Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
Στην Ασφάλεια Υπολογιστών και Δικτύων

Κρυπτογραφικές ιδιότητες των γεννητριών ακολουθιών De Bruijn.

Ιωάννης Κολλητίδης

Επιβλέπων Καθηγητής
Κωνσταντίνος Λιμνιώτης

Μάιος 2018

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Κρυπτογραφικές ιδιότητες των γεννητριών ακολουθιών De Bruijn.

Ιωάννης Κολλητίδης

Επιβλέπων Καθηγητής
Κωνσταντίνος Λιμνιώτης

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στην Ασφάλεια Υπολογιστών και Δικτύων

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Μάιος 2018

ΛΕΥΚΗ ΣΕΛΙΔΑ

Περίληψη

Οι δυαδικές ακολουθίες De Bruijn αποτελούν μια σημαντική οικογένεια ακολουθιών με πάρα πολλές εφαρμογές, συμπεριλαμβανομένης της κρυπτογραφίας. Σε κρυπτογραφικές εφαρμογές, πρόκειται για ακολουθίες που παράγονται από μη γραμμικούς καταχωρητές ολίσθησης με ανάδραση (NLFSRs) μεγέθους n, οποίοι κατά τη λειτουργία τους «διατρέχουν» όλες τις πιθανές καταστάσεις (maximal length NLFSRs). Οι NLFSRs εφαρμόζονται σε κρυπταλγορίθμους ροής (stream ciphers), οι οποίοι εμφανίζουν ιδιότητες που τους επιτρέπουν να χρησιμοποιηθούν σε εφαρμογές με απαιτήσεις για χαμηλή κατανάλωση ενέργειας, και για μικρή επιφάνεια ανάπτυξης υλικού – π.χ. σε IoT εφαρμογές.

Οι συναρτήσεις ανάδρασης εκείνων των NLFSRs που παράγουν ακολουθίες De Bruijn δεν έχουν μελετηθεί μέχρι τώρα εκτενώς στην ερευνητική κοινότητα. Για παράδειγμα, δεν είναι ακόμα γνωστή συγκεκριμένη μεθοδολογία κατασκευής NLFSRs που παράγει εγγυημένα τέτοιες ακολουθίες μεγίστης περιόδου. Σκοπός της διατριβής είναι ο προσδιορισμός κρυπτογραφικών χαρακτηριστικών - όπως η μη γραμμικότητα, η αλγεβρική ανθεκτικότητα, ανθεκτικότητα στις συσχετίσεις, αλλά και η ύπαρξη γραμμικών δομών - των λογικών συναρτήσεων οι οποίες, όταν αποτελούν συναρτήσεις ανάδρασης ενός NLFSR, παράγουν ακολουθίες De Bruijn. Απώτερος στόχος είναι η διερεύνηση των συσχετίσεων μεταξύ των κρυπτογραφικών αυτών ιδιοτήτων για τις περιπτώσεις συναρτήσεων οι οποίες αντιστοιχούν σε ακολουθίες De Bruijn που παράγονται η μία από την άλλη μέσω γνωστών μαθηματικών τεχνικών.

Στο πλαίσιο της διατριβής, με ανάπτυξη και αξιοποίηση κατάλληλων εφαρμογών λογισμικού, μελετήθηκαν τυχαία παραγόμενες ακολουθίες De Bruijn ως προς τις τιμές των κρυπτογραφικών ιδιοτήτων των αντίστοιχων συναρτήσεών τους, ενώ επίσης μελετήθηκαν το πώς «αντανακλώνται» οι κρυπτογραφικές αυτές ιδιότητες σε συναρτήσεις που αντιστοιχούν σε άλλες De Bruijn ακολουθίες, οι οποίες προκύπτουν με κάποια μαθηματική τεχνική από μία ή δύο άλλες αρχικές ακολουθίες De Bruijn. Τα πειραματικά αποτελέσματα καταδεικνύουν ότι είναι πολύ δύσκολο να βρεθεί συνάρτηση που να παράγει ακολουθία De Bruijn και να ικανοποιεί συγκεκριμένα κρυπτογραφικά κριτήρια (όπως η ανθεκτικότητα σε συσχετίσεις). Ωστόσο, τα αποτελέσματά μας δείχνουν ότι είναι εφικτή, αξιοποιώντας τις γνωστές μαθηματικές τεχνικές

κατασκευής ακολουθιών De Bruijn, η βελτίωση συγκεκριμένων κρυπτογραφικών κριτηρίων συναρτήσεων που παράγουν ακολουθίες De Bruijn.

Summary

Binary De Bruijn sequences constitute an important family of sequences with many applications, including cryptography. In cryptographic applications, such sequences correspond to sequences produced by the so-called maximal-length Non-Linear Feedback Shift Register (NLFSR), i.e. NLFSRs that pass through all the possible states. NLFSRs are main building blocks in stream ciphers, which in turn have properties that allow them to be used in applications that require low energy consumption and small layout area - e.g. in IoT applications.

The feedback functions of those NLFSRs that produce De Bruijn sequences have not been studied extensively till now in the research community. For example, it is still unknown how to construct a NLFSR that is bound to generate De Bruijn sequences. The aim of this thesis is the study of cryptographic characteristics - such as nonlinearity, algebraic immunity, correlation immunity and linear structure's existence - of those Boolean functions that, when forming feedback functions of an NLFSR, the corresponding output sequence is a De Bruijn sequence. More precisely, our ultimate goal is to investigate whether there exist correlations between these cryptographic properties for Boolean functions that correspond to De Bruijn sequences produced from each other using known mathematical techniques.

To this end, randomly generated De Bruijn sequences have been studied, in terms of the behavior of the corresponding Boolean functions with regard to their cryptographic properties. Moreover, we examined how these cryptographic properties are "reflected" in functions corresponding to other De Bruijn sequences, which are obtained by one or two other De Bruijn sequences via applying known mathematical techniques. The experimental results demonstrate that it is very difficult to establish a Boolean function which produces De Bruijn sequences meeting specific cryptographic criteria (such as correlation immunity). However, current study's results show it is feasible to improve specific cryptographic criteria of Boolean functions generating De Bruijn sequences, whilst ensuring that the new functions still generate De Bruijn sequences.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες προς τον επιβλέποντα καθηγητή κ. Κωνσταντίνο Λιμνιώτη που συνέβαλε τα μέγιστα για την ορθή ολοκλήρωση της διατριβής.

Την οικογένεια μου.

Περιεχόμενα

Κεφάλαιο 1
Εισαγωγή

Η κρυπτογραφία μελετά τεχνικές με τις οποίες ένα μήνυμα μπορεί να μετασχηματιστεί σε ακατάληπτη μορφή. Η διαδικασία μετατροπής ενός μηνύματος σε ακατάληπτη μορφή ονομάζεται κρυπτογράφηση ενώ η αντίστροφη διαδικασία ονομάζεται αποκρυπτογράφηση. Κύριος στόχος της κρυπτογραφίας είναι η εμπιστευτικότητα του μηνύματος.

Στην εποχή μας ο όρος κρυπτογραφία είναι πολύ πιο ευρύς: συγκεκριμένα, αναφερόμαστε στη μελέτη μαθηματικών τεχνικών που έχουν σαν στόχους την διασφάλιση διαφόρων ζητημάτων που άπτονται της ασφάλειας της πληροφορίας όπως είναι η εμπιστευτικότητα, γνησιότητα χρηστών και πληροφοριών, ακεραιότητα των δεδομένων καθώς και η μη αποποίηση γεγονότων που έχουν συμβεί. Διάφοροι κρυπτογραφικοί αλγόριθμοι ή κρυπτογραφικές τεχνικές αναπτύχθηκαν για την διασφάλιση των παραπάνω ζητημάτων.

Με τον όρο κρυπτανάλυση αναφερόμαστε στην μελέτη μαθηματικών τεχνικών που στοχεύουν στην ακύρωση των κρυπτογραφικών μεθόδων προκειμένου να πληγεί η ασφάλεια του. Με τον όρο κρυπτολογία αναφερόμαστε στις μεθόδους της κρυπτογραφίας και της κρυπτανάλυσης.

Ένα κρυπτοσύστημα αποτελείται από δύο αλγόριθμους, έναν αλγόριθμο κρυπτογράφησης (encryption algorithm) E και έναν αλγόριθμο αποκρυπτογράφησης (decryption algorithm) D. Το αρχικό κείμενο (plain text) είναι το κείμενο προς κρυπτογράφηση. Χρησιμοποιώντας το αρχικό κείμενο για είσοδο του αλγόριθμου κρυπτογράφησης , παίρνουμε στην έξοδο το κρυπτοκείμενο (cipher text). Ο αλγόριθμος αποκρυπτογράφησης χρησιμοποιεί για είσοδο το κρυπτοκείμενο και εξάγει το αντίστοιχο αρχικό κείμενο. Γενικά τα κρυπτοσυστήματα μπορούν να ταξινομηθούν με βάση τον αριθμό κλειδιών που χρησιμοποιούνται.

- Κανένα κλειδί : Όλο το κρυπτοσύστημα βασίζεται στον αλγόριθμο κρυπτογράφησης και αποκρυπτογράφησης. Ο χρησιμοποιούμενος αλγόριθμος θα πρέπει να κρατείται μυστικός, δηλαδή να είναι γνωστός μόνο στους ιδιοκτήτες του κρυπτοσυστήματος.

