησης Φωνής στις Τηλεπικοινωνίες

Στο παρελθόν, όταν τα μόνα τηλέφωνα που υπήρχαν ήταν τα ενσύρματα τεχνολογίας PSTN, η υποκλοπή των κλήσεων ήταν εφικτή είτε από την ίδια την τηλεφωνική εταιρεία ή από κάποιον που είχε φυσική πρόσβαση στη «γραμμή» μεταξύ των δύο συνομιλητών. Η εξέλιξη της τεχνολογίας είχε ως φυσικό επακόλουθο την εύρεση νέων κακόβουλων τεχνικών που σκοπό είχαν να υποσκάψουν τα θεμέλια της ασφάλειας των επικοινωνιών.

Ταυτόχρονα, με τη διείσδυση των τηλεπικοινωνιών στην καθημερινότητά μας, η ασφάλειά τους έχει επίσης τοποθετηθεί στο επίκεντρο του ενδιαφέροντος. Αυτή περιλαμβάνει μέτρα όπως η εξασφάλιση της εμπιστευτικότητας και της αυθεντικοποίησης, απαραίτητων εργαλείων για την εγγύηση τόσο της ιδιωτικότητας του χρήστη, όσο και του εισοδήματος των παρόχων.

Στη χώρα μας, είναι νωπές ακόμα οι μνήμες από το σκάνδαλο των τηλεφωνικών υποκλοπών σε βάρος ανώτατων κυβερνητικών στελεχών κατά την προ-ολυμπιακή περίοδο.

Στο κεφάλαιο αυτό θα αναφερθούμε περιληπτικά στους αλγορίθμους κρυπτογράφησης που χρησιμοποιούνται στα δίκτυα κινητής τηλεφωνίας και σκοπό έχουν να προστατέψουν την εμπιστευτικότητα της επικοινωνίας. Κατόπιν, θα εστιάσουμε στους κυριότερους από αυτούς τους αλγορίθμους που χρησιμοποιούμε μέχρι και σήμερα και θα αναλύσουμε τον τρόπο λειτουργίας τους.

Θα πρέπει ωστόσο να αναφέρουμε ότι οι παραπάνω κρυπταλγόριθμοι προστατεύουν την επικοινωνία «κατά τη μεταφορά» μεταξύ συσκευής χρήστη και παρόχου τηλεπικοινωνιών. Με άλλα λόγια το περιεχόμενο είναι πλήρως κατανοητό στην τηλεπικοινωνιακή εταιρεία και επομένως δεν προσφέρουν από άκρη σε άκρη κρυπτογράφηση. Με τον όρο αυτό ονομάζεται ένα σύστημα επικοινωνίας όπου μόνο οι χρήστες που επικοινωνούν μπορούν να κατανοήσουν τα μηνύματα, αφού θεωρητικά παρεμποδίζει τρίτα μέρη - ακόμα και τον πάροχο των υπηρεσιών αυτών - από το να έχουν πρόσβαση στο απαραίτητο για την κατανόηση της συνομιλίας κλειδί αποκρυπτογράφησης.

Στο τέλος του κεφαλαίου θα παρουσιάσουμε τους αλγόριθμους κρυπτογράφησης εφαρμογών VOIP και κρυπτοφώνων, που ισχυρίζονται ότι είναι σε θέση να παρέχουν υπηρεσίες κρυπτογράφησης από άκρη σε άκρη.

5.1 Αυθεντικοποίηση

Η χρησιμοποίηση ισχυρών κρυπτογραφικών αλγορίθμων για την προστασία της εμπιστευτικότητας δεν εγγυάται από μόνη της ότι ένα σύστημα επικοινωνιών είναι ασφαλές. Θα πρέπει ταυτόχρονα η δομή του συστήματος να έχει σχεδιαστεί προσεκτικά. Μία βασική αρχή για τη διασφάλιση ενός συστήματος κρυπτογράφησης ασύρματων τηλεπικοινωνιών είναι αυτό να πληροί το χαρακτηριστικό της αυθεντικοποίησης, δηλαδή να επιβεβαιώνει ότι ο σταθμός βάσης επικοινωνεί με έναν εξουσιοδοτημένο χρήστη και αντίθετα, ότι ο χρήστης επικοινωνεί με έναν εξουσιοδοτημένο σταθμό βάσης. Πριν αναλύσουμε λοιπόν τους αλγορίθμους που εξασφαλίζουν την εμπιστευτικότητα στις τηλεπικοινωνίες, θα κάνουμε μια σύντομη αναφορά στον τρόπο και το ιστορικό λειτουργίας της αυθεντικοποίησης.

Οι κρυπτογραφικοί αλγόριθμοι που εφαρμόζονται στα δίκτυα κινητών τηλεπικοινωνιών χρησιμοποιούν αλγορίθμους δημόσιου κλειδιού για να επιτύχουν την αρχή της αυθεντικοποίησης μέσω πρωτοκόλλων πρόκλησης – απάντησης (challenge–response protocols). Εν γένει η διαδικασία αυτή περιγράφεται με τον όρο AKA (authentication key agreement). Οι αλγόριθμοι που αποτελούν τον πυρήνα του συνήθως ταυτίζονται με τους αντίστοιχους που αποτελούν τον πυρήνα του αλγορίθμου κρυπτογράφησης για την προστασία της εμπιστευτικότητας. Επίσης, από τη διαδικασία της αυθεντικοποίησης προκύπτει ένα κλειδί που παίζει καθοριστικό ρόλο στην κρυπτογράφηση για την προστασία της εμπιστευτικότητας, την οποία θα δούμε παρακάτω.

Το AKA διαφέρει ανάλογα με τη γενιά του δικτύου. Επί της ουσίας η διαδικασία αυτή δεν υπήρχε στα δίκτυα πρώτης γενιάς όπου η αυθεντικοποίηση του χρήστη γινόταν απλά με την αντιστοίχιση κάποιων στοιχείων που μεταδίδονταν χωρίς κρυπτογράφηση. Τα στοιχεία αυτά μπορούσαν να υποκλαπούν εύκολα και έπειτα να επαναπρογραμματιστούν σε άλλη συσκευή, παραπλανώντας έτσι τον πάροχο ο οποίος κάλυπτε εντέλει το κόστος από αυτές κλήσεις που ουδέποτε πραγματοποιούσε ο πραγματικός χρήστης.

Στα δίκτυα δεύτερης γενιάς το AKA προέκυπτε μέσω μιας ελλιπούς - όπως αποδείχτηκε - διαδικασίας μονόδρομης αυθεντικοποίησης, όπου μόνο ο σταθμός βάσης ασκούσε έλεγχο σχετικά με το αν ο χρήστης ήταν εξουσιοδοτημένος. Ξεκινώντας από αυτή τη λεπτομέρεια καθίστατο εφικτή η παράκαμψη της αυθεντικοποίησης. Πώς; Με τη δημιουργία ενός ψευδο-σταθμού βάσης ο οποίος εξέπεμπε ιδιαίτερα δυνατό σήμα ώστε να αναγκάσει τα κινητά τηλέφωνα των χρηστών της γύρω περιοχής να συνδεθούν με αυτόν! Οι συσκευές των χρηστών δεν ήταν σε θέση να γνωρίζουν ότι έχουν συνδεθεί με έναν μη πιστοποιημένο σταθμό βάσης. Έπειτα, ο ψευδο-σταθμός βάσης ήταν σε θέση να υποκλέψει τα μυστικά κλειδιά που χρησιμοποιούνταν κατά την επικοινωνία του χρήστη με τον υποτιθέμενο σταθμό βάσης προκειμένου να αποδειχθεί η αυθεντικότητα του πρώτου, καθώς ανάγκαζε τη συσκευή του χρήστη να επικοινωνεί με μη ισχυρή κρυπτογράφηση την οποία στη συνέχεια έσπαγε. Το κενό ασφαλείας που περιγράψαμε ήταν γνωστό κατά το σχεδιασμό του αλγορίθμου, ωστόσο με τα τότε τεχνολογικά δεδομένα δεν θεωρήθηκε ότι προκαλούσε μεγάλο ρίσκο.

Στις επόμενες γενιές δικτύων το παραπάνω κενό ασφαλείας φυσικά διορθώθηκε, ωστόσο είναι δυνατή ακόμα και σήμερα η αξιοποίησή του προσβάλλοντας όσες τηλεφωνικές συσκευές συνδέονται με τα δίκτυα 2ης γενιάς.

5.2 Κρυπτογράφηση ανά γενιά δικτύου

Στην ενότητα αυτή θα παρουσιαστούν τα βασικότερα στοιχεία των αλγορίθμων κρυπτογράφησης για την προστασία της εμπιστευτικότητας.

1η γενιά - Έλλειψη κρυπτογράφησης

Η πρώτη γενιά δικτύων χαρακτηριζόταν από έλλειψη επαρκών χαρακτηριστικών ασφαλείας. Οι κλήσεις μπορούσαν να υποκλαπούν με τη χρησιμοποίηση συσκευών χαμηλού κόστους που ανίχνευαν και επεξεργάζονταν τη μετάδοση των ραδιοκυμάτων. Έτσι, δεν προξενεί εντύπωση το γεγονός ότι υπήρξαν πολλά περιστατικά υποκλοπών.

2η γενιά - A5

Τα προβλήματα ασφαλείας των αναλογικών επικοινωνιών της πρώτης γενιάς διευθετήθηκαν εν μέρει με την κρυπτογράφηση που επέβαλε η τεχνολογία GSM κατά τη μεταφορά της φωνής από τον χρήστη στο σταθμό βάσης καθώς και με αλλαγές στον έλεγχο αυθεντικοποίησης των χρηστών από το δίκτυο.

Για την επίτευξη των παραπάνω σκοπών εισήχθησαν οι οικογένειες κρυπταλγορίθμων A3 (αυθεντικοποίησης συνδρομητή), A5 (εμπιστευτικότητας) και A8 (παραγωγής κλειδιού συνόδου).

Όσον αφορά στην εμπιστευτικότητα, υπάρχουν 5 κρυπταλγόριθμοι ροής, ο απλούστατος A5/0, οι A5/1, A5/2 και A5/3 με μήκος κλειδιού 64 bits και τέλος ο A5/4 με μήκος κλειδιού 128 bits.

Οι A5/3 και A5/4 έχουν ως βάση τους τον κρυπταλγόριθμο KASUMI για τον οποίο θα μιλήσουμε παρακάτω και εισήχθησαν μεταγενέστερα. Κατά την περίοδο της πρώτης εμφάνισης της τεχνολογίας GSM οι διαθέσιμοι κρυπταλγόριθμοι ήταν οι A5/0, A5/1 και A5/2 για την ισχύ των οποίων υπήρξαν πιέσεις από ορισμένα κράτη που έθεταν ζητήματα κρατικής ασφάλειας σε περίπτωση ύπαρξης ισχυρών αλγορίθμων κρυπτογράφησης.

Ο ασθενέστερος από όλους τους κρυπταλγορίθμους είναι ο A5/0. Καταχρηστικά αναφέρεται ως κρυπταλγόριθμος καθώς πρόκειται για έναν αλγόριθμο στον οποίο αρχικό κείμενο και κρυπτοκείμενο ταυτίζονται. Χρησιμοποιείται σε ειδικές περιπτώσεις όπως σε κλήσεις έκτακτης ανάγκης.

Οι αλγόριθμοι A5/1 και A5/2 αποτελούσαν ουσιαστικά τις μόνες επιλογές κρυπτογράφησης. Ο A5/1 χρησιμοποιήθηκε κυρίως από χώρες της Ευρώπης και τις ΗΠΑ, ενώ χώρες με λιγότερο ευσταθές πολιτικό περιβάλλον χρησιμοποίησαν τον A5/2. Ο κώδικας των συγκεκριμένων κρυπταλγορίθμων δεν δημοσιοποιήθηκε ποτέ, ωστόσο αποκαλύφθηκε έπειτα από λίγο καιρό με τη χρήση ανάδρομης μηχανικής (reverse engineering), καθώς ήταν υλοποιημένος τόσο στον εκάστοτε σταθμό βάσης, όσο και στη συσκευή του χρήστη. Αυτό που αποκαλύφθηκε τότε ήταν ότι οι δύο αλγόριθμοι είχαν σημαντικά κενά ασφαλείας, με τον A5/2 να αποδεικνύεται πιο ευάλωτος σε σχέση με τον A5/1 καθώς μπορούσε να σπάσει μέσα σε ελάχιστα δευτερόλεπτα ακόμα και με περιορισμένους υπολογιστικούς πόρους. Ωστόσο ούτε ο A5/1 ήταν ιδιαίτερα ασφαλής κάτι που οδήγησε αργότερα στην εισαγωγή των κρυπταλγορίθμων A5/3 και A5/4 όπως είδαμε. Ταυτόχρονα ξεκίνησε η διαδικασία για την απόσυρση του A5/2, του πλέον απλού αλγορίθμου κρυπτογράφησης μετά τον A5/0.

3η γενιά – UEA1 & UEA2

Κατά τη μετάβαση στην τρίτη γενιά δικτύων προστέθηκε η δυνατότητα αυθεντικοποίησης του δικτύου από τον χρήστη συντελώντας έτσι στην αποτροπή επιθέσεων τύπου ψευδο-σταθμού βάσης για αυτά τα δίκτυα. Επίσης, η κρυπτογράφηση που μέχρι πρότινος έφτανε έως το σταθμό βάσης, μεταφέρθηκε πιο εσωτερικά στο δίκτυο του παρόχου. Τέλος, οι αλγόριθμοι κρυπτογράφησης που χρησιμοποιήθηκαν δημοσιοποιήθηκαν εξαρχής σε αντίθεση με ό,τι συνέβη στους πρώτους αλγόριθμους κρυπτογράφησης των δικτύων δεύτερης γενιάς.

Ο αλγόριθμος κρυπτογράφησης που χρησιμοποιήθηκε καταρχήν στα δίκτυα 3ης γενιάς ήταν ο UEA1, ένας αλγόριθμος ροής που κρυπτογραφεί/αποκρυπτογραφεί πακέτα δεδομένων με χρήση ενός κλειδιού εμπιστευτικότητας που προέκυπτε από τη διαδικασία της αυθεντικοποίησης. Το μήκος των πακέτων δεδομένων κυμαινόταν από 1 μέχρι 20000 bits. Ο αλγόριθμος UEA1 χρησιμοποιούσε ως γεννήτρια κλειδοροής τον KASUMI σε τύπο λειτουργίας ανάδρασης εξόδου.

Αργότερα, προστέθηκε και άλλος ένα κρυπτογραφικός αλγόριθμος, ο UEA2, που διέφερε αρκετά σε σύγκριση με τον UEA1 προκειμένου να διαφυλαχθεί η εμπιστευτικότητα ακόμα και αν ένας από τους δύο αλγορίθμους κατέρρεε. Ο UEA2 ήταν επίσης αλγόριθμος ροής με τη διαφορά ότι το μήκος του πακέτου που κρυπτογραφούσε/αποκρυπτογραφούσε κυμαινόταν από 1 μέχρι 2^{32} bits, ενώ για την

γεννήτρια κλειδοροής χρησιμοποιούταν ο αλγόριθμος SNOW 3G.

Στους παραπάνω αλγορίθμους θα πρέπει να προσθέσουμε και τον UEA0, ο οποίος αντίστοιχα με τον A5/0 δεν πραγματοποιεί κρυπτογράφηση της ομιλίας.

4η γενιά - 128-EEA1 & 128-EEA2

Αναμφίβολα η μεγαλύτερη αλλαγή που σηματοδότησε η έλευση των δικτύων 4ης γενιάς ήταν η εξ ολοκλήρου χρησιμοποίηση της τεχνολογίας μεταγωγής πακέτου ακόμα και για υπηρεσίες φωνής. Το παραπάνω όπως προαναφέραμε οδήγησε σε πολλές αλλαγές της αρχιτεκτονικής του δικτύου σε σχέση με τα προγενέστερα δίκτυα. Ένα ακόμα χαρακτηριστικό που διαφοροποιήθηκε σε αυτή τη γενιά δικτύων ήταν ότι η κρυπτογράφηση έφτανε ακόμα πιο εσωτερικά στο δίκτυο του παρόχου σε σχέση με τις προηγούμενες γενιές δικτύων. Κατά τον καθορισμό των αλγορίθμων κρυπτογράφησης για τα δίκτυα 4ης γενιάς (LTE) θεωρείτο δεδομένο ότι θα έπρεπε ο αλγόριθμος AES να αξιοποιηθεί. Ταυτόχρονα, υπήρχε ο ήδη επιτυχημένος αλγόριθμος SNOW 3G που είχε επιλεχθεί ως εναλλακτικός αλγόριθμος κρυπτογράφησης στα δίκτυα 3ης γενιάς.

Βασιζόμενοι σε αυτούς τους 2 αλγορίθμους προέκυψαν οι αλγόριθμοι κρυπτογράφησης 128-EEA1 και 128-EEA2. Ο αριθμός 128 αναφέρεται στο μήκος του κλειδιού κρυπτογράφησης και συμπεριλαμβάνεται στην ονομασία προκειμένου να είναι εύκολη η διάκριση των αλγορίθμων σε περίπτωση μελλοντικής μετάβασης σε κλειδί μεγαλύτερου μεγέθους. Ο κρυπταλγόριθμος 128-EEA1 ταυτίζεται με τον UEA2 των δικτύων 3ης γενιάς. Ο κρυπταλγόριθμος 128-EEA2 είναι ένας αλγόριθμος ροής που βασίζεται στον αλγόριθμο AES σε τύπο λειτουργίας μετρητή (counter mode). Ένας τρίτος κρυπταλγόριθμος, ο 128-EEA3, χρησιμοποιείται στα δίκτυα 4ης γενιάς στην Κίνα. Ο αλγόριθμος αυτός βασίζεται στον αλγόριθμο κρυπτογράφησης ZUC που σχεδιάστηκε σε ερευνητικό κέντρο της Κινεζικής Ακαδημίας Επιστημών και φέρει ορισμένες ομοιότητες στη δομή του με τον SNOW 3G.