- Ένα κλειδί : Οι αλγόριθμοι κρυπτογράφησης E και αποκρυπτογράφησης D χρειάζονται μια παράμετρο K, η οποία ουσιαστικά είναι το μυστικό κλειδί (κρυπτογράφησης – αποκρυπτογράφησης). Οι αλγόριθμοι κρυπτογράφησης και αποκρυπτογράφησης μπορούν να κοινοποιηθούν αλλά η παράμετρος K (μυστικό κλειδί) θα πρέπει να παραμείνει μυστικό.

- Δύο κλειδιά : Οι αλγόριθμοι κρυπτογράφησης – αποκρυπτογράφησης χρησιμοποιούν διαφορετικές παραμέτρους (κλειδιά). Ο αλγόριθμος κρυπτογράφησης χρησιμοποιεί το κλειδί κρυπτογράφησης K, το οποίο είναι δημόσιο, και ο αλγόριθμος αποκρυπτογράφησης χρησιμοποιεί το κλειδί αποκρυπτογράφησης K′ που θα πρέπει να είναι ιδιωτικό.

1.1 Ιστορική αναδρομή

Η κρυπτογραφία χρησιμοποιείται πάνω από 2500 χρόνια από τον άνθρωπο. Στην αρχαία Ελλάδα οι έφοροι στην Σπάρτη επικοινωνούσαν με τους στρατηγούς χρησιμοποιώντας μακριές, στενές κορδέλες οι οποίες τυλίγονταν γύρω από έναν κύλινδρο (σκυτάλη) και στην συνέχεια έγραφαν το μήνυμα κατά μήκους του κυλίνδρου. Για να διαβάσει κανείς το μήνυμα, έπρεπε να έχει μια παρόμοια σκυτάλη που είχε χρησιμοποιηθεί έτσι ώστε να τυλίξει την σκυτάλη με τον ίδιο τρόπο. Ο Ιούλιος Καίσαρας χρησιμοποιούσε ένα κρυπτοσύστημα για να επικοινωνήσει με τους συμμάχους του, με το οποίο το κρυπτογραφημένο κείμενο προέκυπτε με την αντικατάσταση του κάθε γράμματος με ένα άλλο σύμφωνα με ολίσθηση των

γραμμάτων κατά βήμα Κ στο αλφάβητο[39]. Το παραπάνω κρυπτοσύστημα είναι από τα πιο απλά και ανασφαλή κρυπτοσυστήματα που έχουν προταθεί.

Οι πρώτες δημοσιεύσεις σχετικά με την κρυπτογραφία εμφανίστηκαν το 1518 από τον αββά Ιωάννη Τριθέμιο[35] και αργότερα δημοσιεύτηκε το «Περί κρυπτικών συμβόλων και γραμμάτων»[27] από τον Τζανμπατίστα ντε λα Πόρτα 1563. Από τότε η κρυπτογραφία έγινε αντικείμενο ιδιαίτερου ενδιαφέροντος. Πολύ φιλόσοφοι ενδιαφέρθηκαν για την κρυπτογραφία, όπως ο Σερ Φράνσις Μπέικον (1561 – 1626)[09] ο οποίος επινόησε ένα σύστημα κρυπτογράφησης όπου κάθε γράμμα αντικαθίσταται με μια λέξη πέντε γραμμάτων καθώς επίσης και ο Λεονάρντο Ντα Βίντσι (1452 – 1519) χρησιμοποίησε μια μέθοδο κρυπτογράφησης με καθρέπτη.

Ο Έντγκαρ Άλαν Πόε (1809 – 1849) στο διήγημα του «Το χρυσό έντομο»[34] εξηγεί τις βασικές αρχές αποκρυπτογράφησης κωδικών, υποστηρίζοντας ότι ο ανθρώπινος νους μπορεί να σπάσει οποιονδήποτε κρυπτογραφημένο κείμενο μπορεί να επινοήσει.

Επίσης ένα από τα γνωστότερα κρυπτογραφημένα κείμενα είναι το κρυπτογραφημένο τηλεγράφημα του Ζίμερμαν (Zimmerman Note)[38] και το οποίο ώθησε τις ΗΠΑ στον πρώτο παγκόσμιο πόλεμο. Το τηλεγράφημα αποκρυπτογραφήθηκε το 1917, οι Αμερικάνικες δυνάμεις έμαθαν ότι η Γερμανία είχε προσπαθήσει να πείσει το Μεξικό να μπει στο πόλεμο με το μέρος της , υποσχόμενη παραχωρήσεις εδαφών των ΗΠΑ στο Μεξικό.

Λίγο αργότερα ο Γκιλμπερτ Βέρναμ , μηχανικός της AT&T ανέπτυξε το πρώτο πραγματικά άθραυστο κώδικα που ονομάστηκε κρυπτογράφηση Βέρναμ[36]. Μια ιδιότητα αυτού του κώδικα είναι η απαίτηση για κλειδί με μήκος όσο και το μήνυμα που πρέπει να μεταδοθεί και το οποίο δεν ξαναχρησιμοποιείται για την αποστολή άλλου μηνύματος. Η ανακάλυψη του συστήματος αυτού δεν εκτιμήθηκε ιδιαίτερα εκείνη την εποχή, γιατί δεν είχε αποδειχθεί ότι είναι άθραυστος καθώς επίσης και η απαίτηση για ένα κλειδί όσο και το μήκος του μηνύματος τον έκαναν πρακτικά μη χρήσιμο.

Η περίφημη μηχανή Enigma[28] χρησιμοποιήθηκε από τους Γερμανούς κατά το δεύτερο παγκόσμιο πόλεμο για κρυπτογράφηση των ραδιοτηλεπικοινωνιών και πυροδότησε μια από τις πιο έντονες προσπάθειες αποκρυπτογράφησης στην ιστορία. Μια βασική ιδιότητα της μηχανής αυτής ήταν η αυτο-αντιστροφή. Αυτό-αντιστροφή σημαίνει ότι εάν το κωδικοποιημένο μήνυμα δινόταν ως είσοδος στην μηχανή , τότε η έξοδος θα ήταν το αρχικό

μήνυμα , με την προϋπόθεση ότι η μηχανή είχε την ίδια αρχική κατάσταση με την μηχανή που είχε κάνει την κωδικοποίηση.

Η σημερινή μορφή των κρυπτογραφικών συστημάτων έχει καθοριστεί σε πολύ μεγάλο βαθμό από δύο επιστημονικές εργασίες του Kerchoff (1883) και του Shannon (1949). Ο Auguste Kerchoff[30, 15] έθεσε την βασική σχεδιαστική αρχή που έκτοτε διέπει κάθε κρυπτογραφικό σύστημα, σύμφωνα με την οποία η ασφάλεια ενός συστήματος πρέπει να έγκειται μόνο στην μυστικότητα του κλειδιού και να μην εξαρτάται από την μυστικότητα του αλγόριθμου κρυπτογράφησης. Η δεύτερη εργασία ανήκει στον Claude Shannon[05] με την οποία η κρυπτογραφία μετατρέπεται σε αυστηρό επιστημονικό πεδίο , όπου ορίζεται η έννοια του κρυπτοσυστήματος και η απόλυτη ασφάλεια. Η εργασία του Shannon αποτέλεσε το έναυσμα για την ταχεία εξέλιξη της έρευνας στο χώρο της κρυπτογραφίας και η οποία συνεχίζεται μέχρι σήμερα. Όλοι οι σύγχρονοι κρυπτογραφικοί αλγόριθμοι σχεδιάζονται με βάση τις έννοιες που εισήγαγε ο Shannon.

1.2 Σύγχρονοι αλγόριθμοι κρυπτογράφησης

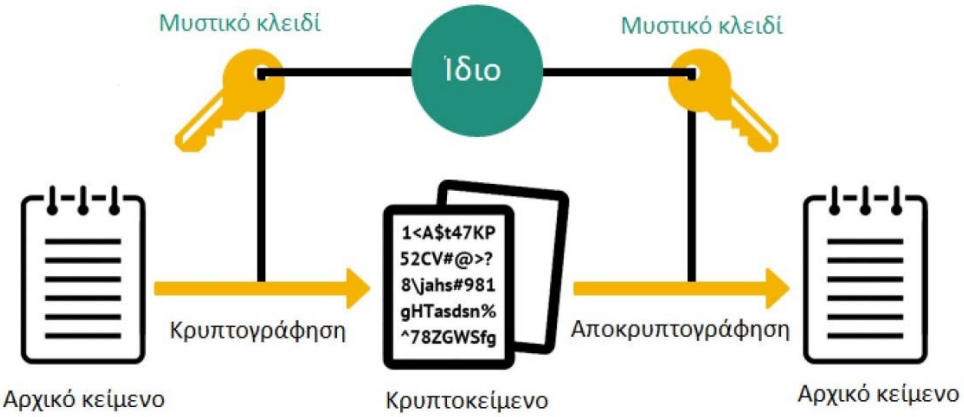
Η κρυπτογραφία είναι παρούσα σε μια πληθώρα εφαρμογών και τις οποίες συναντούμε καθημερινά. Κρυπτογραφία εφαρμόζεται στις ασφαλείς συναλλαγές με τις τράπεζες (ATM), στην κινητή τηλεφωνία (3G, 4G), ασύρματα δίκτυα (802.1X, Bluetooth), ηλεκτρονικό ταχυδρομείο, τηλεφωνία μέσω διαδικτύου (VoIP), ηλεκτρονικό εμπόριο (πληρωμές μέσω πιστωτικών καρτών) , εφαρμογές IoT αλλά και πολλές άλλες.