Τέλος, όπως και στις δύο προηγούμενες γενιές δικτύων υπάρχει ο ειδικών χρήσεων αλγόριθμος 128-EEA0.

5.3 KASUMI

Η ανάπτυξή του βασίστηκε στον κρυπταλγόριθμο MISTY-1 που είχε αναπτυχθεί από την Mitsubishi Electric Corporation. Ο αρχικός αλγόριθμος τροποποιήθηκε ελαφρώς προκειμένου να είναι ευκολότερη η φυσική (hardware) υλοποίησή του, αλλά και για να προσαρμοστεί σε κάποιες απαιτήσεις ασφαλείας των δικτύων τηλεπικοινωνιών 3ης γενιάς.

Ο KASUMI είναι ένας αλγόριθμος πακέτου με κλειδί μήκους 128-bit. Η είσοδος και η έξοδός του αποτελούνται από 64-bit. Ο πυρήνας του αλγορίθμου αποτελείται από ένα δίκτυο Feistel 8 γύρων/επαναλήψεων. Οι κυκλικές συναρτήσεις στο κύριο δίκτυο Feistel είναι μη αντιστρέψιμοι μετασχηματισμοί που προσομοιάζουν στο δίκτυο Feistel. Σε κάθε κύκλο, η κυκλική συνάρτηση χρησιμοποιεί ένα κυκλικό κλειδί που αποτελείται από 8 υποκλειδιά μήκους 16-bit τα οποία προέρχονται από το αρχικό κλειδί μήκους 128-bit χρησιμοποιώντας μία γραμμική συνάρτηση προσδιορισμού κλειδιού.

Προσδιορισμός Κλειδιού

Το αρχικό κλειδί μήκους 128-bit K χωρίζεται σε οκτώ υποκλειδιά μήκους 16-bit, K_i :

$$K = K_1||K_2||K_3||K_4||K_5||K_6||K_7||K_8$$

Επιπλέον, δημιουργείται ένα τροποποιημένο κλειδί K', ομοίως διαχωρισμένο σε υποκλειδιά μήκους 16-bit, K'_i. Το τροποποιημένο κλειδί προέρχεται από το αρχικό κλειδί αφού αυτό γίνει XOR με τον αριθμό 0x123456789ABCDEFFEDCBA9876543210.

Τα κυκλικά κλειδιά έχουν ως εξής:
$KL_{i,1} = ROL(K_i, 1)$
$KL_{i,2} = K'_{i+2}$
$KO_{i,1} = ROL(K_{i+1}, 5)$
$KO_{i,2} = ROL(K_{i+5}, 8)$
$KO_{i,3} = ROL(K_{i+6}, 13)$
$KI_{i,1} = K'_{i+4}$
$KI_{i,2} = K'_{i+3}$
$KI_{i,3} = K'_{i+7}$

Σε περίπτωση που ο δείκτης $i+j$ είναι μεγαλύτερος του 8, αφαιρούμε 8 για να πάρουμε τον πραγματικό δείκτη του υποκλειδιού.

Ο αλγόριθμος

Ο αλγόριθμος KASUMI επεξεργάζεται "λέξεις" μήκους 64-bit, διαχωρίζοντάς τις σε δύο τμήματα μήκους 32-bit το καθένα, το αριστερό L_i και το δεξιό R_i. Η αρχική λέξη είναι η ένωση του αριστερού και του δεξιού τμήματος του πρώτου γύρου:

$$input = R_0||L_0$$

Σε κάθε κύκλο το δεξί τμήμα υφίσταται XOR με το αποτέλεσμα της κυκλικής συνάρτησης, και έπειτα τα δύο τμήματα εναλλάσσονται:

$L_i = F_i(KL_i, KOi, KI_i, L_{i-1}) \oplus R_{i-1}$
$R_i = L_{i-1}$,

όπου τα *KLi, KOi, KIi* είναι τα κυκλικά κλειδιά του i-στου γύρου.

Οι κυκλικές συναρτήσεις για τους άρτιους και περιττούς κύκλους διαφέρουν. Σε κάθε περίπτωση πάντως, η κυκλική συνάρτηση είναι σύνθεση δύο συναρτήσεων, της FL_i και της FO_i.

Πιο συγκεκριμένα, για έναν περιττό κύκλο είναι:
$F_i(K_i, L_{i-1}) = FO(KO_i, KL_i, FL(KL_i, L_{i-1}))$,

ενώ για έναν άρτιο
$F_i(K_i, L_{i-1}) = FL(KL_i, FO(KO_i, KI_i, L_{i-1}))$

Η έξοδος είναι η ένωση των εξόδων του τελευταίου γύρου.

Οι συναρτήσεις FL και FO χωρίζουν τις εισόδους μήκους 32-bit σε δύο τμήματα των 16-bit. Η συνάρτηση FL πραγματοποιεί μη αντιστρέψιμες πράξεις σε επίπεδο bit, ενώ η FO είναι μία μη αντιστρέψιμη συνάρτηση τριών γύρων που προσομοιάζει στο δίκτυο Feistel.

Βασικά στοιχεία του KASUMI

Η συνάρτηση FL

Η είσοδος της συνάρτησης $FL(KL_i,x)$ μήκους 32-bit διαχωρίζεται σε δύο τμήματα των 16-bit $x = l||r$.
Αρχικά το αριστερό τμήμα της εισόδου l υφίσταται δυαδική πρόσθεση με το κυκλικό κλειδί $KL_{i,1}$ και ολισθαίνει αριστερά κατά ένα bit. Το αποτέλεσμα υφίσταται XOR με το δεξί τμήμα της εισόδου r ώστε να προκύψει το δεξί τμήμα της εξόδου r'.

$$r' = ROL(l \wedge KL_{i,1}, 1) \oplus r$$

Έπειτα το δεξί τμήμα της εξόδου r' γίνεται XOR με το κυκλικό κλειδί $KL_{i,2}$ και ολισθαίνει αριστερά κατά ένα bit. Το αποτέλεσμα γίνεται ξανά XOR με το αριστερό μισό της εισόδου l ώστε να προκύψει το αριστερό μισό της εξόδου l'.

$$l' = ROL(r' \vee KL_{i,2}, 1) \oplus l$$

Η έξοδος της συνάρτησης είναι η συνένωση των αριστερών και των δεξιών τμημάτων $x' = l'||r'$.

Η συνάρτηση FO

Η είσοδος της συνάρτησης $FO(KO_i, KI_i, x)$ διαχωρίζεται σε δύο τμήματα μήκους 16-bit
$x = l_0||r_0$ που αποτελούν τα ορίσματα ενός δικτύου Feistel τριών γύρων.

Σε κάθε έναν από τους τρεις γύρους (συμβολίζονται με τον δείκτη j, ο οποίος προφανώς παίρνει τις τιμές 1, 2 ή 3) τα δύο τμήματα τροποποιούνται ως εξής:

$r_j = FI(KI_{i,j}, l_{j-1} \oplus KO_{i,j}) \oplus r_{j-1}$
$l_j = r_{j-1}$

Η έξοδος της συνάρτησης είναι $x' = l_3||r_3$.

Η συνάρτηση FI

Η συνάρτηση FI προσομοιάζει στο δίκτυο Feistel. Η είσοδος της συνάρτησης $FL(K_i, x)$ μήκους 16-bit διαχωρίζεται σε δύο μέρη $x = l_0||r_0$ από τα οποία το l_0 έχει μήκος 9 bit και το r_0 7 bit.

Τα bits του αριστερού τμήματος l_0 αρχικά αναμειγνύονται με τη χρήση ενός πίνακα αντικατάστασης (S9) μήκους 9-bit και το αποτέλεσμα γίνεται XOR με το δεξί τμήμα r_0 το οποίο προηγουμένως έχει επεκταθεί κατάλληλα. Το αποτέλεσμα που προκύπτει είναι η νέα τιμή του δεξιού τμήματος r_1 μήκους 9-bit.

$$r_1 = S9(l_0) \oplus (00||r_0)$$

Τα bits του δεξιού τμήματος r_0 αναμειγνύονται με τη χρήση ενός πίνακα αντικατάστασης (S7) μήκους 7-bit και το αποτέλεσμα γίνεται XOR με τα 7 λιγότερο σημαντικά bits ("LS7") του δεξιού τμήματος r_1 για να προκύψουν τα 7 bit του αριστερού τμήματος l_1.

$$l_1 = S7(r_0) \oplus LS7(r_1)$$

Η ενδιάμεση λέξη $x_1 = l_1||r_1$ γίνεται XOR με το κυκλικό κλειδί KI για να προκύψει το $x_2 = l_2||r_2$ από τα οποία το l_2 έχει μήκος 7 bits και το r_2 9 bits.

$$x_2 = KI \oplus x_1$$

Έπειτα, τα bits του δεξιού τμήματος r_2 αναμειγνύονται με χρήση του πίνακα αντικατάστασης S9 και το αποτέλεσμα γίνεται XOR με το κατάλληλα επεκτεταμένο αριστερό τμήμα l_2, ώστε να προκύψει το νέο δεξί τμήμα μήκους 9-bit ως έξοδος του r_3.

$$r_3 = S9(r_2) \oplus (00||l_2)$$

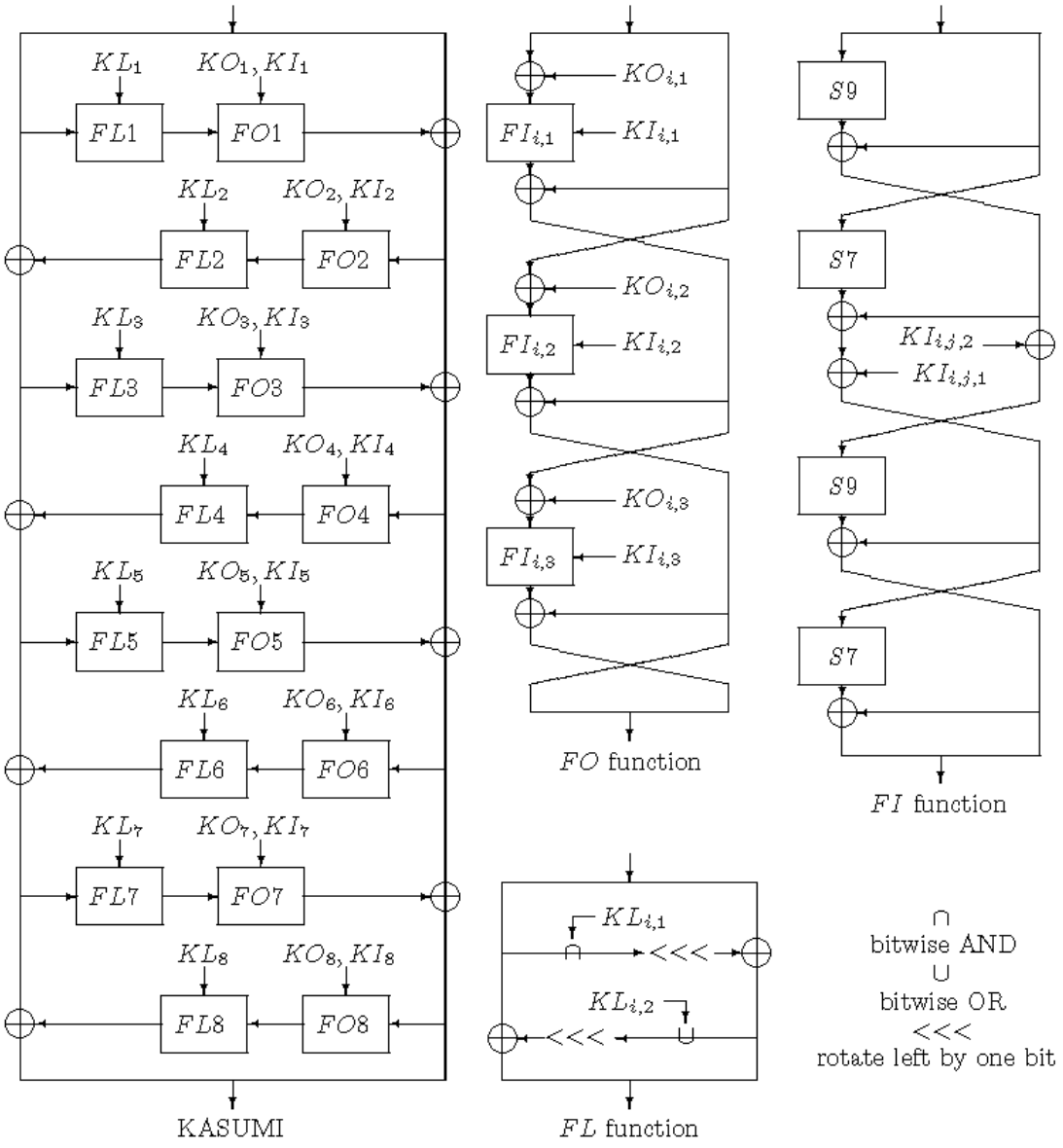
Τέλος, τα bits του αριστερού μέρους l_2 αναμειγνύονται με χρήση του πίνακα αντικατάστασης S7 και το αποτέλεσμα γίνεται XOR με τα 7 λιγότερο σημαντικά bits (LS7) του δεξιού τμήματος της εξόδου r_3 ώστε να προκύψει το αριστερό τμήμα της εξόδου l_3 μήκους 7 bit.

$$l_3 = S7(l_2) \oplus LS7(r_3)$$

Η έξοδος είναι η συνένωση του τελικού αριστερού και δεξιού τμήματος $x' = l_3||r_3$.

Πίνακες Αντικατάστασης

Οι πίνακες αντικατάστασης (S-boxes) S7 και S9 αποτελούνται από δυαδικές εκφράσεις AND-XOR, ώστοσο στις μέρες μας έχει κυριαρχήσει η χρήση άμεσων πινάκων αναζήτησης.



Σχήμα 5.1: Η αρχιτεκτονική του κρυπτογραφικού αλγορίθμου KASUMI

Κρυπτανάλυση του KASUMI

Καθώς ο αλγόριθμος ήταν δημοσιευμένος εξαρχής, υπήρξαν πολλές κρυπταναλυτικές προσπάθειες εναντίον του. Μια τυπική κρυπταναλυτική προσέγγιση εναντίον των αλγορίθμων με επαναληπτική δομή είναι η μείωση του αριθμού των γύρων και η επίθεση στην ασθενέστερη αυτή έκδοση του αλγορίθμου. Στο πλαίσιο αυτό υπάρχουν επιθέσεις που είναι ταχύτερες από την επίθεση εκτενούς αναζήτησης για μια έκδοση του KASUMI με 6 αντί για 8 γύρους στο δίκτυο Feistel. Επίσης υπάρχει μία επίθεση ελαφρώς ταχύτερη από την επίθεση εκτενούς αναζήτησης στον πλήρη αλγόριθμο KASUMI (8 γύροι). Τέλος, έχουν ανακαλυφθεί πολύ αποτελεσματικές επιθέσεις τύπου related key - chosen plaintexts εναντίον του αλγορίθμου, ωστόσο οι επιθέσεις αυτές δεν βρίσκουν εφαρμογή στην αρχιτεκτονική των δικτύων 3ης γενιάς και στον τύπο λειτουργίας του κρυπταλγορίθμου UEA1.

5.4 SNOW 3G

Ο SNOW 3G είναι η τρίτη έκδοση του αλγορίθμου SNOW. Πρόκειται για κρυπταλγόριθμο ροής ο οποίος χρησιμοποιώντας μια μεταβλητή αρχικοποίησης και ένα κλειδί μήκους 128 bit, παράγει μια ακολουθία κρυπτογραμμάτων μήκους 32 bit το καθένα. Αυτά τα κρυπτογράμματα χρησιμοποιούνται ακολούθως για την απόκρυψη της αρχικής πληροφορίας.

Πρώτα πραγματοποιείται η αρχικοποίηση του κλειδιού και ο χρονισμός του κρυπταλγορίθμου, χωρίς την παραγωγή εξόδου. Έπειτα ο αλγόριθμος παράγει το κλειδί και παίρνοντας 32 bit αρχικού κειμένου παράγει 32 bit κρυπτογράμματος σε κάθε κύκλο (ή και το αντίστροφο). Επιπλέον, ο SNOW 3G αποτελείται από δύο αλληλεπιδρούντα στοιχεία, έναν Κυλιόμενο Καταχωρητή Γραμμικής Ανάδρασης (LFSR) και μία Μηχανή Πεπερασμένων Καταστάσεων (FSM).

Βασικά Στοιχεία του SNOW 3G

Η συνάρτηση MULx

Η συνάρτηση MULx λαμβάνει ως είσοδο δύο τιμές V και c μήκους 8 bit η κάθε μία και δίνει ως έξοδο μία τιμή μήκους 8 bit.Ορίζεται ως εξής:

Αν το αριστερότερο bit της εισόδου V ισούται με 1, τότε
$\text{MULx}(V, c) = (V <<_8 1) \oplus c$,
διαφορετικά
$\text{MULx}(V, c) = (V <<_8 1)$.

Η συνάρτηση MULxPOW

Η συνάρτηση MULxPOW λαμβάνει ως είσοδο δύο τιμές V και c μήκους 8 bit και έναν θετικό ακέραιο i και δίνει ως έξοδο μία τιμή μήκους 8-bit. Ορίζεται αναδρομικά ως εξής:

Αν το i ισούται με 0, τότε
$\text{MULxPOW}(V, i, c) = V$,
διαφορετικά
$\text{MULxPOW}(V, i, c) = \text{MULx}(\text{MULxPOW}(V, i-1, c), c)$.

Γραμμικός Καταχωρητής Ολίσθησης με Ανάδραση (LFSR)

Ο LFSR αποτελείται από 16 στάδια, s_0 - s_{15}, καθένα από τα οποία κρατάει 32 bit πληροφορίας και η ανάδρασή του καθορίζεται από ένα πρωταρχικό πολυώνυμο επί του πεπερασμένου πεδίου $GF(2^{32})$.

Μηχανή Πεπερασμένων Καταστάσεων (FSM)

Η FSM βασίζεται σε τρεις καταχωρητές μήκους 32-bit, $R1, R2, R3$. Η λειτουργία του FSM βασίζεται στην είσοδο από το LFSR. Επίσης χρησιμοποιεί τους πίνακες αντικατάστασης S1 και S2 για την ανανέωση των τιμών των καταχωρητών $R2$ και $R3$.

Η διαδικασία χρονισμού του LFSR

Ο χρονισμός του LFSR έχει δύο διαφορετικούς τύπους λειτουργίας, τον τύπο αρχικοποίησης και τον τύπο κλειδοροής. Και στις δύο λειτουργίες εμπλέκονται οι συναρτήσεις MUL_a και DIV_a, τα αποτελέσματα των οποίων προκύπτουν είτε μέσω χρήσης της συνάρτησης $MULxPOW$ που ορίσαμε προηγουμένως, είτε μέσω πινάκων αναζήτησης.

LFSR - Τύπος Αρχικοποίησης

Αρχικά, ο LFSR λαμβάνει ως είσοδο μια λέξη F μήκους 32 bit η οποία είναι η έξοδος του FSM. Έστω $s_i = s_{i,0}||s_{i,1}||s_{i,2}||s_{i,3}$, όπου με $s_{i,0}$ συμβολίζουμε το πιο σημαντικό και με $s_{i,3}$ το λιγότερο σημαντικό byte του s-οστού στοιχείου του LFSR.

Υπολογίζουμε την ενδιάμεση τιμή v ως εξής:

$v = (s_{0,1}||s_{0,2}||s_{0,3}||0x00) \oplus MUL_a(s_{0,0}) \oplus s_2 \oplus (0x00||s_{11,0}||s_{11,1}||s_{11,2}) \oplus DIV_a(s_{11,3} \oplus F)$.

Έπειτα, θέτουμε
$s_i = s_{i+1}, 0 \leqslant i \leqslant 14$
$s_i = v, i = 15$

LFSR - Τύπος Κλειδοροής

Στη λειτουργία κλειδοροής ο LFSR δεν δέχεται κανένα όρισμα. Ο υπολογισμός της ενδιάμεσης τιμής v γίνεται ως εξής:
$v = (s_{0,1}||s_{0,2}||s_{0,3}||0x00) \oplus MUL_a(s_{0,0}) \oplus s_2 \oplus (0x00||s_{11,0}||s_{11,1}||s_{11,2}) \oplus DIV_a(s_{11,3})$.

Έπειτα, όπως προηγουμένως θέτουμε
$s_i = s_{i+1}, 0 \leqslant i \leqslant 14$
$s_i = v, i = 15$

Η διαδικασία χρονισμού του FSM

Η FSM δέχεται ως είσοδο δύο λέξεις από το LFSR, τις s_5, s_{15} και παράγει μία λέξη εξόδου F μήκους 32 bit ως εξής:

$F = (s_15 \boxplus R1) \oplus R2$

Έπειτα οι καταχωρητές ανανεώνονται και υπολογίζεται η ενδιάμεση τιμή r ως εξής:
$r = R2 \boxplus (R3 \oplus s_5)$.

Τέλος, θέτουμε
$R3 = S_2(R2)$,
$R2 = S_1(R1)$,
$R1 = r$.

Αρχικοποίηση SNOW 3G

Ο αλγόριθμος αρχικοποιείται με τη χρήση ενός κλειδιού μήκους 128 bit που αποτελείται από τέσσερις λέξεις k_0, k_1, k_2, k_3 μήκους 32 bit η κάθε μία, και μιας μεταβλητής αρχικοποίησης μήκους 128 bit η οποία ομοίως χωρίζεται σε τέσσερις ισομεγέθεις λέξεις, τις IV_0, IV_1, IV_2, IV_3. Η αρχικοποίηση δίνει τις παρακάτω τιμές στα στοιχεία του LFSR (όπου 1 εννοείται ο αριθμός $(0xffffffff)$):

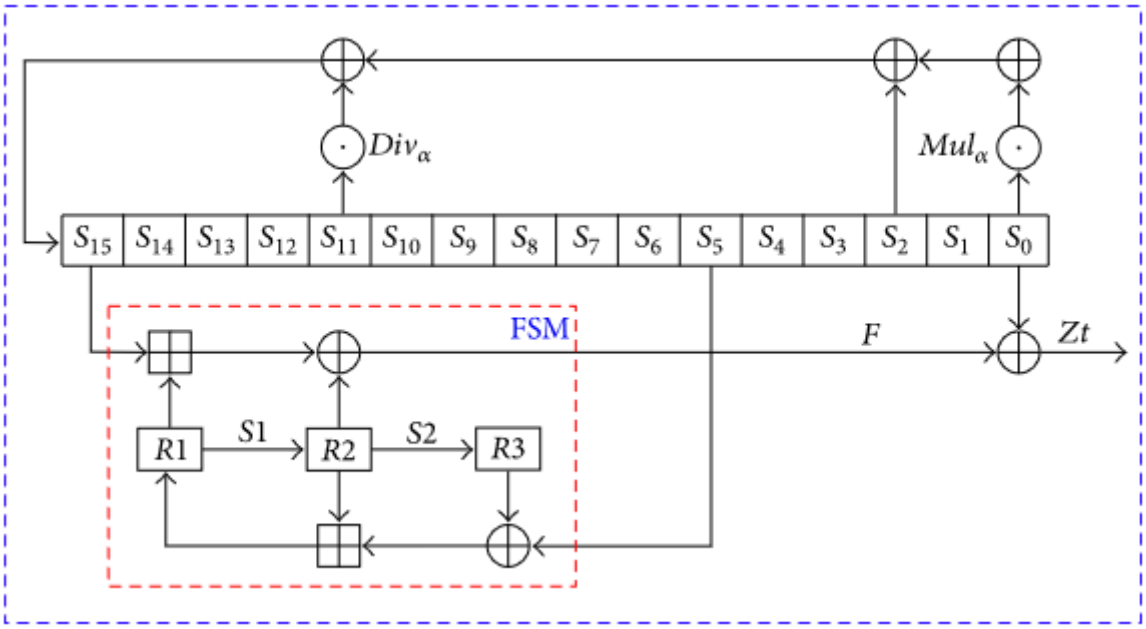
$s_{15} = k_3 \oplus IV_0$
$s_{14} = k_2$
$s_{13} = k_1$
$s_{12} = k_0 \oplus IV_1$
$s_{11} = k_3 \oplus 1$
$s_{10} = k_2 \oplus 1 \oplus IV_2$
$s_9 = k_1 \oplus 1 \oplus IV_3$
$s_8 = k_0 \oplus 1$
$s_7 = k_3$
$s_6 = k_2$
$s_5 = k_1$
$s_4 = k_0$
$s_3 = k_3 \oplus 1$
$s_2 = k_2 \oplus 1$
$s_1 = k_1 \oplus 1$
$s_0 = k_0 \oplus 1$

Το FSM αρχικοποιείται στις τιμές R1 = R2 = R3 = 0.
Έπειτα ο αλγόριθμος τρέχει σε ειδική λειτουργία χωρίς να παράγει έξοδο, εκτελώντας διαδοχικά τις διαδικασίες χρονισμού του FSM και αρχικοποίησης του LFSR για 32 φορές.

Παραγωγή κλειδορροής

Αρχικά το FSM χρονίζεται με την έξοδο να παραμένει αναξιοποίητη και το LFSR χρονίζεται με τη λειτουργία κλειδορροής. Έπειτα παράγονται επαναλαμβανόμενα λέξεις μήκους 32 bit με την εξής διαδικασία:

- Πρώτα, το FSM χρονίζεται και παράγει μία έξοδο F μήκους 32 bit.
- Δεύτερον, υπολογίζεται η επόμενη λέξη κλειδορροής $z_t = F \oplus s_0$.
- Τρίτον, το LFSR χρονίζεται με τη λειτουργία κλειδορροής.



Σχήμα 5.2: Ο αλγόριθμος SNOW 3G κατά την παραγωγή κλειδορροής

Κρυπτανάλυση

Το επίπεδο ασφαλείας του SNOW 3G έχει αξιολογηθεί έναντι πλήθους κρυπταναλυτικών επιθέσεων, όπως:

1. Αλγεβρικές Επιθέσεις. Ο συγκεκριμένος τύπος επίθεσης βασίζεται στην εύρεση και επίλυση ενός συστήματος πολυδιάστατων πολυωνυμικών εξισώσεων επί ενός πεπερασμένου πεδίου. Στην περίπτωση του SNOW 3G κρίνεται ότι η εισαγωγή του καταχωρητή R3 επιδρά καταλυτικά στην αντιμετώπιση αυτής της τακτικής κρυπτανάλυσης.

2. Επιθέσεις τύπου Heuristic Guess-and-determine. Η συγκεκριμένη κατηγορία επιθέσεων εφαρμόζεται εναντίον αλγορίθμων ροής και αφορά σε χρήση ενός συγκεκριμένου αλγορίθμου εναντίον τους. Η μόνη προϋπόθεση για τη χρήση της είναι ότι όλες οι μεταβλητές το υποκείμενου αλγορίθμου θα πρέπει να έχουν το ίδιο μέγεθος και κάθε μία να καθορίζεται μοναδικά αν όλες οι άλλες είναι γνωστές. Βελτίωση στα αποτελέσματα της παραπάνω μεθόδου κρυπτανάλυσης προσφέρει η χρήση βοηθητικών πολυωνύμων σχετικά μικρού βαθμού, καθώς με βάση τα αποτελέσματα η πολυπλοκότητα μειώνεται στο μισό ($O(2^{160})$ από $O(2^{320})$).

3. Επιθέσεις επανασυγχρονισμού (Resynchronization attacks) - Η αξιολόγηση αυτής της μεθόδου κρυπτανάλυσης έγινε με τη χρήση επιθέσεων τύπου chosen IV . Οι ερευνητές εκμεταλλευόμενοι το γεγονός ότι το S-box S2 δεν περιείχε διαδικασία μετάθεσης έδειξαν ότι ο βαθμός εξασφάλισης της αρχής της διάχυσης της πληροφορίας δεν παρείχε μεγάλο περιθώριο ασφαλείας ενάντια σε επιθέσεις αυτού του τύπου.

5.5 AES

Ο AES (Advance Encryption Standard) είναι ένας ευρέως διαδεδομένος συμμετρικός κρυπταλγόριθμος πακέτου ο οποίος - μεταξύ άλλων - έχει γίνει αποδεκτός ως ο επίσημος αλγόριθμος κρυπταγράφησης από την ομοσπονδιακή κυβέρνηση των ΗΠΑ. Είναι μία παραλλαγή του Rijndael όπου χρησιμοποιούνται πακέτα μήκους 128 bit και κλειδιά μήκους 128, 192 ή 256 bits. Αυτό είναι και το μοναδικό σημείο διαφοροποίησης του AES από το Rijndael καθώς στον τελευταίο, το μέγεθος του πακέτου και το μέγεθος του κλειδιού μπορούν να καθοριστούν ανεξάρτητα σε οποιοδήποτε πολλαπλάσιο του 32 μεταξύ 128 και 256 bit. Βασίζεται σε μια σχεδιαστική αρχή των επαναληπτικών αλγορίθμων δέσμης που είναι γνωστή ως δίκτυο αντικατάστασης - μετάθεσης. Όπως φανερώνει η ονομασία της, αποτελεί συνδυασμό αντικαταστάσεων και μεταθέσεων.

Ο AES λειτουργεί επί ενός 4 x 4 πίνακα από byte, που ονομάζεται state. Θεωρούμε τα byte ως πολυώνυμα πάνω από το $GF(2)$, όπου για να ορίσουμε τον πολλαπλασιασμό χρησιμοποιούμε το ανάγωγο πολυώνυμο $m(x) = x^8 + x^4 + x^3 + x + 1$. Με αυτόν τον τρόπο κατασκευάζουμε μια αναπαράσταση για το $GF(2^8)$ και κάθε byte θεωρείται στοιχείο του συγκεκριμένου πεπερασμένου σώματος. Το μήκος του κλειδιού καθορίζει και τον αριθμό των επαναλήψεων της κυκλικής συνάρτησης και έχει ως εξής: 10 επαναλήψεις για κλειδιά μήκους 128 bits, 12 για κλειδιά μήκους 192 bits και 14 για κλειδιά μήκους 256 bits.

Συνοπτική παρουσίαση του αλγορίθμου

Παρακάτω παρουσιάζονται συνοπτικά τα βήματα που ακολουθεί ο αλγόριθμος για την κρυπτογράφηση ενός πακέτου. Προκειμένου να καταστεί εφικτή η αποκρυπτογράφηση εφαρμόζεται ένα σύνολο από αντίστροφα (ως προς τα παρακάτω) βήματα.

1. Επέκταση Κλειδιού - Ο AES απαιτεί ένα ξεχωριστό κυκλικό κλειδί μήκους 128 bit για κάθε γύρο, συν ένα επιπλέον.

2. Αρχική εφαρμογή του AddRoundKey() - συνδυασμός του state με το κυκλικό κλειδί χρησιμοποιώντας bitwise XOR.

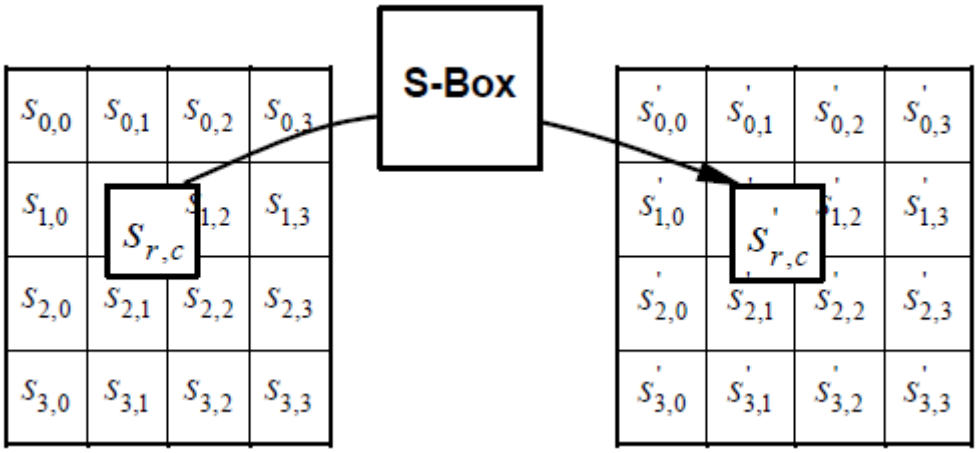
3. 9, 11 ή 13 κύκλοι, αποτελούμενοι από:
   i) SubBytes — μη γραμμική αντικατάσταση των byte
   ii) ShiftRows — ολίσθηση γραμμών του state
   iii)MixColumns— γραμμικός μετασχηματισμός των στηλών του state
   iv) AddRoundKey

4. Τελικός κύκλος:
   i) SubBytes
   ii) ShiftRows
   iii)AddRoundKey

Βασικά στοιχεία του AES

SubBytes

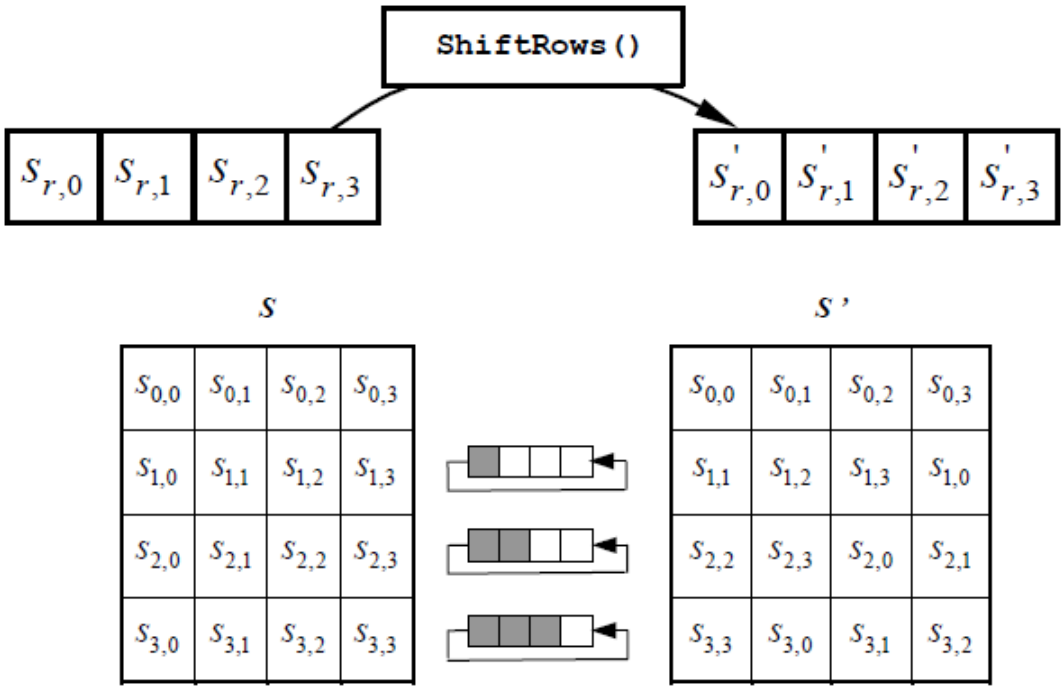
Σε αυτό το βήμα κάθε byte $s_{i,j}$ στον πίνακα κατάστασης αντικαθίσταται από ένα SubByte $S(s_{i,j})$ χρησιμοποιώντας έναν προκαθορισμένο και αντιστρέψιμο πίνακα αντικατάστασης (S-box) μήκους 8 bit που αποτελεί σύνθεση του πολλαπλασιαστικού αντιστρόφου επί του πεπερασμένου πεδίου $GF(2^8)$ και ενός αφινικού μετασχηματισμού επί του πεπερασμένου πεδίου $GF(2)$. Με αυτή τη διαδικασία εξασφαλίζεται η μη γραμμικότητα του κρυπταλγορίθμου.