Οι κρυπτογραφικοί αλγόριθμοι χωρίζονται σε δύο μεγάλες βασικές κατηγορίες: τους συμμετρικούς (symmetric) και τους ασύμμετρους (asymmetric) ή γνωστούς και ως δημοσίου κλειδιού (public key).

1.3 Συμμετρικοί αλγόριθμοι κρυπτογράφησης

Στους συμμετρικούς αλγόριθμους το κλειδί κρυπτογράφησης ταυτίζεται με το κλειδί της αποκρυπτογράφησης. Ως εκ τούτου, αυτοί οι αλγόριθμοι χρειάζονται την εκ των προτέρων συμφωνία μεταξύ του αποστολέα και του παραλήπτη για το κλειδί που θα χρησιμοποιηθεί, για να μπορέσουν να επικοινωνήσουν με ασφάλεια. Η ασφάλεια των αλγόριθμων βασίζεται στην μυστικότητα αυτού του κλειδιού. Για όσο καιρό επιθυμούμε η επικοινωνία να παραμείνει

μυστική, για τον ίδιο καιρό πρέπει και το κλειδί να παραμείνει μυστικό. Μπορούμε να δούμε στη εικόνα 1.1 τη διαδικασία της κρυπτογράφησης και αποκρυπτογράφησης.



Εικόνα 1.1: Λειτουργία συμμετρικού αλγόριθμου κρυπτογράφησης

Οι συμμετρικοί αλγόριθμοι μπορούν να ταξινομηθούν σε δύο υποκατηγορίες :

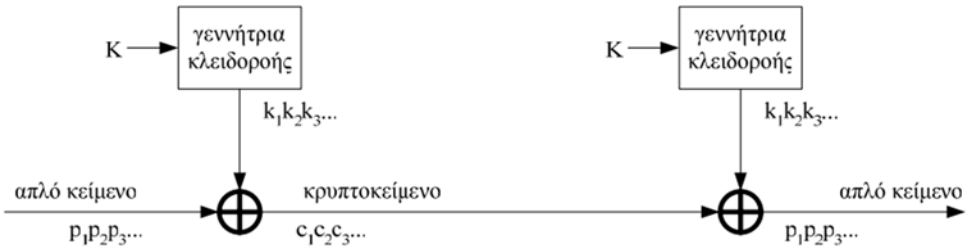
α) Αλγόριθμοι ροής (stream ciphers), οι οποίοι λειτουργούν bit προς bit και β) οι αλγόριθμοι τμήματος (block ciphers) οι οποίοι λειτουργούν πάνω σε «κομμάτια» δεδομένων (τμήματα).

1.3.1 Κρυπτογραφικός αλγόριθμος ροής

Οι κρυπτογραφικοί αλγόριθμοι ροής χρησιμοποιούνται ευρέως σε εφαρμογές με απαιτήσεις για υψηλή ταχύτητα και χαμηλή κατανάλωση ισχύος. Χαρακτηριστικά παραδείγματα είναι τα ασύρματα δίκτυα (WiFi, GSM, 3G), πρωτόκολλο Bluetooth , δίκτυα RFID, στο διαδίκτυο (αλγόριθμοι RC4, ChaCha20, SSL/TLS) καθώς επίσης γίνεται εκτενής χρήση στις συσκευές IoT (Internet of Things).

Το Internet of Things είναι μια εντελώς νέα τεχνολογία, η οποία έχει τραβήξει το ερευνητικό ενδιαφέρον της επιστημονικής κοινότητας καθώς επίσης του τύπου και των social media. Δεν υπάρχει ένας σαφής ορισμός του IoT: συνήθως αναφερόμαστε στην δυνατότητα δικτύωσης και διασύνδεσης διαφόρων υπολογιστικών αλλά όχι μόνο συσκευών. Βασικός σκοπός είναι ο συνδυασμός διαφορετικών τεχνολογιών μεταξύ τους όπως είναι υπολογιστές, δίκτυα, οθόνες, σένσορες και άλλα με απώτερο στόχο την συνδυασμένη λειτουργίας τους. Όμως εκτός από τη χρησιμότητα των εφαρμογών και των υπηρεσιών των IoT υπάρχει μια μεγάλη πρόσκληση στην λειτουργία των IoT συσκευών που είναι η ασφάλειά τους. Γι' αυτό το λόγο γίνεται εκτεταμένη χρήση των κρυπταλγόριθμων ροής για τη διασφάλιση την ακεραιότητας της πληροφορίας.

Οι κρυπταλγόριθμοι ροής ενεργούν σε ένα σύμβολο ή bit την φορά. Η λειτουργία τους φαίνεται στην παρακάτω Εικόνα 1.2. Σε ένα δυαδικό κρυπτοσύστημα η αλφάβητος του απλού κειμένου p_i, της κλειδοροής k_i και του κρυπτοκειμένου c_i αποτελούνται από bits.



Εικόνα 1.2: Λειτουργία συμμετρικού αλγόριθμου κρυπτογράφησης

Βασικό συστατικό ενός κρυπταλγόριθμου ροής είναι η γεννήτρια κλειδοροής. Η κλειδοροή είναι μια περιοδική ακολουθία με την οποία γίνεται η πρόσθεση XOR του κειμένου που θέλουμε να κρυπτογραφηθεί ώστε να παραχθεί το κρυπτοκείμενο. Η αρχή της περιόδου της κλειδοροής καθορίζεται από το κλειδί K το οποίο είναι και το μυστικό κλειδί του κρυπτοσυστήματος. Δύο είναι οι βασικοί λόγοι της περιοδικής φύσης της κλειδοροής.

Ο πρώτος λόγος είναι ότι θα πρέπει η γεννήτρια της κλειδοροής να έχει την δυνατότητα να παράγει την ίδια ακολουθία σε δύο διαφορετικές τοποθεσίες την ίδια χρονική στιγμή. Επομένως η υλοποίηση των γεννητριών αυτών πρέπει να γίνεται με συσκευές οι οποίες μπορούν αξιόπιστα να αναπαράγουν την ίδια ακολουθία. Δεύτερον οι μόνες γνωστές συσκευές σήμερα που μπορούν να ανταπεξέλθουν στον περιορισμό αυτό είναι οι μηχανές πεπερασμένων καταστάσεων, στην κατηγορία των οποίων ανήκουν και οι ηλεκτρονικοί υπολογιστές.

1.3.2 Σημειωματάριο μια χρήσης – τέλειος κρυπτογραφικός αλγόριθμος ροής

Το σημειωματάριο μιας χρήσης (γνωστό και ως κρυπτοσύστημα του Vernam) είναι ένα τέλειο κρυπτογραφικό σύστημα που παρέχει την απόλυτη μυστικότητα. Με τον όρο απόλυτη μυστικότητα εννοούμε ότι το κρυπτοκείμενο δεν αποκαλύπτει καμία πληροφορία για το αρχικό κείμενο. Ο κρυπταναλυτής έχει την ίδια πληροφορία για το αρχικό κείμενο είτε γνωρίζοντας το κρυπτοκείμενο είτε όχι, αυτό ισχύει μάλιστα ακόμα και εάν γνωρίζει ένα μέρος του αρχικού μηνύματος.

Το σημειωματάριο μιας χρήσης είναι ένα κρυπτοσύστημα μυστικού κλειδιού όπου το κλειδί έχει το ίδιο μήκος με το κείμενο προς κρυπτογράφηση. Το μυστικό κλειδί χρησιμοποιείται μόνο μια φορά και δεν επαναχρησιμοποιείται. Το αρχικό κείμενο M αναπαρίσταται ως δυαδική ακολουθία , όπως επίσης και το κλειδί K. Το κρυπτοκείμενο C προκύπτει από την πρόσθεση XOR (πρόσθεση modulo 2) του αρχικού κειμένου με το κλειδί : $C = M \oplus K$. Η αποκρυπτογράφηση γίνεται από $M = C \oplus K$. Αποδεικνύεται ότι είναι αδύνατο για τον κρυπταναλυτή να σπάσει τον κώδικα που χρησιμοποιεί σημειωματάριο μιας χρήσης. Οποιοδήποτε κρυπτοκείμενο C δεν αποκαλύπτει καμιά πληροφορία για το αρχικό κείμενο M, αφού κάθε μήνυμα M θα μπορούσε να παράγει το C , αν το κλειδί K ήταν ίσο με $K = C \oplus M$.

Η κρυπτογράφηση με σημειωματάριο μιας χρήσης είναι αποδεδειγμένα ασφαλής, ως προς την έννοια της πληροφορίας, αφού ο υποκλοπέας δεν έχει ποτέ αρκετή πληροφορία για να αποκρυπτογραφήσει το κρυπτοκείμενο και καμία ποσότητα υπολογιστικής δύναμης δεν μπορεί να το υπολογίσει. Όμως το σημειωματάριο μιας χρήσης δεν είναι πρακτικά εφικτό αφού ένα μεγάλο κλειδί (ουσιαστικά όσο και η ποσότητα της πληροφορίας) πρέπει να δημιουργηθεί, να διανεμηθεί και να αποθηκευτεί.