Σχήμα 5.3: Ο μετασχηματισμός SubBytes()

ShiftRows

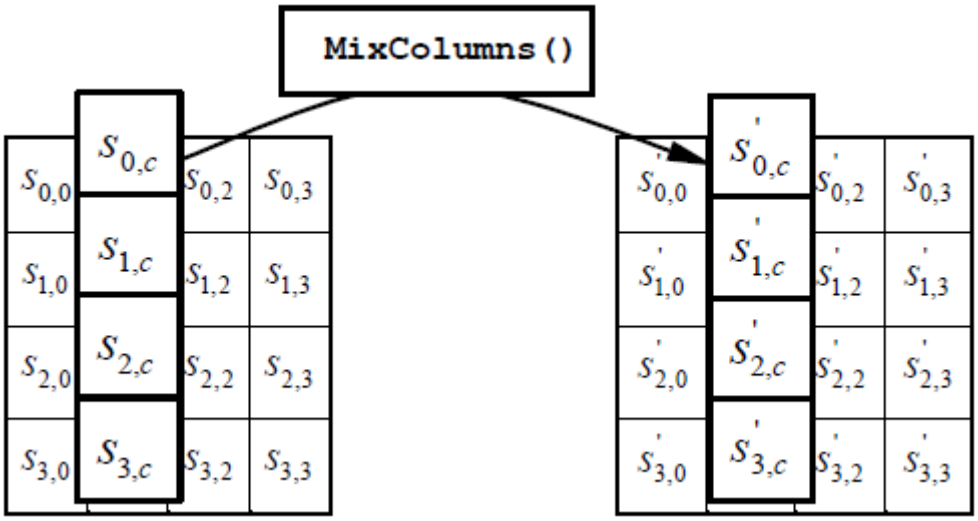
Σε αυτό το στάδιο τα bytes των τριών τελευταίων γραμμών ολισθαίνουν κυκλικά κατά προκαθορισμένο τρόπο: Τα bytes της δεύτερης γραμμής μετατοπίζονται κατά μία θέση αριστερά, της τρίτης κατά δύο θέσεις, ενώ της τέταρτης κατά τρεις θέσεις. Με αυτόν τον τρόπο κάθε στήλη του state μετά την εφαρμογή του (αντιστρέψιμου) μετασχηματισμού $ShiftRows()$ αποτελείται από bytes από κάθε στήλη του state πριν την εφαρμογή του μετασχηματισμού. Η σημασία του παραπάνω μετασχηματισμού είναι πολύ μεγάλη καθώς έτσι αποφεύγεται η ανεξάρτητη κρυπτογράφηση των στηλών κατά την οποία ο AES θα εκφυλιζόταν σε τέσσερις ανεξάρτητους κρυπταλγορίθμους δέσμης.



Σχήμα 5.4: Ο μετασχηματισμός ShiftRows()

MixColumns

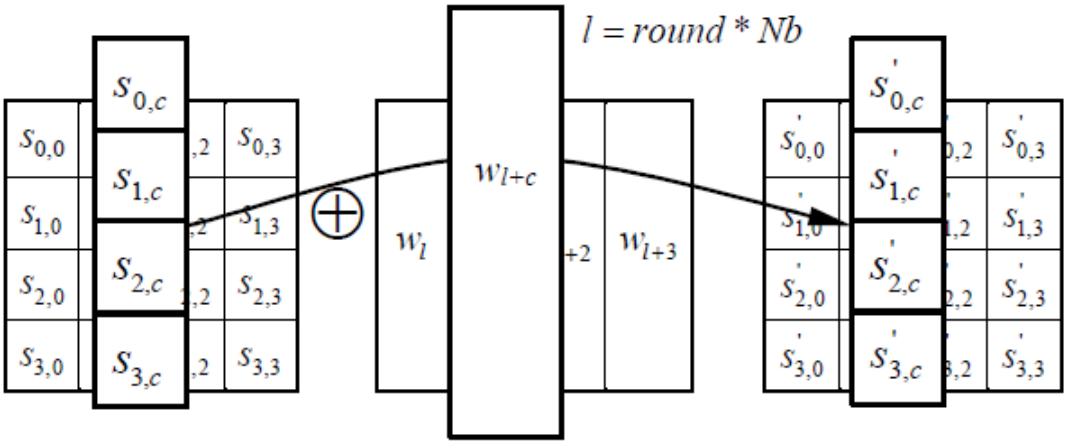
Σε αυτό το βήμα, τα τέσσερα bytes κάθε στήλης συνδυάζονται με τη χρήση ενός αντιστρέψιμου γραμμικού μετασχηματισμού. Η συνάρτηση δέχεται στην είσοδο και δίνει στην έξοδο τέσσερα bytes, όπου κάθε byte πριν την εφαρμογή του μετασχηματισμού επηρεάζει και τα τέσσερα bytes μετά την εφαρμογή του. Η χρήση των μετασχηματισμών $ShiftRows()$ και $MixColumns()$ εξασφαλίζει στον κρυπταλγόριθμο το απαραίτητο χαρακτηριστικό της διάχυσης της πληροφορίας. Κατά τον μετασχηματισμό αυτό κάθε στήλη μετασχηματίζεται με τη χρήση ενός σταθερού πίνακα.



Σχήμα 5.5: Ο μετασχηματισμός MixColumns()

AddRoundKey

Σε αυτό το βήμα, το υποκλειδί συνδυάζεται με το state. Για κάθε γύρο, ένα υποκλειδί προκύπτει από το κύριο κλειδί με τη χρήση του Προσδιορισμού Κλειδιού Rijndael. Το υποκλειδί έχει το ίδιο μήκος με το state. Κάθε byte από αυτό το υποκλειδί συνδυάζεται με το αντίστοιχο byte του state με χρήση bitwise XOR.



Σχήμα 5.6: Ο μετασχηματισμός AddRoundKey()

Επέκταση Κλειδιού

Η διαδικασία Επέκτασης Κλειδιού δέχεται σαν είσοδο το αρχικό κλειδί, K, που δίνει ο χρήστης και το χρησιμοποιεί για να παραγάγει ένα κλειδί για κάθε γύρο. Ο αλγόριθμος χρειάζεται αρχικά ένα σύνολο $4N_b$ [1] byte (για την πρώτη $AddRoundKey()$) και στη συνέχεια κάθε ένας από τους N_r [2] γύρους χρειάζεται $4N_b$ byte για το αντίστοιχο κλειδί. Έτσι η επέκταση κλειδιού παράγει το διάνυσμα w μήκους $N_b(N_r + 1)$, του οποίου κάθε συνιστώσα $[w_i]$, με $0 \leq i < N_b(N_r + 1)$, είναι μία λέξη τεσσάρων byte. Βασικό ρόλο στην παραπάνω διαδικασία διαδραματίζουν:

- Η συνάρτηση $SubWord()$, η οποία δέχεται είσοδο μήκους 4 byte και εφαρμόζει έναν πίνακα αντικατάστασης σε κάθε ένα byte προκειμένου να προκύψει η έξοδος.
- Η συνάρτηση $RotWord()$, η οποία δέχεται μια λέξη $[\alpha_0, \alpha_1, \alpha_2, \alpha_3]$ ως είσοδο και πραγματοποιεί μια κυκλική μετάθεση επιστρέφοντας ως έξοδο τη λέξη $[\alpha_1, \alpha_2, \alpha_3, \alpha_0]$.
- Ο πίνακας κυκλικά σταθερής λέξης $Rcon[i]$ περιέχει τις τιμές $[x^{i-1}, 0, 0, 0, 0, 0, 0]$, όπου το x^{i-1} είναι δυνάμεις του x επί του πεπερασμένου πεδίου $GF(2^8)$.

Επίσης με μια προσεκτική ματιά στον αλγόριθμο συμπεραίνουμε ότι οι πρώτες N_k λέξεις του επεκτεταμένου κλειδιού προκύπτουν από το κλειδί κρυπτογράφησης. Έπειτα, κάθε λέξη $w[i]$ προκύπτει ως αποτέλεσμα XOR της προηγούμενης λέξης $w[i-1]$ με την λέξη που βρισκόταν Nk θέσεις πριν, $w[i-Nk]$. Στην ειδική περίπτωση που οι θέσεις είναι πολλαπλάσια του Nk, εφαρμόζεται ένας μετασχηματισμός στο $w[i-1]$ πριν το XOR, και ακολούθως γίνεται XOR με την κυκλική σταθερά $Rcon[i]$. Ο μετασχηματισμός που μνημονεύσαμε παραπάνω αποτελείται από εφαρμογή των συναρτήσεων $RotWord()$ και $SubWord()$.

Κρυπτανάλυση

Ο κρυπταλγόριθμος AES έχει δοκιμαστεί εναντίον πλήθους κρυπταναλυτικών επιθέσεων, χωρίς ωστόσο να έχει ανακαλυφθεί κάποια αδυναμία του αλγορίθμου που να επιτρέπει την αποδοτική κρυπτανάλυσή του. Οι βασικότερες κρυπταναλυτικές επιθέσεις που έχουν εξαπολυθεί εναντίον του AES είναι:

1. Οι επιθέσεις τύπου Fault Attack, οι οποίες αλλάζουν τις σχέσεις αντιστοίχισης των S-box, εισάγοντας τεχνηέντως λάθη τα οποία στη συνέχεια αξιοποιούνται για την ανάκτηση του μυστικού κλειδιού με χρήση επίθεσης ωμής βίας περιορισμένης πολυπλοκότητας μέχρι και $O(2^{32})$.
2. Αλγεβρικές επιθέσεις, έναντι των οποίων ο AES δεν ήταν σχεδιασμένος εξαρχής έτσι ώστε να είναι ιδιαίτερα ασφαλής απέναντί τους. Οι επιθέσεις αυτές εκμεταλλεύονται το γεγονός των στατικών S-box. Έτσι σχεδιάστηκαν τροποποιήσεις του AES όπου πλέον τα S-box είναι μεταβαλλόμενα, με τη μεταβολή τους εξαρτάται κάθε φορά από το κλειδί.
3. Επιθέσεις τύπου side - channel. Όπως έχουμε ήδη αναφέρει, αυτού του τύπου οι επιθέσεις δεν επικεντρώνονται στην αξιοποίηση μαθηματικών αδυναμιών του κρυπταλγορίθμου αλλά στην αξιοποίηση γνώσης που πηγάζει από υλοποίηση του αλγορίθμου σε φυσική μορφή. Για την αποτροπή αυτού του είδους των επιθέσεων έχουν ληφθεί αντίμετρα που βασίζονται σε μαθηματικές ιδιότητες του αλγορίθμου Rijndael και είναι συμβατές με τα δημοσιευμένα πρότυπα, ενώ παράλληλα αυξάνουν τη πολυπλοκότητα του αλγορίθμου σε τέτοιο βαθμό ώστε οι επιθέσεις να καθίστανται πλέον πολύ απαιτητικές, αν όχι ανέφικτες.

[1] Όπου N_b το πηλίκο του μεγέθους πακέτου προς 32

[2] Όπου N_r ο αριθμός των επαναλήψεων της κυκλικής συνάρτησης

5.6 VOIP

Όπως ήδη αναφέραμε είναι πλέον πολύ διαδεδομένη η επικοινωνία με χρήση υπηρεσιών VOIP. Κατά τη χρήση αυτών των υπηρεσιών, τα δεδομένα (φωνή, εικόνες, βίντεο, κτλ) μεταφέρονται μέσω του παρόχου τηλεπικοινωνιών (σταθερής ή κινητής).

Στην περίπτωση της κινητής τηλεφωνίας και προκειμένου για δίκτυα 3ης ή 4ης γενιάς, τα δεδομένα που μεταφέρονται έχουν την τυπική κρυπτογράφηση που αναλύσαμε στις προηγούμενες ενότητες. Ωστόσο, ανάλογα με την εφαρμογή που χρησιμοποιείται, ενδέχεται να υπάρχει ακόμα ένα επίπεδο κρυπτογράφησης που έχει εφαρμοστεί από την ίδια την εφαρμογή πριν αυτά αποσταλούν μέσω του δικτύου. Όμως ποιος έχει εντέλει πρόσβαση στα αποκρυπτογραφημένα δεδομένα;

Στην ενότητα αυτή θα εξετάσουμε τις διαδικασίες που ακολουθεί το WhatsApp, καθώς σύμφωνα με τα στοιχεία που παρατίθενται, χρησιμοποιεί από άκρη σε άκρη κρυπτογράφηση και φυσικά πρόκειται για μία από τις γνωστότερες εφαρμογές τέτοιου είδους.

Προκειμένου να είναι σε θέση η εφαρμογή να παρέχει από άκρη σε άκρη κρυπτογράφηση, οι διαδικασίες που ακολουθεί ένας χρήστης που πραγματοποιεί μια κλήση είναι οι ακόλουθες:

- Αρχικά δημιουργεί μια κρυπτογραφημένη σύνοδο με τον καλούμενο.
- Έπειτα παράγει ένα τυχαίο κύριο μυστικό κλειδί ("master secret") μήκους 256-bit για χρησιμοποίηση με το πρωτόκολλο SRTP.
- Τέλος, ο χρήστης στέλνει ένα κρυπτογραφημένο μήνυμα στον καλούμενο που περιέχει το SRTP master secret.

Αν ο καλούμενος απαντήσει, τότε έχει εγκαθιδρυθεί μια κρυπτογραφημένη κλήση. Όλες αυτές οι διαδικασίες φυσικά γίνονται αυτοματοποιημένα από τη συσκευή του χρήστη, χωρίς επέμβαση του ίδιου. Τόσο για τη δημιουργία της αρχικής κρυπτογραφημένης συνόδου, όσο και για την δημιουργία του master secret, χρησιμοποιούνται αλγόριθμοι που αξιοποιούν τις ελλειπτικές καμπύλες Diffie – Hellman για την εγκαθίδρυση του κλειδιού κρυπτογράφησης. Χαρακτηριστικό παράδειγμα είναι ο Curve25519, ένας από τους ταχύτερους αλγορίθμους κρυπτογράφησης ελλειπτικών καμπυλών. Το πρωτόκολλο SRTP χρησιμοποιείται για την ασφαλή μετάδοση δεδομένων σε πραγματικό χρόνο με χρήση συμμετρικών μεθόδων κρυπτογράφησης. Ο προεπιλεγμένος κρυπταλγόριθμος που χρησιμοποιεί είναι ο AES, τον οποίο αναλύσαμε προηγουμένως.

5.7 Κρυπτόφωνα

Αντίστοιχη λογική με αυτήν που εξετάσαμε στην αμέσως προηγούμενη ενότητα διέπει και τη λειτουργία των κρυπτοφώνων. Τα κρυπτόφωνα είναι συσκευές κινητής τηλεφωνίας, όμοιες με αυτές που χρησιμοποιεί στην καθημερινότητά του η πλειονότητα του κόσμου. Η διαφορά τους έγκειται στο ότι ενσωματώνουν εξειδικευμένες λειτουργίες που παρέχουν ένα επίπεδο από άκρη σε άκρη κρυπτογράφησης και συνεπώς ασφάλειας έναντι των υποκλοπών σε σχέση με την τυπική κρυπτογράφηση των δικτύων κινητής τηλεφωνίας που εξετάσαμε προηγουμένως. Το παραπάνω επιτυγχάνεται με τη χρήση ενός κρυπτογραφικού τσιπ που χειρίζεται τις διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης. Εντός του τσιπ έχουν προγραμματιστεί δύο αλγόριθμοι: Ένας αλγόριθμος ανταλλαγής κλειδιού (ασύμμετρος) για τον καθορισμό του κλειδιού κρυπτογράφησης και ένας συμμετρικός αλγόριθμος για την κρυπτογράφηση της φωνής. Το γεγονός της ύπαρξης του επιπλέον επιπέδου κρυπτογράφησης εξασφαλίζει ότι ακόμα και αν κάποιος καταφέρει να σπάσει την κρυπτογράφηση που παρέχεται από το δίκτυο π.χ. GSM, πάλι δεν θα είναι σε θέση να κατανοήσει τη συνομιλία.