Στόχος των κρυπταλγόριθμων ροής είναι να προσομοιάσουν όσον το δυνατό περισσότερο την λειτουργία του σημειωματάριου μιας χρήσης.

1.4 Ασύμμετροι αλγόριθμοι κρυπτογράφησης

Οι ασύμμετροι αλγόριθμοι είναι σχεδιασμένοι έτσι ώστε το κλειδί που χρησιμοποιούν για την κρυπτογράφηση να είναι διαφορετικό από το κλειδί για την αποκρυπτογράφηση. Οι αλγόριθμοι αυτοί ονομάζονται και δημοσίου κλειδιού γιατί το κλειδί της κρυπτογράφησης μπορεί να δημοσιοποιηθεί. Ο καθένας μπορεί να κρυπτογραφήσει ένα μήνυμα με το δημόσιο κλειδί του παραλήπτη αλλά μόνο ο κάτοχος του ιδιωτικού κλειδιού – δηλαδή, ο παραλήπτης - μπορεί να το αποκρυπτογραφήσει. Παραδείγματα ασύμμετρων αλγορίθμων κρυπτογράφησης είναι ο RSA, αλγόριθμοι ελλειπτικών καμπυλών, El Gamal και άλλοι.

Οι συμμετρικοί αλγόριθμοι είναι πολύ πιο γρήγοροι εφαρμοσμένοι είτε σε υλικό ή λογισμικό από τους ασύμμετρους. Γι' αυτό το λόγο οι συμμετρικοί αλγόριθμοι χρησιμοποιούνται για την κρυπτογράφηση του κυρίως μέρους των δεδομένων του μηνύματος, ενώ οι αλγόριθμοι

δημοσίου κλειδιού προτιμώνται σε πρωτόκολλα ανταλλαγής κλειδιών και ψηφιακών υπογραφών (τα οποία εκ φεύγουν του αντικειμένου της παρούσας διατριβής).

1.5 Οι ακολουθίες De Bruijn στους κρυπταλγόριθμους ροής

Οι καταχωρητές ολίσθησης με ανάδραση (Feedback Shift Register - FSR) είναι οι βασικοί δομικοί λίθοι των περισσότερων κρυπταλγόριθμων ροής. Για παράδειγμα, στους αλγόριθμους ροής Grain και Trivium έχουν χρησιμοποιηθεί γραμμικοί καταχωρητές με ανάδραση ως βασικό συστατικό της σχεδίασής τους. Η θεωρία των μη γραμμικών καταχωρητών ολίσθησης με ανάδραση (Non Linear Feedback Shift Register - NLFSR) δεν έχει ερευνηθεί εις βάθος ακόμα. Πολλά από τα γνωστά αποτελέσματα αναφέρονται στο βιβλίο του Golomb [12]. Εξαιτίας της αποδοτικής εφαρμογής των NLFSR στο υλικό (hardware) έχουν μια ευρεία χρήση σε περιβάλλοντα με περιορισμένους πόρους (IoT, RFID, sensor).

Στον σχεδιασμό γεννητριών κλειδοροών στους κρυπταλγόριθμους ροής , ένας NLFSR δεν μπορεί να χρησιμοποιηθεί από μόνος του γιατί οι ιδιότητες τυχαιότητας της ακολουθίας που παράγεται δεν είναι γνωστές και δύσκολο να προσδιοριστούν. Μια από τις πιο κλασσικές προσεγγίσεις της χρήσης NLFSR σε γεννήτρια κλειδοροής είναι ο συνδυασμός του NLFSR με έναν γραμμικό καταχωρητή ανάδρασης με ολίσθηση (LFSR), όπου έτσι μπορεί να προσδιοριστεί με ακρίβεια η περίοδος της κλειδοροής.

Μια δυαδική ακολουθία de Bruijn τάξης n, είναι μια ακολουθία με περίοδο 2^n όπου κάθε n-άδα της ακολουθίας εμφανίζεται ακριβώς μία φορά μέσα στην περίοδο της ακολουθίας. Άρα, από τον ορισμό αυτό γίνεται σαφές ότι μια de Bruijn ακολουθία τάξης n είναι κάθε ακολουθία που παράγεται από έναν NLFSR μεγέθους n ο οποίος παράγει ακολουθίες με τη μέγιστη δυνατή περίοδο 2^n , το οποίο είναι κρυπτογραφικά επιθυμητό [10, 11, 02].

1.6 Σκοπός έρευνας και βασικά ερευνητικά ερωτήματα

Σκοπός της έρευνας είναι η μελέτη των κρυπτογραφικών ιδιοτήτων που έχουν οι συναρτήσεις εκείνες οι οποίες παράγουν ακολουθίες De Bruijn. Ουσιαστικά, μελετούμε τις κρυπτογραφικές

ιδιότητες (μη γραμμικότητα, ανθεκτικότητα σε αλγεβρικές επιθέσεις κτλ.) που έχουν οι συναρτήσεις εκείνες οι οποίες, όταν αποτελέσουν συνάρτηση ανάδρασης ενός μη γραμμικού καταχωρητή ολίσθησης με ανάδραση (NLFSR), παράγεται ακολουθία De Bruijn, Προς το σκοπό αυτό, σε κάθε De Bruijn συνάρτηση αντιστοιχούμε μονοσήμαντα μια λογική συνάρτηση (Boolean Function), έτσι ώστε να μπορέσουμε να μελετήσουμε διάφορα κρυπτογραφικά χαρακτηριστικά αυτής. Επίσης σκοπός μας είναι η μελέτη των κρυπτογραφικών αυτών ιδιοτήτων σε περιπτώσεις όπου μία De Bruijn ακολουθία παράγεται – με κάποια εκ των γνωστών μαθηματικών τεχνικών - από κάποια άλλη (μία ή περισσότερες) De Bruijn ακολουθία, υπό την έννοια ότι εξετάζουμε αν υπάρχει συσχέτιση των κρυπτογραφικών ιδιοτήτων των συναρτήσεων που αντιστοιχούν στις αρχικές ακολουθίες de Bruijn με τις αντίστοιχες ιδιότητες των συναρτήσεων που αντιστοιχούν στις νέες ακολουθίες de Bruijn.

Βασικά ερευνητικά ερωτήματα που θα τεθούν είναι τα ακόλουθα:

1) Ποιες κρυπτογραφικές ιδιότητες έχει μια λογική συνάρτηση η οποία παράγει de Bruijn ακολουθία;

2) Εάν δημιουργηθεί μια νέα de Bruijn ακολουθία από μια ή δύο άλλες de Bruijn οι κρυπτογραφικές ιδιότητες της συναρτήσεων που αντιστοιχούν στις αρχικές de Bruijn «αντανακλώνται» στην καινούργια συνάρτηση;

Τα ερωτήματα συνεισφέρουν αφενός στο να προσδιοριστούν οι περιορισμοί που ενδεχομένως υπάρχουν κατά την κατασκευή γεννητριών ακολουθιών De Bruijn και, αφετέρου, στο να αναδειχθούν τεχνικές ανάπτυξης συναρτήσεων οι οποίες μπορούν να αποτελούν δομικό στοιχείο ακολουθιών De bruijn.

1.7 Δομή Μεταπτυχιακής Διατριβής

Η δομή της μεταπτυχιακή διατριβής διαμορφώνεται όπως φαίνεται παρακάτω με τα εξής κεφάλαια:

1ο Κεφάλαιο: Εισαγωγή

Στο κεφάλαιο αυτό γίνεται αναφορά των δομικών στοιχείων της κρυπτογραφίας καθώς επίσης διατυπώνεται ο σκοπός της έρευνας καθώς επίσης και τα βασικά ερευνητικά ερωτήματα.

2ο Κεφάλαιο: Γεννήτριες κρυπτογραφικών ακολουθιών – Κρυπτογραφικά κριτήρια συναρτήσεων

Θα γίνει μελέτη των γνωστότερων γεννητριών ψευδοτυχαίων ακολουθιών καθώς επίσης θα δούμε τα κριτήρια τυχαιότητας που χαρακτηρίζουν τις παραγόμενες ακολουθίες. Επίσης θα διερευνήσουμε την δομή των γραμμικών και μη καταχωρητών ολίσθησης και τον ρόλο που διαδραματίζουν οι ψευδοτυχαίες ακολουθίες. Στο τέλος θα γίνει αναφορά και στις γνωστότερες επιθέσεις κρυπτανάλυσης που διέπουν τους κρυπταλγόριθμους που μελετήσαμε.

3ο Κεφάλαιο: Ακολουθίες de Bruijn

Σε αυτό το κεφάλαιο θα μελετήσουμε τις ακολουθίες De Bruijn, οι οποίες χρησιμοποιούνται σε πλήθος εφαρμογών όπως στους κρυπταλγόριθμους ροής, στην παραγωγή ψευδοτυχαίων ακολουθιών και σε πολλούς ακόμα τομείς. Θα διερευνήσουμε τρόπους με τους οποίους παράγουμε de Bruijn ακολουθίες καθώς επίσης και πώς κατασκευάζουμε νέες de Bruijn ακολουθίες από ήδη υπάρχουσες.

4ο Κεφάλαιο: Κρυπτογραφικές ιδιότητες γεννητριών de Bruijn

Θα γίνει μελέτη των κρυπτογραφικών ιδιοτήτων των γεννητριών που παράγουν ακολουθίες de Bruijn καθώς επίσης θα μελετηθούν τρόποι κατασκευής των ακολουθιών αυτών με διάφορες μεθοδολογίες. Η μελέτη θα γίνει με διεξαγωγή πειραμάτων πάνω στις αρχικές και νέες ακολουθίες.

5ο Κεφάλαιο: Επίλογος

Σε αυτό το κεφάλαιο θα διατυπωθούν τα συμπεράσματα της διατριβής.

Κεφάλαιο 2
Γεννήτριες κρυπτογραφικών ακολουθιών – Κρυπτογραφικά κριτήρια

2.1 Εισαγωγή

Το κυνήγι των τυχαίων αριθμών θεωρείται μια από τις μεγαλύτερες προκλήσεις των θετικών επιστημών. Η επιλογή τυχαίων αριθμών είναι ένα κρίσιμο χαρακτηριστικό στην ασφάλεια των κρυπτοσυστημάτων. Στο κεφάλαιο αυτό θα αναλύσουμε τον όρο τυχαία ακολουθία καθώς επίσης θα διερευνήσουμε ποια είναι τα κριτήρια τυχαιότητας. Η ασφάλεια ενός αλγορίθμου ροής έγκειται στην τυχαιότητα της παραγόμενης κλειδοροής και στο κατά πόσο μπορεί αυτή να προβλεφθεί. Η κλειδοροή που παράγεται από τον αλγόριθμο ροής εξαρτάται από την αρχική κατάσταση της γεννήτριας, η οποία με τη σειρά της εξαρτάται από το μυστικό κλειδί. Βασικός σχεδιαστικός στόχος είναι η γεννήτρια κλειδοροής να παράγει ακολουθία η οποία δεν μπορεί να προβλεφθεί, και που εκπληρώνει στο μέγιστο δυνατό κριτήρια τυχαιότητας όπως περιοδικότητα, πολυπλοκότητα υλοποίησης, βαθμός γραμμικής πολυπλοκότητας καθώς επίσης διαθέτει καλά στατιστικά χαρακτηριστικά.

Στο κεφάλαιο αυτό θα δούμε ποιες είναι οι πιο γνωστές κατηγορίες γεννητριών ψευδοτυχαίων ακολουθιών, καθώς επίσης και μερικά από τα βασικά κριτήρια τυχαιότητας που πρέπει να πληροί μια ακολουθία έτσι ώστε να θεωρείται κρυπτογραφικά ισχυρή. Επίσης θα διερευνήσουμε τη δομή των μη γραμμικών καταχωρητών ολίσθησης με ανάδραση καθώς επίσης θα ασχοληθούμε με τις λογικές συναρτήσεις και τις ιδιότητες τους και το ρόλο που διαδραματίζουν στους μη γραμμικούς καταχωρητές με ανάδραση. Στο τέλος θα δούμε τις γνωστότερες κρυπτογραφικές επιθέσεις που υφίστανται οι κρυπτογραφικοί αλγόριθμοι που μελετήσαμε.

2.2 Γεννήτριες ψευδοτυχαίων ακολουθιών

Πολλά κρυπτογραφικά συστήματα περιέχουν σαν βασικό τους στοιχείο την χρήση τυχαίων αριθμών ως κλειδιά (keys), μοναδικά αναγνωριστικά (unique identifier), προκλήσεις (challenges) κτλπ. Όμως η παραγωγή τέτοιων αριθμών σε έναν υπολογιστή είναι μια αρκετά δύσκολη και πολύπλοκη διαδικασία ώστε να υλοποιηθεί. Μια πιθανή μέθοδος για την παραγωγή τέτοιων αριθμών είναι με την χρήση γεννητριών ψευδοτυχαίων ακολουθιών. Είναι αναγκαίο να καταλήξουμε στους ορισμούς που ακολουθούν:

Μια ακολουθία αριθμών είναι ψευδοτυχαία όταν περνά επιτυχώς όλους τους γνωστούς στατιστικούς ελέγχους περί τυχαιότητας καθώς επίσης η ακολουθία είναι απρόβλεπτη, δηλαδή δοθέντος ενός τμήματος της ακολουθίας αυτής είναι υπολογιστικά αδύνατο για τον κρυπταναλυτή να καθορίσει τον αμέσως επόμενο αριθμό.

Πραγματικά τυχαία ακολουθία είναι η ακολουθία που χαρακτηρίζεται από πλήρη έλλειψη μοτίβων και είναι τέτοια ώστε να μην μπορεί να γίνει καμία πρόβλεψη για τα επόμενα στοιχεία της καθώς επίσης δεν μπορεί να αναπαραχθεί με αξιοπιστία.

Η **γεννήτρια ψευδοτυχαίων ακολουθιών** (ΓΨΑ - pseudorandom number generator - PRNG) είναι ένα υπολογιστικό σύστημα (αλγόριθμος) η οποία έχει σχεδιαστεί έτσι ώστε να παράγει ακολουθίες αριθμών μήκους L που να προσεγγίζουν τις πραγματικά τυχαίες ακολουθίες αφού πρώτα έχουν αρχικοποιηθεί με μια τυχαία τιμή (μήκους K) που ονομάζεται σπόρος (seed) όπου $L>>K$.

Είναι σημαντικό να σημειώσουμε πως οι γεννήτριες ψευδοτυχαίων ακολουθιών (ΓΨΑ) βασίζονται σε ντετερμινιστικές διαδικασίες, ουσιαστικά σε αναδρομικές σχέσεις, οι οποίες

καθορίζουν τις συσχετίσεις μεταξύ των παραγόμενων αριθμών. Η ακολουθία των αριθμών που παράγεται από ΓΨΑ δεν είναι πραγματικά τυχαία, αλλά είναι προκαθορισμένη από ένα σύνολο αρχικών τιμών εισόδου της γεννήτριας που ονομάζεται κατάσταση της ΓΨΑ. Αυτό έχει σαν αποτέλεσμα να μην μπορούν να υπάρξουν απόλυτα τυχαίες ακολουθίες αλλά μόνο ψευδοτυχαίες. Βέβαια αυτή η ιδιότητα των ΓΨΑ τις κάνει ιδανικό συστατικό για τους κρυπτογραφικούς αλγόριθμους ροής.

Αν μια ακολουθία λαμβάνει τιμές στο σύνολο {0,1} τότε καλείται δυαδική ακολουθία όπου και είναι η συνηθέστερη για κρυπτογραφικές εφαρμογές. Στην παρούσα διατριβή θα αναφερόμαστε σε δυαδικές ακολουθίες.

Στις μέρες μας γεννήτριες ψευδοτυχαίων αριθμών εφαρμόζονται στα περισσότερα λειτουργικά συστήματα: για παράδειγμα στο Linux υπάρχει στο /dev/random καθώς επίσης σε πολλές βιβλιοθήκες από διάφορες γλώσσες προγραμματισμού. Σε πολλές εξ αυτών υπεισέρχονται και καταχωρητές ολίσθησης με ανάδραση (FSR), οι οποίοι περιγράφονται στις επόμενες παραγράφους.

2.3 Τυχαιότητα μιας ακολουθίας

Για να μπορέσουμε να χρησιμοποιήσουμε μια ακολουθία ως κλειδοροή σε έναν αλγόριθμο ροής θα πρέπει, όπως προαναφέρθηκε να είναι «τυχαία». Όμως τι εννοούμε με την λέξη «τυχαία»;

Η τυχαιότητα μιας ψευδοτυχαίας ακολουθίας αναφέρεται στην μη προβλεψιμότητα της ακολουθίας. Η μη προβλεψιμότητα ορίζεται ως εξής: δοθέντος ενός τμήματος της ακολουθίας είναι υπολογιστικά αδύνατο για τον κρυπταναλυτή να καθορίσει το αμέσως επόμενο bit της ακολουθίας με πιθανότητα μεγαλύτερη του 50%.