Για να υπάρξει ασφαλής επικοινωνία μέσω κρυπτοφώνων θα πρέπει και οι δύο χρήστες να επιλέξουν στα κρυπτόφωνά τους τη λειτουργία επικοινωνίας με κρυπτογράφηση. Αμέσως συμφωνείται ένα κλειδί συνόδου μέσω του αλγορίθμου ανταλλαγής κλειδιού και έπειτα δημιουργείται ένας σύντομος κωδικός επιβεβαίωσης που προέρχεται από το κλειδί συνόδου. Στο τέλος - αφού έχει ενεργοποιηθεί η λειτουργία κρυπτογράφησης - κάθε συνομιλητής διαβάζει τον κωδικό επιβεβαίωσής του και συγκρίνει τον κωδικό που ακούει από τον άλλο χρήστη με τον κωδικό που ο άλλος χρήστης θα έπρεπε να βλέπει. Αν οι δύο αυτοί κωδικοί δεν ταιριάζουν σημαίνει ότι έχει πραγματοποιηθεί επίθεση τύπου man-in-the-middle. Το κλειδί συνόδου χρησιμοποιείται για μία και μόνο κλήση. Μετά την ολοκλήρωσή της διαγράφεται οριστικά από τη μνήμη του τηλεφώνου, χωρίς να υπάρχει η δυνατότητα επανακατασκευής του. Όλες οι υποκλαπείσες κλήσεις είναι κρυπτογραφημένες. Μπορούν να αναλυθούν, ωστόσο δεν υπάρχει τρόπος ανάκτησης της μη κρυπτογραφημένης ομιλίας παρά μόνο με ιδιαιτέρως χρονοβόρες μεθόδους.

Τα κρυπτόφωνα τελευταίας γενιάς ενσωματώνουν και άλλες λειτουργίες πέραν της απλής παροχής κρυπτογραφημένης ομιλίας και δεν υστερούν σε τίποτα σε σχέση με τα σύγχρονα, έξυπνα κινητά τηλέφωνα. Πολλά μάλιστα προσφέρουν τη δυνατότητα επιλογής από το χρήστη του αλγόριθμου κρυπτογράφησης που θα εφαρμοστεί, μέσα από μία λίστα μερικών επιλογών. Ωστόσο ένα αρνητικό για αυτού του είδους τις συσκευές είναι ότι η από άκρη σε άκρη κρυπτογραφημένη επικοινωνία καθίσταται εφικτή μόνο στην περίπτωση που και οι δύο χρήστες χρησιμοποιούν συμβατές μεταξύ τους συσκευές, συνεπώς η επικοινωνία με χρήστες συμβατικών τηλεφώνων ή και κρυπτοφώνων άλλων κατασκευαστών δεν μπορεί να εξασφαλιστεί με κρυπτογράφηση από άκρη σε άκρη.

Κεφάλαιο 6

Συμπεράσματα

Στα προηγούμενα μελετήσαμε τη διαρκώς εξελισσόμενη πορεία των κρυπτογραφικών αλγορίθμων. Όπως παρατηρήσαμε, σε κάθε μετάβαση διατηρούνται τα καλά στοιχεία και γίνεται προσπάθεια για την αντιμετώπιση των ευάλωτων σημείων των αλγορίθμων. Οι επιτυχημένοι αλγόριθμοι διατηρούνται συνήθως, ωστόσο αξιοποιούνται και νέοι που θεωρούμε ότι θα μας φανούν ιδιαίτερα χρήσιμοι στο μέλλον. Επίσης είναι λογικό να επιδιώκεται η συνύπαρξη ριζικά διαφορετικών αλγορίθμων κρυπτογράφησης προκειμένου να είναι πιο δύσκολη η ταυτόχρονη διάσπασή τους από μεγάλα τεχνολογικά ή επιστημονικά βήματα που ενδέχεται να πραγματοποιηθούν στον τομέα της κρυπτανάλυσης.

Στο πλαίσιο αυτό, έχουν γίνει ήδη αποδεκτοί κάποιοι αλγόριθμοι για την υλοποίηση της κρυπτογράφησης στα δίκτυα 5ης γενιάς. Πρόκειται για τους ίδιους αλγορίθμους που χρησιμοποιήθηκαν στην 4η γενιά δικτύων και αποδείχτηκαν ιδιαίτερα ασφαλείς επιλογές ενώ υποστηρίζουν και τις αναγκαίες προδιαγραφές ταχύτητας. Το παραπάνω είναι πολύ σημαντικό καθώς οι αλγόριθμοι κρυπτογράφησης στα δίκτυα 5ης γενιάς είναι βέβαιο ότι θα πρέπει να υποστηρίζουν πολύ υψηλή ταχύτητα φυσικής υλοποίησης ώστε να είναι σε θέση να υποστηρίξουν τη μέγιστη ταχύτητα του δικτύου χωρίς να δημιουργούν φαινόμενα «συνωστισμού».

Σε επόμενη φάση, ένας δυνατός υποψήφιος αλγόριθμος κρυπτογράφησης φαίνεται ότι είναι ο AES με χρήση του Real-time Transport Protocol όπως αποκαλύπτουν πρόσφατες διαρροές στην ιστοσελίδα της 3GPP. Αυτή η μέθοδος κρυπτογράφησης μας παραπέμπει στη διαδικασία κρυπτογράφησης που ακολουθεί και η εφαρμογή WhatsApp και την οποία παρουσιάσαμε νωρίτερα. Ωστόσο, δεν θα πρέπει να ξεχνάμε ότι σε αυτή την περίπτωση δεν αναφερόμαστε σε κρυπτογράφηση από άκρη σε άκρη, αλλά σε κρυπτογράφηση κατά τη μεταφορά στο δίκτυο του παρόχου. Ένα άλλο στοιχείο που φαίνεται ότι εξετάζεται ήδη, είναι το ενδεχόμενο επέκτασης του μήκους του κλειδιού. Σε αυτήν την περίπτωση, η 3GPP έχει ταχθεί υπέρ του διπλασιασμού σε σχέση με το προηγούμενο, πράγμα που σημαίνει ότι πιθανότατα οδηγούμαστε σε κλειδί 256 bits.

Όλα τα παραπάνω σχεδιάζονται έχοντας υπόψιν και το ενδεχόμενο της ανάπτυξης των κβαντικών υπολογιστών, γεγονός το οποίο αναμένεται να αλλάξει δραματικά τα μέχρι σήμερα δεδομένα σχετικά με τη μέγιστη υπολογιστική ισχύ των συστημάτων. Πώς θα διασφαλιστεί το απόρρητο των επικοινωνιών σε μια τέτοια περίπτωση; Είμαστε σε θέση να προσφέρουμε από άκρη σε άκρη κρυπτογράφηση των επικοινωνιών η οποία θα ελέγχεται από τους χρήστες και με τη δυνατότητα εφαρμογής της να μην περιορίζεται σε συγκεκριμένα μοντέλα τηλεφωνικών συσκευών αλλά να είναι ευρέως προσιτή και διαλειτουργική; Τα παραπάνω ερωτήματα καθώς και οι προτάσεις για λύσεις τους θα μπορούσαν να εξεταστούν στο πλαίσιο μιας διδακτορικής διατριβής για το μέλλον της κρυπτογραφίας στα δίκτυα κινητής τηλεφωνίας εν αναμονή του ψηφιακού μετασχηματισμού που βρίσκεται προ των πυλών.

Παράρτημα

UEA1 - KASUMI

* Ο ίδιος ουσιαστικά αλγόριθμος χρησιμοποιείται και στα δίκτυα 2ης γενιάς όπου φέρει την ονομασία A5/3 και A5/4, ανάλογα με το μήκος του κλειδιού με το οποίο λειτουργεί (64/128 bits).

Header file

```
/*--------------------------------------------------------
 * Kasumi.h
 *--------------------------------------------------------*/

typedef unsigned  char   u8;
typedef unsigned short  u16;
typedef unsigned  long  u32;

/*----- a 64-bit structure to help with endian issues -----*/

typedef union {
u32 b32[2];
u16 b16[4];
u8  b8[8];
} REGISTER64;

/*------------- prototypes --------------------------------*/

void KeySchedule( u8 *key );
void Kasumi( u8 *data );
u8 * f9( u8 *key,int count,int fresh, int dir,u8 *data,int length );
void f8( u8 *key,int count,int bearer,int dir,u8 *data,int length );
```

Code

```
/*---------------------------------------------------------------
 * F8 - Confidentiality Algorithm
 *---------------------------------------------------------------
 *
 * A sample implementation of f8, the 3GPP Confidentiality algorithm.
 *
 * This has been coded for clarity, not necessarily for efficiency.
 *
 * This will compile and run correctly on both Intel (little endian)
 *  and Sparc (big endian) machines. (Compilers used supported 32-bit ints)
 *
 * Version 1.0 05 November  1999
 *
 *---------------------------------------------------------------*/

#include "kasumi.h"
#include <stdio.h>

/*--------------------------------------------------------
 * f8()
 * Given key, count, bearer, direction,  data,
 * and bit length  encrypt the bit stream
 *--------------------------------------------------------*/
void f8( u8 *key, int count, int bearer, int dir, u8 *data, int length )
{
REGISTER64 A; /* the modifier */
REGISTER64 temp; /* The working register */
int i, n;
```

```
u8  ModKey[16]; /* Modified key */
u16 blkcnt; /* The block counter */

/* Start by building our global modifier */

temp.b32[0]  = temp.b32[1]  = 0;
A.b32[0]     = A.b32[1]     = 0;

/* initialise register in an endian correct manner*/

A.b8[0]  = (u8) (count>>24);
A.b8[1]  = (u8) (count>>16);
A.b8[2]  = (u8) (count>>8);
A.b8[3]  = (u8) (count);
A.b8[4]  = (u8) (bearer<<3);
A.b8[4] |= (u8) (dir<<2);

/* Construct the modified key and then "kasumi" A */

for( n=0; n<16; ++n )
ModKey[n] = (u8)(key[n] ^ 0x55);
KeySchedule( ModKey );

Kasumi( A.b8 ); /* First encryption to create modifier */

/* Final initialisation steps */

blkcnt = 0;
KeySchedule( key );

/* Now run the block cipher */

while( length > 0 )
{
/* First we calculate the next 64-bits of keystream */

/* XOR in A and BLKCNT to last value */

temp.b32[0] ^= A.b32[0];
temp.b32[1] ^= A.b32[1];
temp.b8[7] ^= (u8)  blkcnt;
temp.b8[6] ^= (u8) (blkcnt>>8);

/* KASUMI it to produce the next block of keystream */

Kasumi( temp.b8 );

/* Set <n> to the number of bytes of input data *
 * we have to modify.  (=8 if length <= 64) */

if( length >= 64 )
n = 8;
else
n = (length+7)/8;

/* XOR the keystream with the input data stream */

for( i=0; i<n; ++i )
*data++ ^= temp.b8[i];
```

```
length -= 64; /* done another 64 bits */
++blkcnt; /* increment BLKCNT */
}
}

/*---------------------------------------------------------
 * e n d    o f    f 8 . c
 *---------------------------------------------------------*/
```

Header file

```
/*---------------------------------------------------------
 * Kasumi.h
 *---------------------------------------------------------*/

typedef unsigned  char   u8;
typedef unsigned short  u16;
typedef unsigned  long  u32;

void KeySchedule( u8 *key );
void Kasumi( u8 *data );
```

C Code

```
/*---------------------------------------------------------------------
 * Kasumi.c
 *---------------------------------------------------------------------
 *
 * A sample implementation of KASUMI, the core algorithm for the
 * 3GPP Confidentiality and Integrity algorithms.
 *
 * This has been coded for clarity, not necessarily for efficiency.
 *
 * This will compile and run correctly on both Intel (little endian)
 * and Sparc (big endian) machines. (Compilers used supported 32-bit ints).
 *
 * Version 1.1 08 May 2000
 *
 *---------------------------------------------------------------------*/

#include "Kasumi.h"

/*--------- 16 bit rotate left ----------------------------------------*/

#define ROL16(a,b) (u16)((a<<b)|(a>>(16-b)))

/*------- unions: used to remove "endian" issues ----------------------*/

typedef union {
u32 b32;
u16 b16[2];
u8  b8[4];
} DWORD;

typedef union {
u16 b16;
u8  b8[2];
} WORD;

/*-------- globals: The subkey arrays ---------------------------------*/
```

```
static u16 KLi1[8], KLi2[8];
static u16 KOi1[8], KOi2[8], KOi3[8];
static u16 KIi1[8], KIi2[8], KIi3[8];


/*------------------------------------------------------------------
 * FI()
 * The FI function (fig 3).  It includes the S7 and S9 tables.
 * Transforms a 16-bit value.
 *------------------------------------------------------------------*/
static u16 FI( u16 in, u16 subkey )
{
u16 nine, seven;
static u16 S7[] = {
54, 50, 62, 56, 22, 34, 94, 96, 38, 6, 63, 93, 2, 18,123, 33,
55,113, 39,114, 21, 67, 65, 12, 47, 73, 46, 27, 25,111,124, 81,
53, 9,121, 79, 52, 60, 58, 48,101,127, 40,120,104, 70, 71, 43,
20,122, 72, 61, 23,109, 13,100, 77, 1, 16, 7, 82, 10,105, 98,
117,116, 76, 11, 89,106, 0,125,118, 99, 86, 69, 30, 57,126, 87,
112, 51, 17, 5, 95, 14, 90, 84, 91, 8, 35,103, 32, 97, 28, 66,
102, 31, 26, 45, 75, 4, 85, 92, 37, 74, 80, 49, 68, 29,115, 44,
64,107,108, 24,110, 83, 36, 78, 42, 19, 15, 41, 88,119, 59, 3};
static u16 S9[] = {
167,239,161,379,391,334,  9,338, 38,226, 48,358,452,385, 90,397,
183,253,147,331,415,340, 51,362,306,500,262, 82,216,159,356,177,
175,241,489, 37,206, 17,  0,333, 44,254,378, 58,143,220, 81,400,
 95,  3,315,245, 54,235,218,405,472,264,172,494,371,290,399, 76,
165,197,395,121,257,480,423,212,240, 28,462,176,406,507,288,223,
501,407,249,265, 89,186,221,428,164, 74,440,196,458,421,350,163,
232,158,134,354, 13,250,491,142,191, 69,193,425,152,227,366,135,
344,300,276,242,437,320,113,278, 11,243, 87,317, 36, 93,496, 27,
487,446,482, 41, 68,156,457,131,326,403,339, 20, 39,115,442,124,
475,384,508, 53,112,170,479,151,126,169, 73,268,279,321,168,364,
363,292, 46,499,393,327,324, 24,456,267,157,460,488,426,309,229,
439,506,208,271,349,401,434,236, 16,209,359, 52, 56,120,199,277,
465,416,252,287,246,  6, 83,305,420,345,153,502, 65, 61,244,282,
173,222,418, 67,386,368,261,101,476,291,195,430, 49, 79,166,330,
280,383,373,128,382,408,155,495,367,388,274,107,459,417, 62,454,
132,225,203,316,234, 14,301, 91,503,286,424,211,347,307,140,374,
 35,103,125,427, 19,214,453,146,498,314,444,230,256,329,198,285,
 50,116, 78,410, 10,205,510,171,231, 45,139,467, 29, 86,505, 32,
 72, 26,342,150,313,490,431,238,411,325,149,473, 40,119,174,355,
185,233,389, 71,448,273,372, 55,110,178,322, 12,469,392,369,190,
  1,109,375,137,181, 88, 75,308,260,484, 98,272,370,275,412,111,
336,318,  4,504,492,259,304, 77,337,435, 21,357,303,332,483, 18,
 47, 85, 25,497,474,289,100,269,296,478,270,106, 31,104,433, 84,
414,486,394, 96, 99,154,511,148,413,361,409,255,162,215,302,201,
266,351,343,144,441,365,108,298,251, 34,182,509,138,210,335,133,
311,352,328,141,396,346,123,319,450,281,429,228,443,481, 92,404,
485,422,248,297, 23,213,130,466, 22,217,283, 70,294,360,419,127,
312,377,  7,468,194,  2,117,295,463,258,224,447,247,187, 80,398,
284,353,105,390,299,471,470,184, 57,200,348, 63,204,188, 33,451,
 97, 30,310,219, 94,160,129,493, 64,179,263,102,189,207,114,402,
438,477,387,122,192, 42,381,  5,145,118,180,449,293,323,136,380,
 43, 66, 60,455,341,445,202,432, 8,237, 15,376,436,464, 59,461};

/* The sixteen bit input is split into two unequal halves,  *
 * nine bits and seven bits - as is the subkey   */
```

```
nine  = (u16)(in>>7);
seven = (u16)(in&0x7F);

/* Now run the various operations */

nine  = (u16)(S9[nine]  ^ seven);
seven = (u16)(S7[seven] ^ (nine & 0x7F));

seven ^= (subkey>>9);
nine  ^= (subkey&0x1FF);

nine  = (u16)(S9[nine]  ^ seven);
seven = (u16)(S7[seven] ^ (nine & 0x7F));

in = (u16)((seven<<9) + nine);

return( in );
}


/*---------------------------------------------------------------------
 * FO()
 * The FO() function.
 * Transforms a 32-bit value.  Uses <index> to identify the
 * appropriate subkeys to use.
 *---------------------------------------------------------------------*/
static u32 FO( u32 in, int index )
{
u16 left, right;

/* Split the input into two 16-bit words */

left  = (u16)(in>>16);
right = (u16) in;

/* Now apply the same basic transformation three times         */

left ^= KOi1[index];
left  = FI( left, KIi1[index] );
left ^= right;

right ^= KOi2[index];
right  = FI( right, KIi2[index] );
right ^= left;

left ^= KOi3[index];
left  = FI( left, KIi3[index] );
left ^= right;

in = (((u32)right)<<16)+left;

return( in );
}

/*---------------------------------------------------------------------
 * FL()
 * The FL() function.
 * Transforms a 32-bit value.  Uses <index> to identify the
 * appropriate subkeys to use.
```

```
 *--------------------------------------------------------------------*/
static u32 FL( u32 in, int index )
{
u16 l, r, a, b;

/* split out the left and right halves */

l = (u16)(in>>16);
r = (u16)(in);

/* do the FL() operations */

a  = (u16) (l & KLi1[index]);
r ^= ROL16(a,1);

b  = (u16)(r | KLi2[index]);
l ^= ROL16(b,1);

/* put the two halves back together */

in = (((u32)l)<<16) + r;

return( in );
}


/*---------------------------------------------------------------------
 * Kasumi()
 * the Main algorithm (fig 1).  Apply the same pair of operations
 * four times.  Transforms the 64-bit input.
 *--------------------------------------------------------------------*/
void Kasumi( u8 *data )
{
u32 left, right, temp;
DWORD *d;
int n;

/* Start by getting the data into two 32-bit words (endian corect) */

d = (DWORD*)data;
left  = (((u32)d[0].b8[0])<<24)+(((u32)d[0].b8[1])<<16)
+(d[0].b8[2]<<8)+(d[0].b8[3]);
right = (((u32)d[1].b8[0])<<24)+(((u32)d[1].b8[1])<<16)
+(d[1].b8[2]<<8)+(d[1].b8[3]);
n = 0;
do{  temp = FL( left, n   );
temp = FO( temp,  n++ );
right ^= temp;
temp = FO( right, n   );
temp = FL( temp,   n++ );
left ^= temp;
}while( n<=7 );

/* return the correct endian result */
d[0].b8[0] = (u8)(left>>24); d[1].b8[0] = (u8)(right>>24);
d[0].b8[1] = (u8)(left>>16); d[1].b8[1] = (u8)(right>>16);
d[0].b8[2] = (u8)(left>>8); d[1].b8[2] = (u8)(right>>8);
d[0].b8[3] = (u8)(left); d[1].b8[3] = (u8)(right);
}
```

```
/*---------------------------------------------------------------------
 * KeySchedule()
 * Build the key schedule.  Most "key" operations use 16-bit
 * subkeys so we build u16-sized arrays that are "endian" correct.
 *---------------------------------------------------------------------*/
void KeySchedule( u8 *k )
{
static u16 C[] = {
0x0123,0x4567,0x89AB,0xCDEF, 0xFEDC,0xBA98,0x7654,0x3210 };
u16 key[8], Kprime[8];
WORD *k16;
int n;

/* Start by ensuring the subkeys are endian correct on a 16-bit basis */

k16 = (WORD *)k;
for( n=0; n<8; ++n )
key[n] = (u16)((k16[n].b8[0]<<8) + (k16[n].b8[1]));

/* Now build the K'[] keys */

for( n=0; n<8; ++n )
Kprime[n] = (u16)(key[n] ^ C[n]);

/* Finally construct the various sub keys */

for( n=0; n<8; ++n )
{
KLi1[n] = ROL16(key[n],1);
KLi2[n] = Kprime[(n+2)&0x7];
KOi1[n] = ROL16(key[(n+1)&0x7],5);
KOi2[n] = ROL16(key[(n+5)&0x7],8);
KOi3[n] = ROL16(key[(n+6)&0x7],13);
KIi1[n] = Kprime[(n+4)&0x7];
KIi2[n] = Kprime[(n+3)&0x7];
KIi3[n] = Kprime[(n+7)&0x7];
}
}
/*---------------------------------------------------------------------
 * e n d    o f    k a s u m i . c
 *---------------------------------------------------------------------*/
```