Θα μπορούσαμε να πούμε ότι είναι προφανές ότι μια ψευδοτυχαία ακολουθία πρέπει να έχει μεγάλη περίοδο αλλά αυτό δεν είναι αρκετό για να χαρακτηριστεί τυχαία. Τα πρώτα κριτήρια τυχαιότητας μιας δυαδικής ακολουθίας προτάθηκαν από τον **Golomb** και είναι τα ακόλουθα:

1. Ισοκατανεμημένο πλήθος 0 και 1. Δηλαδή για μια περιοδική ακολουθία το πλήθος των ψηφίων 0 θα πρέπει να είναι κατά προσέγγιση ίσο με το πλήθος των ψηφίων 1. (Balance property)

2. Κριτήριο «διαδρομών» των 0 και 1. Ως διαδρομή ορίζεται το τμήμα της ακολουθίας που αποτελείται από συνεχόμενα 0 ή 1 καθώς και πριν και μετά από αυτά βρίσκονται διαφορετικά bit από τα ήδη υπάρχοντα. Το εν λόγω κριτήριο ικανοποιείται αν σε μια περίοδο της ακολουθίας οι μισές διαδρομές έχουν μήκος 1, το 1/4 των διαδρομών να έχουν μήκος 2, το 1/8 μήκος 3 κ.ο.κ. Η συνθήκη ισχύει όσο ο αριθμός των διαδρομών είναι μεγαλύτερος ή ίσος από το 2^k όπου k το μήκος της διαδρομής. (Run property)

3. Κριτήριο αυτοσυσχέτισης δύο τιμών. Η αυτοσυσχέτιση ουσιαστικά είναι η διαφορά της ακολουθίας με την ολισθημένη εκδοχή της (κυκλική ολίσθηση). Συγκεκριμένα, η συνάρτηση αυτοσυσχέτισης για τιμή τ είναι η ακόλουθη: $C(\tau) = \sum_{i=0}^{N-1}(-1)^{\alpha_i \oplus a_{i+\tau}}$

   για ακολουθία $\alpha_0\alpha_1\alpha_2\alpha_3$.... Σύμφωνα με το εν λόγω κριτήριο, η συνάρτηση αυτοσυσχέτισης πρέπει να λαμβάνει δύο τιμές (two level autocorrelation property): την τιμή N για τ=0 και μία σταθερή τιμή σε όλες τις άλλες περιπτώσεις.

Τα κριτήρια που προτάθηκαν από τον Golomb δεν είναι τα μοναδικά κριτήρια για να αξιολογηθεί μια ακολουθία ως προς την τυχαιότητα και κατ' επέκταση να είναι κρυπτογραφικά ασφαλής. Όπως θα δούμε και στις επόμενες παραγράφους, τα κριτήρια του Golomb δεν αρκούν για το χαρακτηρισμό μιας ακολουθίας ως ψευδοτυχαίας.

Υπάρχει επίσης ένα μεγάλο πλήθος από ελέγχους που έχουν προταθεί, όπως από τον Knuth [4] ο οποίος έχει προτείνει ένα σύνολο από απλούς ελέγχους τυχαιότητας. Αντίστοιχοι έλεγχοι έχουν προταθεί και από τον Marsaglia [21], ενώ και ο οργανισμός NIST (National Institute of Standards and Technlology) παρέχει την δικιά του σουίτα ελέγχων τυχαιότητας (SP800-22) για γεννήτριες τυχαίων και ψευδοτυχαίων αριθμών [31]. Βέβαια ο Yongge Wang έδειξε ότι η σουίτα του NIST δεν είναι αρκετή για να διαπιστώσει ασθενείς γεννήτριες ψευδοτυχαίων αριθμών [37].

Πέραν των κριτηρίων που έχουν προταθεί από το Golomb, θα πρέπει η περίοδος της κλειδοροής να είναι όσο το δυνατόν μεγαλύτερη. Λόγω του περιορισμού της απαίτησης να αναπαράγεται αξιόπιστα η κλειδοροή, οι μηχανισμοί παραγωγής της είναι τέτοιοι ώστε η ακολουθία να έχει πεπερασμένο μέγεθος (δηλαδή περίοδο) και σε κάποια χρονική στιγμή επαναλαμβάνεται. Το πλεονέκτημα του κρυπταναλυτή σχετίζεται αντίστροφα με το μέγεθος της περιόδου. Τα bits της ακολουθίας της κλειδοροής θα πρέπει να περνούν με επιτυχία όλους τους γνωστούς ελέγχους περί τυχαιότητας. Έτσι η ακολουθία της κλειδοροής θα μοιάζει με μια τυχαία ακολουθία bits.

Στην περίπτωση αυτή ο συνδυασμός της κλειδοροής με το απλό κείμενο θα έχει σαν αποτέλεσμα τη δημιουργία μιας ψευδοτυχαίας ακολουθίας, της οποίας η κρυπτανάλυση θα είναι δύσκολη.

2.4 Γραμμικοί καταχωρητές ολίσθησης με ανάδραση (LFSR)

Οι γραμμικοί καταχωρητές ολίσθησης με ανάδραση (LFSR) είναι τα βασικά και δημοφιλή συστατικά πολλών γεννητριών κλειδοροής σε κρυπταλγόριθμους ροής, γιατί είναι εύκολη η υλοποίησή τους σε hardware. Επίσης παράγουν εγγυημένα ακολουθίες με καλά στατιστικά χαρακτηριστικά που ικανοποιούν τα 3 κριτήρια του Golomb.

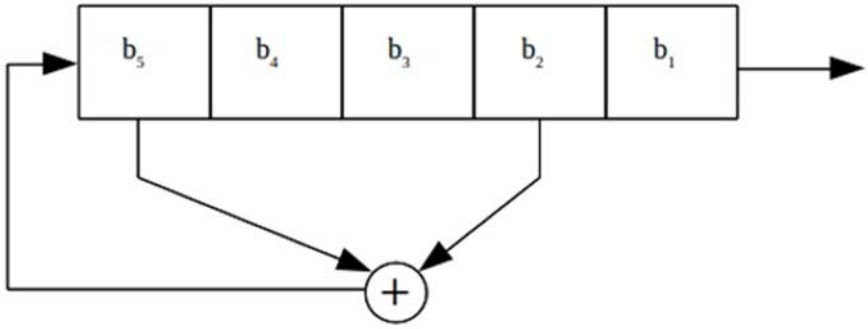
Οι LFSR – όπως και οι NLFSR που θα δούμε στη συνέχεια - ανήκουν στην κατηγορία των μηχανών πεπερασμένης κατάστασης (finate state machines). Μια μηχανή πεπερασμένης κατάστασης ορίζεται ως η συσκευή η οποία αποτελείται από έναν πεπερασμένο αριθμό καταστάσεων όπου η μεταπήδηση από τη μια κατάσταση στην άλλη ορίζεται από την υπάρχουσα κατάσταση και την είσοδο. Η είσοδος ορίζεται ως μια ακολουθία από ένα πεπερασμένο σύνολο στοιχείων. Η έξοδος μια μηχανής πεπερασμένης κατάστασης είναι μια ακολουθία από ένα πεπερασμένο σύνολο στοιχείων.

2.4.1 Λειτουργία γραμμικού καταχωρητή ολίσθησης με ανάδραση (LFSR)

Ο γραμμικός καταχωρητής με ανάδραση είναι ψηφιακό κύκλωμα που αποτελείται από N βαθμίδες (θέσεις μνήμης) όπου το περιεχόμενο κάθε μιας βαθμίδας είναι είτε 0 είτε 1.

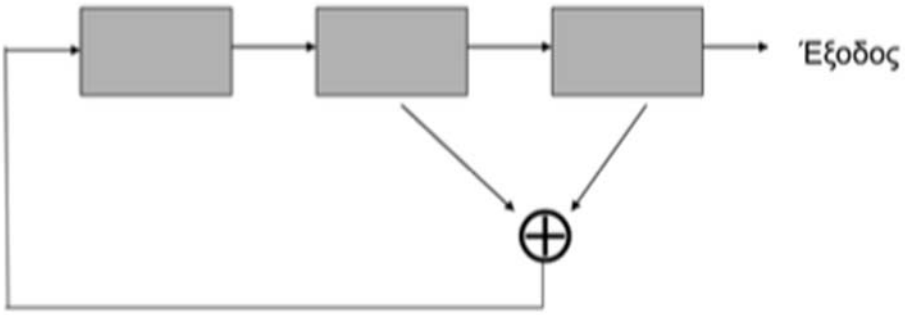
Υπάρχει μια πύλη XOR (πρόσθεση $\oplus$) η οποία έχει ως εισόδους κάποιες από τις βαθμίδες του LFSR. Ουσιαστικά η πύλη XOR υλοποιεί τη συνάρτηση ανάδρασης: επειδή η συνάρτηση αυτή είναι γραμμική (απουσιάζουν γινόμενα), αναφερόμαστε σε γραμμικούς καταχωρητές. Το σύνολο των τιμών των βαθμίδων του LFSR ονομάζεται κατάσταση (state) του LFSR για την τρέχουσα χρονική στιγμή. Κάθε χρονική στιγμή η κατάσταση του LFSR μεταβάλλεται. Η μεταβολή της κατάστασης ενεργοποιείται με τον παλμό του ρολογιού του συστήματος. Σε κάθε νέα κατάσταση, η τιμής της πρώτης βαθμίδας του LFSR προέρχεται από τις τιμές της προηγούμενης κατάστασης βάσει της πρόσθεσης XOR που υλοποιεί ο LFSR, όπως μπορούμε να δούμε και στην εικόνα 2.1. Οι τιμές των υπόλοιπων βαθμίδων προκύπτουν από ολίσθηση προς τα δεξιά των τιμών όλων των βαθμίδων της προηγούμενης κατάστασης. Η έξοδος παράγεται από την

τελευταία (δεξιότερη) βαθμίδα. Για παράδειγμα, στην Εικόνα 2.1, η νέα κάθε φορά τιμή της b_5 καθορίζεται από την πρόσθεση XOR $b_5 \oplus b_2$



Εικόνα 2.1: Αναπαράσταση γραμμικού καταχωρητή ολίσθησης με ανάδραση.

Για λόγους καλύτερης κατανόησης θα περιγράψουμε έναν απλό LFSR μεγέθους 3 με τυχαία αρχική κατάσταση την 101.