UEA2 - SNOW 3G

* Ο ίδιος αλγόριθμος υιοθετήθηκε για χρήση και στα δίκτυα 4ης γενιάς, όπου φέρει την ονομασία 128 - EEA1.

Header File

```
/*----------------------------------------------------------
* f8.h
*----------------------------------------------------------*/
#ifndef F8_H_
#define F8_H_
#include "SNOW_3G.h"
/* f8.
* Input key: 128 bit Confidentiality Key.
* Input count:32-bit Count, Frame dependent input.
* Input bearer: 5-bit Bearer identity (in the LSB side).
* Input dir:1 bit, direction of transmission.
* Input data: length number of bits, input bit stream.
* Input length: 32 bit Length, i.e., the number of bits to be encrypted or
* decrypted.
* Output data: Output bit stream. Assumes data is suitably memory
* allocated.
* Encrypts/decrypts blocks of data between 1 and 2^32 bits in length as
* defined in Section 3.
*/
void f8( u8 *key, u32 count, u32 bearer, u32 dir, u8 *data, u32 length );
#endif
```

Code

```
/*----------------------------------------------------------
* f8.c
*----------------------------------------------------------*/
#include "f8.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/* f8.
* Input key: 128 bit Confidentiality Key.
* Input count:32-bit Count, Frame dependent input.
* Input bearer: 5-bit Bearer identity (in the LSB side).
* Input dir:1 bit, direction of transmission.
* Input data: length number of bits, input bit stream.
* Input length: 32 bit Length, i.e., the number of bits to be encrypted or
* decrypted.
* Output data: Output bit stream. Assumes data is suitably memory
* allocated.
* Encrypts/decrypts blocks of data between 1 and 2^32 bits in length as
* defined in Section 3.
*/
void f8( u8 *key, u32 count, u32 bearer, u32 dir, u8 *data, u32 length )
{
u32 K[4],IV[4];
int n = ( length + 31 ) / 32;
int i=0;
u32 *KS;
/*Initialisation*/
/* Load the confidentiality key for SNOW 3G initialization as in section
3.4. */
for (i=0; i<4; i++)
```

```
K[3-i] = (key[4*i] << 24) ^ (key[4*i+1] << 16) ^ (key[4*i+2] << 8) ^
(key[4*i+3]);
3GPP Confidentiality and Integrity Algorithms UEA2&UIA2 page 24 of 27
UEA2 and UIA2 Specification Version 2.1
/* Prepare the initialization vector (IV) for SNOW 3G initialization as in
section 3.4. */
IV[3] = count;
IV[2] = (bearer << 27) | ((dir & 0x1) << 26);
IV[1] = IV[3];
IV[0] = IV[2];
/* Run SNOW 3G algorithm to generate sequence of key stream bits KS*/
Initialize(K,IV);
KS = (u32 *)malloc(4*n);
GenerateKeystream(n,(u32*)KS);
/* Exclusive-OR the input data with keystream to generate the output bit
stream */
for (i=0; i<n; i++)
{
data[4*i+0] ^= (u8) (KS[i] >> 24) & 0xff;
data[4*i+1] ^= (u8) (KS[i] >> 16) & 0xff;
data[4*i+2] ^= (u8) (KS[i] >> 8) & 0xff;
data[4*i+3] ^= (u8) (KS[i] ) & 0xff;
}
free(KS);
}
/* End of f8.c */
```

Header file

```
/*-----------------------------------------------------------
* SNOW_3G.h
*-----------------------------------------------------------*/
typedef unsigned char u8;
typedef unsigned int u32;
typedef unsigned long long u64;
/* Initialization.
* Input k[4]: Four 32-bit words making up 128-bit key.
* Input IV[4]: Four 32-bit words making 128-bit initialization variable.
* Output: All the LFSRs and FSM are initialized for key generation.
*/
void Initialize(u32 k[4], u32 IV[4]);
/* Generation of Keystream.
* input n: number of 32-bit words of keystream.
* input z: space for the generated keystream, assumes
* memory is allocated already.
* output: generated keystream which is filled in z
*/
void GenerateKeystream(u32 n, u32 *z);
```

Code

```
/*-----------------------------------------------------------------------
* SNOW_3G.c
*-----------------------------------------------------------------------*/
#include "SNOW_3G.h"
/* LFSR */
u32 LFSR_S0 = 0x00;
u32 LFSR_S1 = 0x00;
u32 LFSR_S2 = 0x00;
u32 LFSR_S3 = 0x00;
u32 LFSR_S4 = 0x00;
```

```
u32 LFSR_S5 = 0x00;
u32 LFSR_S6 = 0x00;
u32 LFSR_S7 = 0x00;
u32 LFSR_S8 = 0x00;
u32 LFSR_S9 = 0x00;
u32 LFSR_S10 = 0x00;
u32 LFSR_S11 = 0x00;
u32 LFSR_S12 = 0x00;
u32 LFSR_S13 = 0x00;
u32 LFSR_S14 = 0x00;
u32 LFSR_S15 = 0x00;
/* FSM */
u32 FSM_R1 = 0x00;
u32 FSM_R2 = 0x00;
u32 FSM_R3 = 0x00;

/* Rijndael S-box SR */
u8 SR[256] = {
0x63,0x7C,0x77,0x7B,0xF2,0x6B,0x6F,0xC5,0x30,0x01,0x67,0x2B,0xFE,0xD7,0xAB,0x76,
0xCA,0x82,0xC9,0x7D,0xFA,0x59,0x47,0xF0,0xAD,0xD4,0xA2,0xAF,0x9C,0xA4,0x72,0xC0,
0xB7,0xFD,0x93,0x26,0x36,0x3F,0xF7,0xCC,0x34,0xA5,0xE5,0xF1,0x71,0xD8,0x31,0x15,
0x04,0xC7,0x23,0xC3,0x18,0x96,0x05,0x9A,0x07,0x12,0x80,0xE2,0xEB,0x27,0xB2,0x75,
0x09,0x83,0x2C,0x1A,0x1B,0x6E,0x5A,0xA0,0x52,0x3B,0xD6,0xB3,0x29,0xE3,0x2F,0x84,
0x53,0xD1,0x00,0xED,0x20,0xFC,0xB1,0x5B,0x6A,0xCB,0xBE,0x39,0x4A,0x4C,0x58,0xCF,
0xD0,0xEF,0xAA,0xFB,0x43,0x4D,0x33,0x85,0x45,0xF9,0x02,0x7F,0x50,0x3C,0x9F,0xA8,
0x51,0xA3,0x40,0x8F,0x92,0x9D,0x38,0xF5,0xBC,0xB6,0xDA,0x21,0x10,0xFF,0xF3,0xD2,
0xCD,0x0C,0x13,0xEC,0x5F,0x97,0x44,0x17,0xC4,0xA7,0x7E,0x3D,0x64,0x5D,0x19,0x73,
0x60,0x81,0x4F,0xDC,0x22,0x2A,0x90,0x88,0x46,0xEE,0xB8,0x14,0xDE,0x5E,0x0B,0xDB,
0xE0,0x32,0x3A,0x0A,0x49,0x06,0x24,0x5C,0xC2,0xD3,0xAC,0x62,0x91,0x95,0xE4,0x79,
0xE7,0xC8,0x37,0x6D,0x8D,0xD5,0x4E,0xA9,0x6C,0x56,0xF4,0xEA,0x65,0x7A,0xAE,0x08,
0xBA,0x78,0x25,0x2E,0x1C,0xA6,0xB4,0xC6,0xE8,0xDD,0x74,0x1F,0x4B,0xBD,0x8B,0x8A,
0x70,0x3E,0xB5,0x66,0x48,0x03,0xF6,0x0E,0x61,0x35,0x57,0xB9,0x86,0xC1,0x1D,0x9E,
0xE1,0xF8,0x98,0x11,0x69,0xD9,0x8E,0x94,0x9B,0x1E,0x87,0xE9,0xCE,0x55,0x28,0xDF,
0x8C,0xA1,0x89,0x0D,0xBF,0xE6,0x42,0x68,0x41,0x99,0x2D,0x0F,0xB0,0x54,0xBB,0x16
};
/* S-box SQ */
u8 SQ[256] = {
0x25,0x24,0x73,0x67,0xD7,0xAE,0x5C,0x30,0xA4,0xEE,0x6E,0xCB,0x7D,0xB5,0x82,0xDB,
0xE4,0x8E,0x48,0x49,0x4F,0x5D,0x6A,0x78,0x70,0x88,0xE8,0x5F,0x5E,0x84,0x65,0xE2,
0xD8,0xE9,0xCC,0xED,0x40,0x2F,0x11,0x28,0x57,0xD2,0xAC,0xE3,0x4A,0x15,0x1B,0xB9,
0xB2,0x80,0x85,0xA6,0x2E,0x02,0x47,0x29,0x07,0x4B,0x0E,0xC1,0x51,0xAA,0x89,0xD4,
0xCA,0x01,0x46,0xB3,0xEF,0xDD,0x44,0x7B,0xC2,0x7F,0xBE,0xC3,0x9F,0x20,0x4C,0x64,
0x83,0xA2,0x68,0x42,0x13,0xB4,0x41,0xCD,0xBA,0xC6,0xBB,0x6D,0x4D,0x71,0x21,0xF4,
0x8D,0xB0,0xE5,0x93,0xFE,0x8F,0xE6,0xCF,0x43,0x45,0x31,0x22,0x37,0x36,0x96,0xFA,
0xBC,0x0F,0x08,0x52,0x1D,0x55,0x1A,0xC5,0x4E,0x23,0x69,0x7A,0x92,0xFF,0x5B,0x5A,
0xEB,0x9A,0x1C,0xA9,0xD1,0x7E,0x0D,0xFC,0x50,0x8A,0xB6,0x62,0xF5,0x0A,0xF8,0xDC,
0x03,0x3C,0x0C,0x39,0xF1,0xB8,0xF3,0x3D,0xF2,0xD5,0x97,0x66,0x81,0x32,0xA0,0x00,
0x06,0xCE,0xF6,0xEA,0xB7,0x17,0xF7,0x8C,0x79,0xD6,0xA7,0xBF,0x8B,0x3F,0x1F,0x53,
0x63,0x75,0x35,0x2C,0x60,0xFD,0x27,0xD3,0x94,0xA5,0x7C,0xA1,0x05,0x58,0x2D,0xBD,
0xD9,0xC7,0xAF,0x6B,0x54,0x0B,0xE0,0x38,0x04,0xC8,0x9D,0xE7,0x14,0xB1,0x87,0x9C,
0xDF,0x6F,0xF9,0xDA,0x2A,0xC4,0x59,0x16,0x74,0x91,0xAB,0x26,0x61,0x76,0x34,0x2B,
0xAD,0x99,0xFB,0x72,0xEC,0x33,0x12,0xDE,0x98,0x3B,0xC0,0x9B,0x3E,0x18,0x10,0x3A,
0x56,0xE1,0x77,0xC9,0x1E,0x9E,0x95,0xA3,0x90,0x19,0xA8,0x6C,0x09,0xD0,0xF0,0x86
};
/* MULx.
* Input V: an 8-bit input.
* Input c: an 8-bit input.
* Output : an 8-bit output.
*/
```

```
u8 MULx(u8 V, u8 c)
{
if ( V & 0x80 )
return ( (V << 1) ^ c);
else
return ( V << 1);
}
/* MULxPOW.
* Input V: an 8-bit input.
* Input i: a positive integer.
* Input c: an 8-bit input.
* Output : an 8-bit output.
*/
u8 MULxPOW(u8 V, u8 i, u8 c)



{
if ( i == 0)
return V;
else
return MULx( MULxPOW( V, i-1, c ), c);
}
/* The function MUL alpha.
* Input c: 8-bit input.
* Output : 32-bit output.
*/
u32 MULalpha(u8 c)
{
return ( ( ((u32)MULxPOW(c, 23, 0xa9)) << 24 ) |
( ((u32)MULxPOW(c, 245, 0xa9)) << 16 ) |
( ((u32)MULxPOW(c, 48, 0xa9)) << 8 ) |
( ((u32)MULxPOW(c, 239, 0xa9)) ) ) ;
}
/* The function DIV alpha.
* Input c: 8-bit input.
* Output : 32-bit output.
*/
u32 DIValpha(u8 c)
{
return ( ( ((u32)MULxPOW(c, 16, 0xa9)) << 24 ) |
( ((u32)MULxPOW(c, 39, 0xa9)) << 16 ) |
( ((u32)MULxPOW(c, 6, 0xa9)) << 8 ) |
( ((u32)MULxPOW(c, 64, 0xa9)) ) ) ;
}
/* The 32x32-bit S-Box S1
* Input: a 32-bit input.
* Output: a 32-bit output of S1 box.
*/
u32 S1(u32 w)
{
u8 r0=0, r1=0, r2=0, r3=0;
u8 srw0 = SR[ (u8)((w >> 24) & 0xff) ];
u8 srw1 = SR[ (u8)((w >> 16) & 0xff) ];
u8 srw2 = SR[ (u8)((w >> 8) & 0xff) ];
u8 srw3 = SR[ (u8)((w) & 0xff) ];
r0 = ( ( MULx( srw0 , 0x1b) ) ^
( srw1 ) ^
( srw2 ) ^
```

```
( (MULx( srw3, 0x1b)) ^ srw3 )
);
r1 = ( ( ( MULx( srw0 , 0x1b) ) ^ srw0 ) ^
( MULx(srw1, 0x1b) ) ^
( srw2 ) ^
( srw3 )
);
r2 = ( ( srw0 ) ^
( ( MULx( srw1 , 0x1b) ) ^ srw1 ) ^
( MULx(srw2, 0x1b) ) ^
( srw3 )
);
r3 = ( ( srw0 ) ^


( srw1 ) ^
( ( MULx( srw2 , 0x1b) ) ^ srw2 ) ^
( MULx( srw3, 0x1b) )
);
return ( ( ((u32)r0) << 24 ) | ( ((u32)r1) << 16 ) | ( ((u32)r2) << 8 ) |
( ((u32)r3) ) );
}
/* The 32x32-bit S-Box S2
* Input: a 32-bit input.
* Output: a 32-bit output of S2 box.
*/
u32 S2(u32 w)
{
u8 r0=0, r1=0, r2=0, r3=0;
u8 sqw0 = SQ[ (u8)((w >> 24) & 0xff) ];
u8 sqw1 = SQ[ (u8)((w >> 16) & 0xff) ];
u8 sqw2 = SQ[ (u8)((w >> 8) & 0xff) ];
u8 sqw3 = SQ[ (u8)((w) & 0xff) ];
r0 = ( ( MULx( sqw0 , 0x69) ) ^
( sqw1 ) ^
( sqw2 ) ^
( (MULx( sqw3, 0x69)) ^ sqw3 )
);
r1 = ( ( ( MULx( sqw0 , 0x69) ) ^ sqw0 ) ^
( MULx(sqw1, 0x69) ) ^
( sqw2 ) ^
( sqw3 )
);
r2 = ( ( sqw0 ) ^
( ( MULx( sqw1 , 0x69) ) ^ sqw1 ) ^
( MULx(sqw2, 0x69) ) ^
( sqw3 )
);
r3 = ( ( sqw0 ) ^
( sqw1 ) ^
( ( MULx( sqw2 , 0x69) ) ^ sqw2 ) ^
( MULx( sqw3, 0x69) )
);
return ( ( ((u32)r0) << 24 ) | ( ((u32)r1) << 16 ) | ( ((u32)r2) << 8 ) |
( ((u32)r3) ) );
}
/* Clocking LFSR in initialization mode.
* LFSR Registers S0 to S15 are updated as the LFSR receives a single clock.
* Input F: a 32-bit word comes from output of FSM.
```

```
*/
void ClockLFSRInitializationMode(u32 F)
{
u32 v = ( ( (LFSR_S0 << 8) & 0xffffff00 ) ^
( MULalpha( (u8)((LFSR_S0>>24) & 0xff) ) ) ^
( LFSR_S2 ) ^
( (LFSR_S11 >> 8) & 0x00ffffff ) ^
( DIValpha( (u8)( ( LFSR_S11) & 0xff ) ) ) ^
( F )
);
LFSR_S0 = LFSR_S1;
LFSR_S1 = LFSR_S2;
LFSR_S2 = LFSR_S3;

LFSR_S3 = LFSR_S4;
LFSR_S4 = LFSR_S5;
LFSR_S5 = LFSR_S6;
LFSR_S6 = LFSR_S7;
LFSR_S7 = LFSR_S8;
LFSR_S8 = LFSR_S9;
LFSR_S9 = LFSR_S10;
LFSR_S10 = LFSR_S11;
LFSR_S11 = LFSR_S12;
LFSR_S12 = LFSR_S13;
LFSR_S13 = LFSR_S14;
LFSR_S14 = LFSR_S15;
LFSR_S15 = v;
}
/* Clocking LFSR in keystream mode.
* LFSR Registers S0 to S15 are updated as the LFSR receives a single clock.
*/
void ClockLFSRKeyStreamMode()
{
u32 v = ( ( (LFSR_S0 << 8) & 0xffffff00 ) ^
( MULalpha( (u8)((LFSR_S0>>24) & 0xff) ) ) ^
( LFSR_S2 ) ^
( (LFSR_S11 >> 8) & 0x00ffffff ) ^
( DIValpha( (u8)( ( LFSR_S11) & 0xff ) ) )
);
LFSR_S0 = LFSR_S1;
LFSR_S1 = LFSR_S2;
LFSR_S2 = LFSR_S3;
LFSR_S3 = LFSR_S4;
LFSR_S4 = LFSR_S5;
LFSR_S5 = LFSR_S6;
LFSR_S6 = LFSR_S7;
LFSR_S7 = LFSR_S8;
LFSR_S8 = LFSR_S9;
LFSR_S9 = LFSR_S10;
LFSR_S10 = LFSR_S11;
LFSR_S11 = LFSR_S12;
LFSR_S12 = LFSR_S13;
LFSR_S13 = LFSR_S14;
LFSR_S14 = LFSR_S15;
LFSR_S15 = v;
}
/* Clocking FSM.
* Produces a 32-bit word F.
* Updates FSM registers R1, R2, R3.
```

```
*/
u32 ClockFSM()
{
u32 F = ( ( LFSR_S15 + FSM_R1 ) & 0xffffffff ) ^ FSM_R2 ;
u32 r = ( FSM_R2 + ( FSM_R3 ^ LFSR_S5 ) ) & 0xffffffff ;
FSM_R3 = S2(FSM_R2);
FSM_R2 = S1(FSM_R1);
FSM_R1 = r;
return F;
}
/* Initialization.
* Input k[4]: Four 32-bit words making up 128-bit key.
* Input IV[4]: Four 32-bit words making 128-bit initialization variable.
* Output: All the LFSRs and FSM are initialized for key generation.
*/
void Initialize(u32 k[4], u32 IV[4])
{
u8 i=0;
u32 F = 0x0;
LFSR_S15 = k[3] ^ IV[0];
LFSR_S14 = k[2];
LFSR_S13 = k[1];
LFSR_S12 = k[0] ^ IV[1];
LFSR_S11 = k[3] ^ 0xffffffff;
LFSR_S10 = k[2] ^ 0xffffffff ^ IV[2];
LFSR_S9 = k[1] ^ 0xffffffff ^ IV[3];
LFSR_S8 = k[0] ^ 0xffffffff;
LFSR_S7 = k[3];
LFSR_S6 = k[2];
LFSR_S5 = k[1];
LFSR_S4 = k[0];
LFSR_S3 = k[3] ^ 0xffffffff;
LFSR_S2 = k[2] ^ 0xffffffff;
LFSR_S1 = k[1] ^ 0xffffffff;
LFSR_S0 = k[0] ^ 0xffffffff;
FSM_R1 = 0x0;
FSM_R2 = 0x0;
FSM_R3 = 0x0;
for(i=0;i<32;i++)
{
F = ClockFSM();
ClockLFSRInitializationMode(F);
}
}
/* Generation of Keystream.
* input n: number of 32-bit words of keystream.
* input z: space for the generated keystream, assumes
* memory is allocated already.
* output: generated keystream which is filled in z
*/
void GenerateKeystream(u32 n, u32 *ks)
{
u32 t = 0;
u32 F = 0x0;
ClockFSM(); /* Clock FSM once. Discard the output. */
ClockLFSRKeyStreamMode(); /* Clock LFSR in keystream mode once. */
for ( t=0; t<n; t++)
{
F = ClockFSM(); /* STEP 1 */
```

```
ks[t] = F ^ LFSR_S0; /* STEP 2 */
/* Note that ks[t] corresponds to z_{t+1} in section 4.2
*/
ClockLFSRKeyStreamMode(); /* STEP 3 */
}
}
/*--------------------------------------------------------------------------*/
```

* output: generated keystream which is filled in z

RIJNDAEL

*Όπως ήδη αναφέρθηκε ο αλγόριθμος Rijndael αποτέλεσε τη βάση για την ανάπτυξη του AES.

Header file

```
    #ifdef __cplusplus
extern "C" {
#endif

typedef enum {
    AES_CYPHER_128,
    AES_CYPHER_192,
    AES_CYPHER_256,
} AES_CYPHER_T;

#ifdef _MSC_VER
    #if _MSC_VER >= 1600
        #include <cstdint>
    #else
        typedef __int8              int8_t;
        typedef __int16             int16_t;
        typedef __int32             int32_t;
        typedef __int64             int64_t;
        typedef unsigned __int8     uint8_t;
        typedef unsigned __int16    uint16_t;
        typedef unsigned __int32    uint32_t;
        typedef unsigned __int64    uint64_t;
    #endif
#elif __GNUC__ >= 3
    #include <cstdint>
#endif

int aes_encrypt_ecb(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key);
int aes_decrypt_ecb(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key);
int aes_encrypt_cbc(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key, uint8_t *iv);
int aes_decrypt_cbc(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key, uint8_t *iv);

#ifdef __cplusplus
};
#endif
```

Code

```
    #include <stdio.h>
#include <memory.h>

#include "rijndael.h"

//
// Public Definitions
//

/* moved to rijndael.h */

//
// Internal Definitions
//

/*
 * Encryption Rounds
 */
```

```
int g_aes_key_bits[] = {
    /* AES_CYPHER_128 */ 128,
    /* AES_CYPHER_192 */ 192,
    /* AES_CYPHER_256 */ 256,
};

int g_aes_rounds[] = {
    /* AES_CYPHER_128 */  10,
    /* AES_CYPHER_192 */  12,
    /* AES_CYPHER_256 */  14,
};

int g_aes_nk[] = {
    /* AES_CYPHER_128 */  4,
    /* AES_CYPHER_192 */  6,
    /* AES_CYPHER_256 */  8,
};

int g_aes_nb[] = {
    /* AES_CYPHER_128 */  4,
    /* AES_CYPHER_192 */  4,
    /* AES_CYPHER_256 */  4,
};



/*
 * aes Rcon:
 *
 * WARNING: Rcon is designed starting from 1 to 15, not 0 to 14.
 *          FIPS-197 Page 9: "note that i starts at 1, not 0"
 *
 * i    |   0     1     2     3     4     5     6     7     8     9    10    11    12    13    14
 * -----+-------------------------------------------------------------------------------------------
 *      | [01]  [02]  [04]  [08]  [10]  [20]  [40]  [80]  [1b]  [36]  [6c]  [d8]  [ab]  [4d]  [9a]
 * RCON | [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]
 *      | [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]
 *      | [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]  [00]
 */

static const uint32_t g_aes_rcon[] = {
    0x01000000, 0x02000000, 0x04000000, 0x08000000, 0x10000000, 0x20000000, 0x40000000, 0x80000000,
    0x1b000000, 0x36000000, 0x6c000000, 0xd8000000, 0xab000000, 0xed000000, 0x9a000000
};

/* aes sbox and invert-sbox */
static const uint8_t g_aes_sbox[256] = {
 /* 0     1     2     3     4     5     6     7     8     9     A     B     C     D     E     F  */
    0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,
    0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0,
    0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15,
    0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75,
    0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84,
    0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf,
    0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f, 0xa8,
    0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3, 0xd2,
    0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19, 0x73,
    0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b, 0xdb,
    0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4, 0x79,
```

```
    0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae, 0x08,
    0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b, 0x8a,
    0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d, 0x9e,
    0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28, 0xdf,
    0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb, 0x16
};

static const uint8_t g_inv_sbox[256] = {
 /* 0     1     2     3     4     5     6     7     8     9     A     B     C     D     E     F  */
    0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3, 0x9e, 0x81, 0xf3, 0xd7, 0xfb,
    0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43, 0x44, 0xc4, 0xde, 0xe9, 0xcb,
    0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95, 0x0b, 0x42, 0xfa, 0xc3, 0x4e,
    0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2, 0x49, 0x6d, 0x8b, 0xd1, 0x25,
    0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c, 0xcc, 0x5d, 0x65, 0xb6, 0x92,
    0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46, 0x57, 0xa7, 0x8d, 0x9d, 0x84,
    0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58, 0x05, 0xb8, 0xb3, 0x45, 0x06,
    0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd, 0x03, 0x01, 0x13, 0x8a, 0x6b,
    0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf, 0xce, 0xf0, 0xb4, 0xe6, 0x73,
    0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37, 0xe8, 0x1c, 0x75, 0xdf, 0x6e,
    0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62, 0x0e, 0xaa, 0x18, 0xbe, 0x1b,
    0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0, 0xfe, 0x78, 0xcd, 0x5a, 0xf4,
    0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10, 0x59, 0x27, 0x80, 0xec, 0x5f,
    0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a, 0x9f, 0x93, 0xc9, 0x9c, 0xef,
    0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb, 0x3c, 0x83, 0x53, 0x99, 0x61,
    0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14, 0x63, 0x55, 0x21, 0x0c, 0x7d
};

uint8_t aes_sub_sbox(uint8_t val)
{
    return g_aes_sbox[val];
}

uint32_t aes_sub_dword(uint32_t val)
{
    uint32_t tmp = 0;

    tmp |= ((uint32_t)aes_sub_sbox((uint8_t)((val >>  0) & 0xFF))) <<  0;
    tmp |= ((uint32_t)aes_sub_sbox((uint8_t)((val >>  8) & 0xFF))) <<  8;
    tmp |= ((uint32_t)aes_sub_sbox((uint8_t)((val >> 16) & 0xFF))) << 16;
    tmp |= ((uint32_t)aes_sub_sbox((uint8_t)((val >> 24) & 0xFF))) << 24;

    return tmp;
}

uint32_t aes_rot_dword(uint32_t val)
{
    uint32_t tmp = val;

    return (val >> 8) | ((tmp & 0xFF) << 24);
}

uint32_t aes_swap_dword(uint32_t val)
{
    return (((val & 0x000000FF) << 24) |
            ((val & 0x0000FF00) <<  8) |
            ((val & 0x00FF0000) >>  8) |
            ((val & 0xFF000000) >> 24) );
}
```

```
/*
 * nr: number of rounds
 * nb: number of columns comprising the state, nb = 4 dwords (16 bytes)
 * nk: number of 32-bit words comprising cipher key, nk = 4, 6, 8 (KeyLength/(4*8))
 */

void aes_key_expansion(AES_CYPHER_T mode, uint8_t *key, uint8_t *round)
{
    uint32_t *w = (uint32_t *)round;
    uint32_t  t;
    int      i = 0;

    printf("Key Expansion:\n");
    do {
        w[i] = *((uint32_t *)&key[i * 4 + 0]);
        printf("    %2.2d:  rs: %8.8x\n", i, aes_swap_dword(w[i]));
    } while (++i < g_aes_nk[mode]);

    do {
        printf("    %2.2d: ", i);
        if ((i % g_aes_nk[mode]) == 0) {
            t = aes_rot_dword(w[i - 1]);
            printf(" rot: %8.8x", aes_swap_dword(t));
            t = aes_sub_dword(t);
            printf(" sub: %8.8x", aes_swap_dword(t));
            printf(" rcon: %8.8x", g_aes_rcon[i/g_aes_nk[mode] - 1]);
            t = t ^ aes_swap_dword(g_aes_rcon[i/g_aes_nk[mode] - 1]);
            printf(" xor: %8.8x", t);
        } else if (g_aes_nk[mode] > 6 && (i % g_aes_nk[mode]) == 4) {
            t = aes_sub_dword(w[i - 1]);
            printf(" sub: %8.8x", aes_swap_dword(t));
        } else {
            t = w[i - 1];
            printf(" equ: %8.8x", aes_swap_dword(t));
        }
        w[i] = w[i - g_aes_nk[mode]] ^ t;
        printf(" rs: %8.8x\n", aes_swap_dword(w[i]));
    } while (++i < g_aes_nb[mode] * (g_aes_rounds[mode] + 1));

    /* key can be discarded (or zeroed) from memory */
}

void aes_add_round_key(AES_CYPHER_T mode, uint8_t *state,
                       uint8_t *round, int nr)
{
    uint32_t *w = (uint32_t *)round;
    uint32_t *s = (uint32_t *)state;
    int i;

    for (i = 0; i < g_aes_nb[mode]; i++) {
        s[i] ^= w[nr * g_aes_nb[mode] + i];
    }
}

void aes_sub_bytes(AES_CYPHER_T mode, uint8_t *state)
{
    int i, j;

    for (i = 0; i < g_aes_nb[mode]; i++) {
```

```
        for (j = 0; j < 4; j++) {
            state[i * 4 + j] = aes_sub_sbox(state[i * 4 + j]);
        }
    }
}

void aes_shift_rows(AES_CYPHER_T mode, uint8_t *state)
{
    uint8_t *s = (uint8_t *)state;
    int i, j, r;

    for (i = 1; i < g_aes_nb[mode]; i++) {
        for (j = 0; j < i; j++) {
            uint8_t tmp = s[i];
            for (r = 0; r < g_aes_nb[mode]; r++) {
                s[i + r * 4] = s[i + (r + 1) * 4];
            }
            s[i + (g_aes_nb[mode] - 1) * 4] = tmp;
        }
    }
}

uint8_t aes_xtime(uint8_t x)
{
    return ((x << 1) ^ (((x >> 7) & 1) * 0x1b));
}

uint8_t aes_xtimes(uint8_t x, int ts)
{
    while (ts-- > 0) {
        x = aes_xtime(x);
    }

    return x;
}

uint8_t aes_mul(uint8_t x, uint8_t y)
{
    /*
     * encrypt: y has only 2 bits: can be 1, 2 or 3
     * decrypt: y could be any value of 9, b, d, or e
     */

    return ((((y >> 0) & 1) * aes_xtimes(x, 0)) ^
            (((y >> 1) & 1) * aes_xtimes(x, 1)) ^
            (((y >> 2) & 1) * aes_xtimes(x, 2)) ^
            (((y >> 3) & 1) * aes_xtimes(x, 3)) ^
            (((y >> 4) & 1) * aes_xtimes(x, 4)) ^
            (((y >> 5) & 1) * aes_xtimes(x, 5)) ^
            (((y >> 6) & 1) * aes_xtimes(x, 6)) ^
            (((y >> 7) & 1) * aes_xtimes(x, 7)) );
}

void aes_mix_columns(AES_CYPHER_T mode, uint8_t *state)
{
    uint8_t y[16] = { 2, 3, 1, 1,  1, 2, 3, 1,  1, 1, 2, 3,  3, 1, 1, 2};
    uint8_t s[4];
    int i, j, r;
```

```
    for (i = 0; i < g_aes_nb[mode]; i++) {
        for (r = 0; r < 4; r++) {
            s[r] = 0;
            for (j = 0; j < 4; j++) {
                s[r] = s[r] ^ aes_mul(state[i * 4 + j], y[r * 4 + j]);
            }
        }
        for (r = 0; r < 4; r++) {
            state[i * 4 + r] = s[r];
        }
    }
}


void aes_dump(char *msg, uint8_t *data, int len)
{
    int i;

    printf("%8.8s: ", msg);
    for (i = 0; i < len; i++) {
        printf(" %2.2x", data[i]);
    }
    printf("\n");
}

int aes_encrypt(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key)
{
    uint8_t w[4 * 4 * 15] = {0}; /* round key */
    uint8_t s[4 * 4] = {0}; /* state */

    int nr, i, j;

    /* key expansion */
    aes_key_expansion(mode, key, w);

    /* start data cypher loop over input buffer */
    for (i = 0; i < len; i += 4 * g_aes_nb[mode]) {

        printf("Encrypting block at %u ...\n", i);

        /* init state from user buffer (plaintext) */
        for (j = 0; j < 4 * g_aes_nb[mode]; j++)
            s[j] = data[i + j];

        /* start AES cypher loop over all AES rounds */
        for (nr = 0; nr <= g_aes_rounds[mode]; nr++) {

            printf(" Round %d:\n", nr);
            aes_dump("input", s, 4 * g_aes_nb[mode]);

            if (nr > 0) {

                /* do SubBytes */
                aes_sub_bytes(mode, s);
                aes_dump("  sub", s, 4 * g_aes_nb[mode]);

                /* do ShiftRows */
                aes_shift_rows(mode, s);
                aes_dump("  shift", s, 4 * g_aes_nb[mode]);
```

```
            if (nr < g_aes_rounds[mode]) {
                /* do MixColumns */
                aes_mix_columns(mode, s);
                aes_dump("  mix", s, 4 * g_aes_nb[mode]);
            }
        }

        /* do AddRoundKey */
        aes_add_round_key(mode, s, w, nr);
        aes_dump("  round", &w[nr * 4 * g_aes_nb[mode]], 4 * g_aes_nb[mode]);
        aes_dump("  state", s, 4 * g_aes_nb[mode]);
    }

    /* save state (cypher) to user buffer */
    for (j = 0; j < 4 * g_aes_nb[mode]; j++)
        data[i + j] = s[j];
    printf("Output:\n");
    aes_dump("cypher", &data[i], 4 * g_aes_nb[mode]);
}

    return 0;
}

int aes_encrypt_ecb(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key)
{
    return aes_encrypt(mode, data, len, key);
}

int aes_encrypt_cbc(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key, uint8_t *iv)
{
    uint8_t w[4 * 4 * 15] = {0}; /* round key */
    uint8_t s[4 * 4] = {0}; /* state */
    uint8_t v[4 * 4] = {0}; /* iv */

    int nr, i, j;


    /* key expansion */
    aes_key_expansion(mode, key, w);

    memcpy(v, iv, sizeof(v));

    /* start data cypher loop over input buffer */
    for (i = 0; i < len; i += 4 * g_aes_nb[mode]) {