Εικόνα 2.2: Αναπαράσταση γραμμικού καταχωρητή ολίσθησης με ανάδραση με 3 θέσεις μνήμης.

Έτσι, εφόσον η αρχική κατάσταση του LFSR είναι η 101, τότε η έξοδος του θα είναι το 1 (το περιεχόμενο της δεξιότερης θέσης μνήμης).

Η επόμενη κατάσταση μετά τον παλμό του ρολογιού θα είναι η 110. Το πρώτο 1 (από τα αριστερά) προέκυψε από την πρόσθεση XOR (⊕) των τιμών της δεύτερης και τρίτης βαθμίδας της προηγούμενης κατάστασης. Τα υπόλοιπα 2 bit «10» προκύπτουν από την ολίσθηση δεξιότερα της πρώτη και δεύτερης βαθμίδας της προηγούμενης κατάστασης. Η νέα έξοδος του LFSR θα είναι 0.

Παρακάτω ακολουθεί ο πίνακα 2.1 που περιέχει όλες τις διαδοχικές καταστάσεις του LFSR.

α/α	Κατάσταση	Έξοδος
1	101	1
2	110	0
3	111	1
4	011	1
5	001	1
6	100	0
7	010	0
8	101	1

Πίνακας 2.1: Πίνακας εξόδου του LFSR με 3 θέσεις μνήμης και αρχική κατάσταση 101

Βλέπουμε ότι στην κατάσταση 8 συναντούμε την ίδια κατάσταση με την αρχική 101 και έτσι διαπιστώνουμε ότι υπάρχει περιοδική επανάληψη των καταστάσεων.

Άρα η παραγόμενη ακολουθία είναι η: 1011100 1011100 … (δηλαδή, αποτελεί περιοδική επανάληψη της 1011100)

2.4.2 Ιδιότητες LFSR

Όπως είδαμε και στο κεφάλαιο 2.4.1 από την λειτουργία του LFSR, η παραγόμενη ακολουθία (εξόδου) του LFSR εξαρτάται τόσο από την ανάδρασή του, δηλαδή από το ποιες βαθμίδες έχουν επιλεχθεί ώστε να προστίθενται XOR μεταξύ τους, όσο και από την αρχική του κατάσταση.

‘Ενας LFSR τάξης N μπορεί να διέλθει το πολύ από 2^N-1 διαφορετικές καταστάσεις , όλες δηλαδή τις πιθανές N-άδες πλην της μηδενικής και κατ’ επέκταση θα έχει μέγιστη δυνατή περίοδο 2^N-1 . Οι LFSR που πετυχαίνουν την μέγιστη δυνατή περίοδο ονομάζονται **πρωταρχικοί** (**primitive**).

Εάν ο LFSR είναι πρωταρχικός εξαρτάται αποκλειστικά από την ανάδρασή του και όχι από την αρχική του κατάσταση. Επίσης μόνο οι πρωταρχικοί LFSR έχουν κρυπτογραφική αξία γιατί διασφαλίζουν την μέγιστη δυνατή περίοδο της παραγόμενης ακολουθίας.

Οι ακολουθίες που παράγονται από πρωταρχικούς καταχωρητές ονομάζονται **ακολουθίες μεγίστου μήκους** (**m – sequences** ή **maximal length sequences**).

Τα πλεονεκτήματα λοιπόν των πρωταρχικών LFSR είναι τα ακόλουθα:

- Παράγουν ακολουθίες με την μέγιστη δυνατή περίοδο
- Ικανοποιούν πάντα και τα τρία κριτήρια τυχαιότητας του Golomb

Γενικότερα θα μπορούσαμε να συμπεράνουμε ότι ένας LFSR είναι κατάλληλος ως γεννήτρια κλειδοροής λόγω της εύκολης υλοποίησής του, την παραγωγή ακολουθιών με καλά κρυπτογραφικά χαρακτηριστικά τυχαιότητας καθώς επίσης και ακολουθίες με μεγάλη περίοδο. Όμως η απευθείας χρήση των LFSR, χωρίς κάποια τροποποίηση, λόγω της χαμηλής γραμμικής πολυπλοκότητας των παραγόμενων ακολουθιών - η οποία θα οριστεί στη συνέχεια - τους καθιστά κρυπτογραφικά ακατάλληλους. Περαιτέρω, σε έναν οποιονδήποτε αλγόριθμο ροής, αν ξέρουμε ένα τμήμα του αρχικού μηνύματος, τότε ουσιαστικά ξέρουμε απευθείας το αντίστοιχο τμήμα της κλειδοροής. Έτσι εάν η κλειδοροή παράγεται από έναν LFSR, αυτομάτως θα γνωρίζουμε το αντίστοιχο τμήμα της τρέχουσας κατάστασής του. Συνεπώς, θα πρέπει η γεννήτρια κλειδοροής να έχει σύνθετη δομή, που να μην επιτρέπει την εύκολη εύρεση της αρχικής της κατάστασης με επιθέσεις τύπου γνωστού μηνύματος (known – plaintext).

2.5 Γραμμική πολυπλοκότητα και αλγόριθμος Berlekamp-Massey

Μια οποιαδήποτε ακολουθία $\alpha_0\alpha_1\alpha_2\ldots$ είτε πεπερασμένη είτε περιοδική μπορεί να παραχθεί από πολλούς διαφορετικούς LFSR. Για μια υπάρχουσα ακολουθία το μέγεθος του μικρότερου LFSR που την παράγει ονομάζεται **γραμμική πολυπλοκότητα** (**linear complexity/linear span**) της ακολουθίας. Προφανώς, για κάθε ακολουθία μεγίστου μήκους (m-sequence) με περίοδο 2^N-1 η γραμμική πολυπλοκότητα είναι N. Αυτό συμβαίνει γιατί δεν μπορεί να υπάρξει μικρότερος LFSR

(πχ βαθμού L , όπου L<N) που να παράγει την ίδια ακολουθία περιόδου 2^N-1 γιατί η μέγιστη ακολουθία που μπορεί να παράγει είναι η 2^L-1.

Υπάρχει αλγόριθμος που δοθείσης μιας ακολουθίας υπολογίζει την γραμμική πολυπλοκότητα αλλά και τον μικρότερο LFSR που την παράγει. Αυτός ο αλγόριθμος ονομάζεται αλγόριθμος Berlekamp-Massey και δημιουργήθηκε από τον Massey [20] το 1969 ο οποίος έδειξε ότι ο επαναληπτικός αλγόριθμος που προτάθηκε από τον Berlekamp για την αποκωδικοποίηση BCH κωδίκων, μπορούσε να χρησιμοποιηθεί για να βρεθεί ο μικρότερος LFSR δοθείσης μιας ακολουθίας.

Για δοθείσα ακολουθία περιόδου L και γραμμικής πολυπλοκότητας N με N < L/2 έχει αποδειχτεί ότι ο μικρότερος LFSR περιόδου L – ο οποίος υπολογίζεται από τον αλγόριθμο Berlekamp-Massey - που παράγει την ακολουθία είναι μοναδικός. Για να μπορέσει να υπολογίσει ο αλγόριθμος Berlekamp-Massey αυτόν τον μοναδικό LFSR θα πρέπει να γνωρίζουμε οποιαδήποτε 2N διαδοχικά bits της ακολουθίας. Έτσι εάν για μια ακολουθία η γραμμική της πολυπλοκότητα N είναι μικρότερη από την μισό (1/2) της περιόδου της ακολουθίας και αν γνωρίζουμε μόνο 2N διαδοχικά bits της ακολουθίας ουσιαστικά μπορούμε να βρούμε την γεννήτρια όλης της ακολουθίας με την χρήση του προαναφερθέντος αλγορίθμου και, άρα, να μαντέψουμε ολόκληρη την υπόλοιπη ακολουθία.

Για αυτό τον λόγο μια ακολουθία για να χρησιμοποιηθεί ως κλειδοροή σε κρυπτογραφικό αλγόριθμο ροής θα πρέπει να έχει όσο γίνεται υψηλή γραμμική πολυπλοκότητα έτσι ώστε ο επιτιθέμενος να χρειάζεται να γνωρίζει έναν πολύ μεγάλο αριθμό από τα bit της ακολουθίας για να μπορέσει να βρει ολόκληρη την ακολουθία.

Έτσι, με βάση τα ανωτέρω αλλά και από την παράγραφο 2.3, βλέπουμε ότι υπάρχει πλήθος κριτηρίων που θα πρέπει να ικανοποιούνται από μία ακολουθία προκειμένου να μην είναι προβλέψιμη και άρα κατ' επέκταση κρυπτογραφικά ισχυρή. Συνεπώς, για την παραγωγή ακολουθιών μεγάλης γραμμικής πολυπλοκότητας χρησιμοποιούνται κατάλληλα μη γραμμικές λογικές συναρτήσεις (οι οποίες ορίζονται στη συνέχεια).