        /* init state from user buffer (plaintext) */
        for (j = 0; j < 4 * g_aes_nb[mode]; j++)
            s[j] = data[i + j] ^ v[j];

        /* start AES cypher loop over all AES rounds */
        for (nr = 0; nr <= g_aes_rounds[mode]; nr++) {

            aes_dump("input", s, 4 * g_aes_nb[mode]);

            if (nr > 0) {

                /* do SubBytes */
                aes_sub_bytes(mode, s);
```

```
                aes_dump("  sub", s, 4 * g_aes_nb[mode]);

                /* do ShiftRows */
                aes_shift_rows(mode, s);
                aes_dump("  shift", s, 4 * g_aes_nb[mode]);

                if (nr < g_aes_rounds[mode]) {
                    /* do MixColumns */
                    aes_mix_columns(mode, s);
                    aes_dump("  mix", s, 4 * g_aes_nb[mode]);
                }
            }

            /* do AddRoundKey */
            aes_add_round_key(mode, s, w, nr);
            aes_dump("  round", &w[nr * 4 * g_aes_nb[mode]], 4 * g_aes_nb[mode]);
            aes_dump("  state", s, 4 * g_aes_nb[mode]);
        }

        /* save state (cypher) to user buffer */
        for (j = 0; j < 4 * g_aes_nb[mode]; j++)
            data[i + j] = v[j] = s[j];
    }

    return 0;
}

void inv_shift_rows(AES_CYPHER_T mode, uint8_t *state)
{
    uint8_t *s = (uint8_t *)state;
    int i, j, r;

    for (i = 1; i < g_aes_nb[mode]; i++) {
        for (j = 0; j < g_aes_nb[mode] - i; j++) {
            uint8_t tmp = s[i];
            for (r = 0; r < g_aes_nb[mode]; r++) {
                s[i + r * 4] = s[i + (r + 1) * 4];
            }
            s[i + (g_aes_nb[mode] - 1) * 4] = tmp;
        }
    }
}

uint8_t inv_sub_sbox(uint8_t val)
{
    return g_inv_sbox[val];
}


void inv_sub_bytes(AES_CYPHER_T mode, uint8_t *state)
{
    int i, j;

    for (i = 0; i < g_aes_nb[mode]; i++) {
        for (j = 0; j < 4; j++) {
            state[i * 4 + j] = inv_sub_sbox(state[i * 4 + j]);
        }
    }
}
```

```
void inv_mix_columns(AES_CYPHER_T mode, uint8_t *state)
{
    uint8_t y[16] = { 0x0e, 0x0b, 0x0d, 0x09,  0x09, 0x0e, 0x0b, 0x0d,
                      0x0d, 0x09, 0x0e, 0x0b,  0x0b, 0x0d, 0x09, 0x0e};
    uint8_t s[4];
    int i, j, r;

    for (i = 0; i < g_aes_nb[mode]; i++) {
        for (r = 0; r < 4; r++) {
            s[r] = 0;
            for (j = 0; j < 4; j++) {
                s[r] = s[r] ^ aes_mul(state[i * 4 + j], y[r * 4 + j]);
            }
        }
        for (r = 0; r < 4; r++) {
            state[i * 4 + r] = s[r];
        }
    }
}

int aes_decrypt(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key)
{
    uint8_t w[4 * 4 * 15] = {0}; /* round key */
    uint8_t s[4 * 4] = {0}; /* state */

    int nr, i, j;

    /* key expansion */
    aes_key_expansion(mode, key, w);

    /* start data cypher loop over input buffer */
    for (i = 0; i < len; i += 4 * g_aes_nb[mode]) {

        printf("Decrypting block at %u ...\n", i);

        /* init state from user buffer (cyphertext) */
        for (j = 0; j < 4 * g_aes_nb[mode]; j++)
            s[j] = data[i + j];

        /* start AES cypher loop over all AES rounds */
        for (nr = g_aes_rounds[mode]; nr >= 0; nr--) {

            printf(" Round %d:\n", nr);
            aes_dump("input", s, 4 * g_aes_nb[mode]);

            /* do AddRoundKey */
            aes_add_round_key(mode, s, w, nr);
            aes_dump("  round", &w[nr * 4 * g_aes_nb[mode]], 4 * g_aes_nb[mode]);


            if (nr > 0) {

                if (nr < g_aes_rounds[mode]) {
                    aes_dump("  mix", s, 4 * g_aes_nb[mode]);
                    /* do MixColumns */
                    inv_mix_columns(mode, s);
                }
```

```
                /* do ShiftRows */
                aes_dump("  shift", s, 4 * g_aes_nb[mode]);
                inv_shift_rows(mode, s);

                /* do SubBytes */
                aes_dump("  sub", s, 4 * g_aes_nb[mode]);
                inv_sub_bytes(mode, s);
            }

            aes_dump("  state", s, 4 * g_aes_nb[mode]);
        }

        /* save state (cypher) to user buffer */
        for (j = 0; j < 4 * g_aes_nb[mode]; j++)
            data[i + j] = s[j];
        printf("Output:\n");
        aes_dump("plain", &data[i], 4 * g_aes_nb[mode]);
    }

    return 0;
}

int aes_decrypt_ecb(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key)
{
    return aes_decrypt(mode, data, len, key);
}

int aes_decrypt_cbc(AES_CYPHER_T mode, uint8_t *data, int len, uint8_t *key, uint8_t *iv)
{
    uint8_t w[4 * 4 * 15] = {0}; /* round key */
    uint8_t s[4 * 4] = {0}; /* state */
    uint8_t v[4 * 4] = {0}; /* iv */


    int nr, i, j;

    /* key expansion */
    aes_key_expansion(mode, key, w);

    memcpy(v, iv, sizeof(v));

    /* start data cypher loop over input buffer */
    for (i = 0; i < len; i += 4 * g_aes_nb[mode]) {


        /* init state from user buffer (cyphertext) */
        for (j = 0; j < 4 * g_aes_nb[mode]; j++)
            s[j] = data[i + j];

        /* start AES cypher loop over all AES rounds */
        for (nr = g_aes_rounds[mode]; nr >= 0; nr--) {

            aes_dump("input", s, 4 * g_aes_nb[mode]);

            /* do AddRoundKey */
            aes_add_round_key(mode, s, w, nr);
            aes_dump("  round", &w[nr * 4 * g_aes_nb[mode]], 4 * g_aes_nb[mode]);
```

```
            if (nr > 0) {

                if (nr < g_aes_rounds[mode]) {
                    aes_dump("  mix", s, 4 * g_aes_nb[mode]);
                    /* do MixColumns */
                    inv_mix_columns(mode, s);
                }

                /* do ShiftRows */
                aes_dump("  shift", s, 4 * g_aes_nb[mode]);
                inv_shift_rows(mode, s);

                /* do SubBytes */
                aes_dump("  sub", s, 4 * g_aes_nb[mode]);
                inv_sub_bytes(mode, s);
            }

            aes_dump("  state", s, 4 * g_aes_nb[mode]);
        }

        /* save state (cypher) to user buffer */
        for (j = 0; j < 4 * g_aes_nb[mode]; j++) {
            uint8_t p = s[j] ^ v[j];
            v[j] = data[i + j];
            data[i + j] = p;
        }
    }

    return 0;
}

void aes_cypher_128_test()
{
#if 1
    uint8_t buf[] = { 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
                      0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff };
    uint8_t key[] = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                      0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f };
#else
    uint8_t buf[] = { 0x32, 0x43, 0xf6, 0xa8, 0x88, 0x5a, 0x30, 0x8d,
                      0x31, 0x31, 0x98, 0xa2, 0xe0, 0x37, 0x07, 0x34 };
    uint8_t key[] = { 0x2b, 0x7e, 0x15, 0x16, 0x28, 0xae, 0xd2, 0xa6,
                      0xab, 0xf7, 0x15, 0x88, 0x09, 0xcf, 0x4f, 0x3c };
#endif
    printf("\nAES_CYPHER_128 encrypt test case:\n");
    printf("Input:\n");
    aes_dump("data", buf, sizeof(buf));
    aes_dump("key ",  key, sizeof(key));
    aes_encrypt(AES_CYPHER_128, buf, sizeof(buf), key);

    printf("\nAES_CYPHER_128 decrypt test case:\n");
    printf("Input:\n");
    aes_dump("data", buf, sizeof(buf));
    aes_dump("key ",  key, sizeof(key));
    aes_decrypt(AES_CYPHER_128, buf, sizeof(buf), key);
}

void aes_cypher_192_test()
{
```

```
    uint8_t buf[] = { 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
                      0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff };
    uint8_t key[] = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                      0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
                      0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17 };
    printf("\nAES_CYPHER_192 encrypt test case:\n");
    printf("Input:\n");
    aes_dump("data", buf, sizeof(buf));
    aes_dump("key ",  key, sizeof(key));
    aes_encrypt(AES_CYPHER_192, buf, sizeof(buf), key);

    printf("\nAES_CYPHER_192 decrypt test case:\n");
    printf("Input:\n");
    aes_dump("data", buf, sizeof(buf));
    aes_dump("key ",  key, sizeof(key));
    aes_decrypt(AES_CYPHER_192, buf, sizeof(buf), key);
}

void aes_cypher_256_test()
{
    uint8_t buf[] = { 0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
                      0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff };
    uint8_t key[] = { 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07,
                      0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
                      0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
                      0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f};
    printf("\nAES_CYPHER_256 encrypt test case:\n");
    printf("Input:\n");
    aes_dump("data", buf, sizeof(buf));
    aes_dump("key ",  key, sizeof(key));
    aes_encrypt(AES_CYPHER_256, buf, sizeof(buf), key);

    printf("\nAES_CYPHER_256 decrypt test case:\n");
    printf("Input:\n");
    aes_dump("data", buf, sizeof(buf));
    aes_dump("key ",  key, sizeof(key));
    aes_decrypt(AES_CYPHER_256, buf, sizeof(buf), key);
}
```

Βιβλιογραφία

[1] Ν. Μπάρδης. *Κρυπτογραφία και Συστήματα Ασφαλείας*. Πανεπιστημιακές Σημειώσεις, Αθήνα, 2013.

[2] Γ. Βασιλόγιαννης. *Σύνθεση και Αξιολόγηση Γεννητριών Ψευδοτυχαίων Ακολουθιών*. Αθήνα, 2019.

[3] Θ. Γιαννόπουλος. *Ασφαλής και αποδοτική υλοποίηση της αριθμητικής υπολοίπων στον RSA σε IoT εφαρμογές*. Αθήνα, 2019.

[4] Ε. Ζάχος, Α. Παγουρτζής, and Π. Γρόντας. *Υπολογιστική Κρυπτογραφία*. Εθνικό Μετσόβιο Πολυτεχνείο, 2015.

[5] Κ. Λυκούδης. *Συμμετρικοί Αλγόριθμοι Κρυπτογράφησης Δεδομένων - Η περίπτωση του Αλγορίθμου AES*. Πάτρα, 2012.

[6] Δ. Φλωκατούλα. *Συμμετρικοί Αλγόριθμοι Κρυπτογράφησης Δεδομένων - Οι Περιπτώσεις των Αλγορίθμων DES και TDEA*. Πάτρα, 2012.

[7] O. Damgaard Jensen and K. Alvern Andersen. *A5 Encryption in GSM*, 2017.

[8] ETSI. *Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 1: f8 and f9 Specification*, 2006.

[9] ETSI. *Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: Kasumi Specification*, 2006.

[10] ETSI/SAGE. *Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 and UIA2, Document 1: UEA2 and UIA2 Specification*, 2006.

[11] ETSI/SAGE. *Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 and UIA2, Document 2: SNOW 3G Specification*, 2006.

[12] ETSI. *3G Security; Security Architecture (Release 14)*, 2017.

[13] ETSI. *Specification of the A5/4 Encryption Algorithms for GSM and ECSD, and the GEA4 Encryption Algorithm for GPRS (Release 14)*, 2017.

[14] ETSI. *3GPP System Architecture Evolution; Security Architecture (Release 14)*, 2018.

[15] Fakhri Al-Shaikhli I. Fadhil Jasim, K. *Analysis of Confidentiality Algorithms in Different Mobile Generations*. 2017.

[16] Federal Information Processing Standards. *Advanced Encryption Standard (AES)*, 2001.

[17] D. Forsberg, G. Horn, W.-D. Moeller, and V. Niemi. *LTE Security*. 2013.

[18] A. Ghanim Sulaiman and I. Fakhri Al Shaikhli. *Comparative Study on 4G/LTE Cryptographic Algorithms Based on Different Factors*. 2014.

[19] A. Menezes, P. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[20] D. Mishra Sandip and Dr. Nilesh K. Modi. *False base station attack in GSM Network Environment*. 2014.

[21] National Institute of Standards and Technology. *Recommendation for Block Cipher Modes of Operation*, 2001.

[22] G. Orhanou and S. El-Hajji. *The New LTE Cryptographic Algorithms EEA3 and EIA3.* 2013.

[23] G. Orhanou, S. El-Hajji, and Y. Bentaleb. Snow 3g stream cipher operation and complexity study. 2010.

[24] G. Rose. Authentication and security in mobile phones. 2006.

[25] M. Sipser. *Εισαγωγή στη Θεωρία Υπολογισμού.* Πανεπιστημιακές Εκδόσεις Κρήτης, 2011.

[26] WhatsApp. *WhatsApp Encryption Overview*, 2017.

[27] L. Xian and W. Tingthanathikul. *Advanced Encryption Standard (AES) in Counter Mode*, 2004.

[28] 3GPP. *GSM Association Specification for A5/3*, 2001.

[29] 3GPP. *Study on the support of 256-bit algorithms for 5G (Release 16).* 2018.

[30] Τεχνολογία Επικοινωνιών. *http://comtechamfissa.pbworks.com.*

[31] Cryptography. *https://crypto.stackexchange.com.*

[32] Github. *https://github.com.*

[33] GSMK Cryptophone. *https://www.cryptophone.de.*

[34] Heng Kiong. *https://toughnickel.com.*

[35] Hybrid Encryption Schemes. *https://cryptobook.nakov.com.*

[36] People @ EECS at UC Berkeley. *https://people.eecs.berkeley.edu/.*

[37] The Free Encyplopedia Wikipedia. *https://www.wikipedia.org.*

[38] ZDNet. *https://www.zdnet.com.*