2.6 Λογικές συναρτήσεις (Boolean Functions)

Οι λογικές συναρτήσεις παίζουν κεντρικό ρόλο στη σχεδίαση πολλών συμμετρικών κρυπτοσυστημάτων καθώς και στην ασφάλεια τους. Τι είναι μια λογική συνάρτηση όμως ;

Μια λογική συνάρτηση, n μεταβλητών f(x) όπου x = (x_1,….x_n), είναι μια συνάρτηση από το σύνολο (set) F_2^n όλων των δυαδικών διανυσμάτων μήκους n στο σώμα F_2 = {0,1}. Ο αριθμός των n μεταβλητών πρακτικά σπάνια είναι μεγάλος. Στους κρυπταλγόριθμους ροής συνήθως ο αριθμός n μεταβλητών ήταν λιγότερο από 10 και πλέον συχνά είναι λιγότερο από 20 και αυτό λόγω των αλγεβρικών επιθέσεων που δημιουργήθηκαν σχετικά πρόσφατα.

Κάθε λογική συνάρτηση n μεταβλητών περιγράφεται από τον **πίνακα αληθείας** (**Truth Table**) της. Παρακάτω ακολουθεί ένα παράδειγμα πίνακα αληθείας συνάρτησης f(x) τριών μεταβλητών:

X_3	**X_2**	**X_1**	**Έξοδος**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Πίνακας 2.2: Πίνακας αληθείας τριών μεταβλητών

Από τον παραπάνω πίνακα παίρνουμε τα εξής αποτελέσματα :

f(000) = 1, f(001)= 1, f(010) = 1, f(011) = 0, f(100) = 1, f(101) = 0, f(110) = 0 , f(111) = 1

Μια λογική συνάρτηση μπορεί να αναπαρασταθεί με διάφορες μαθηματικές εκφράσεις και η πιο συνηθισμένη για τις κρυπτογραφικές εφαρμογές είναι η **Αλγεβρική Κανονική Μορφή**

(**Algebraic Normal Form**). Η αλγεβρική κανονική μορφή είναι η αναπαράσταση της συνάρτησης ως πολυώνυμο με εκθέτες μικρότερους ή ίσους με 1 ως XOR άθροισμα των γινομένων των μεταβλητών.

Παρακάτω ακολουθεί παράδειγμα συνάρτησης τριών μεταβλητών με αλγεβρική κανονική μορφή : f(x1, x2, x3) = x1*x2⊕ x2*x3⊕ x3. Στην συνέχεια ακολουθεί ο πίνακας αληθείας της:

x3	**x2**	**x1**	**Έξοδος= x1*x2⊕ x2*x3⊕ x3**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Πίνακας 2.3: Πίνακας αληθείας f(x1, x2, x3) = x1*x2⊕ x2*x3⊕ x3

Παραπάνω είδαμε ότι ξέροντας την αλγεβρική κανονική μορφή μπορούμε να κατασκευάσουμε πολύ εύκολα τον πίνακα αληθείας. Ισχύει ωστόσο και το αντίστροφο, δηλαδή από τον πίνακα αληθείας μιας συνάρτησης f μπορούμε να κατασκευάσουμε την αλγεβρική κανονική μορφή της.

2.6.1 Ιδιότητες και χαρακτηριστικά των λογικών συναρτήσεων

Γραμμική συνάρτηση (**linear**) είναι η συνάρτηση που στην αλγεβρική κανονική μορφή της υπάρχει μόνο πρόσθεση XOR μεταξύ κάποιων μεταβλητών της, χωρίς να εμφανίζεται κανένα γινόμενο μεταβλητών. Για παράδειγμα η συνάρτηση f(x1, x2, x3) = x1⊕ x2⊕ x3 είναι γραμμική. Αν στην έκφραση μιας συνάρτησης υπάρχει έστω και ένας πολλαπλασιασμός τότε είναι **μη**

γραμμική (**non-linear**). Για παράδειγμα η συνάρτηση f(x1, x2, x3) = x1⊕ x2*x3 είναι μη γραμμική.

Επειδή σε έναν LFSR η λογική συνάρτηση που τον περιγράφει είναι πάντα γραμμική, αυτός είναι και ο λόγος που η παραγόμενη κρυπτογραφική ακολουθία έχει χαμηλή γραμμική πολυπλοκότητα. Έτσι είναι προφανές γιατί στους κρυπταλγόριθμους ροής επιθυμούμε μη γραμμικές συναρτήσεις. Ο αλγεβρικός βαθμός (algebraic degree) ή απλά βαθμός μίας συνάρτησης ορίζεται ως το πλήθος των μεταβλητών που εμφανίζεται στο μεγαλύτερο γινόμενο της αλγεβρικής κανονικής μορφής της. Έτσι, οι γραμμικές συναρτήσεις έχουν πάντα βαθμό 1. Για τη μη γραμμική συνάρτηση που είδαμε ανωτέρω, ο βαθμός της είναι 2.

Βάρος (**weight**) μιας συνάρτησης f (συμβολίζεται ως wt(f)) ορίζεται το πλήθος των «1» στην έξοδο του πίνακα αληθείας. Εάν σε μια συνάρτηση f που περιέχει n μεταβλητές το πλήθος των «0» και «1» είναι ισομοιρασμένο στην έξοδό της τότε λέγεται **ισοβαρής** (**balanced**). Δηλαδή σε μία ισοβαρή συνάρτηση n μεταβλητών ισχύει $wt(f) = 2^{n-1}$.

Απόσταση Hamming (**Hamming distance**) δύο λογικών συναρτήσεων f, g ορίζεται η ποσότητα wt(f⊕g), η οποία πρακτικά ισούται με τον αριθμό των θέσεων στους πίνακες αληθείας των δύο συναρτήσεων οι οποίες διαφέρουν μεταξύ τους.

Για οποιαδήποτε συνάρτηση f, η **συμπληρωματική** (**complementary**) αυτής ορίζεται ως η f⊕1. Όπως είναι προφανές η συμπληρωματική συνάρτηση f⊕1 έχει τον ίδιο βαθμό με την f.

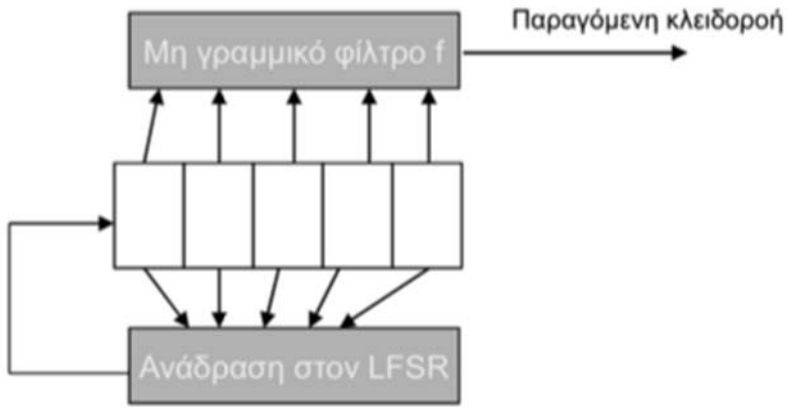
Η **μη γραμμικότητα** (**non linearity**) μιας συνάρτησης f ορίζεται ως nl(f) το μικρότερο πλήθος θέσεων στον πίνακα αληθείας της, οι οποίες εάν μεταβληθούν θα μετατρέψουν τη συνάρτηση σε γραμμική (linear). Θα δούμε στις επόμενες παραγράφους πόσο σημαντικός είναι ο ρόλος της μη γραμμικότητας στις κρυπτογραφικές συναρτήσεις.

2.7 Μη γραμμικές λογικές συναρτήσεις

Για την παραγωγή κλειδοροής, χρησιμοποιούνται μη γραμμικές συναρτήσεις, συχνά εφαρμοζόμενες κατάλληλα σε πρωταρχικούς LFSRs. Τα διάφορα είδη γεννητριών κλειδοροής περιγράφονται στη συνέχεια.

2.7.1 Μη γραμμικά φίλτρα (nonlinear filter generators)

Σε αυτή την περίπτωση του μη γραμμικού φίλτρου, μια μη γραμμική συνάρτηση εφαρμόζεται στις βαθμίδες ενός LFSR και η παραγόμενη κλειδοροή είναι πια η έξοδος αυτής της συνάρτησης. Η τεχνική αυτή οδηγεί σε ακολουθίες μεγάλης γραμμικής πολυπλοκότητας.



Εικόνα 2.3: Λειτουργία LFSR με μη γραμμικό φίλτρο

Από την παραπάνω εικόνα μπορούμε να δούμε την λειτουργία των γραμμικών φίλτρων σε LFSR. Ο LFSR λειτουργεί κανονικά όπως έχει περιγραφεί και στην παραπάνω παράγραφο με την διαφορά τώρα ότι η κλειδοροή δεν λαμβάνεται από την έξοδο της δεξιότερης βαθμίδας του LFSR αλλά από την έξοδο της συνάρτησης f. Η συνάρτηση f ονομάζεται μη γραμμικό φίλτρο (non linear filter).

Εάν θέλουμε κρυπτογραφικά ισχυρή παραγόμενη ακολουθία θα πρέπει το μη γραμμικό φίλτρο να είναι ισοβαρής συνάρτηση ώστε να εξασφαλίζεται ίσος αριθμός bits «0» και «1» στην κλειδοροή. Επίσης θα πρέπει να ο LFSR να είναι **πρωταρχικός** εφόσον επιθυμούμε τη **μέγιστη** περίοδο στην παραγόμενη ακολουθία.

Αν ο βαθμός συνάρτησης f είναι deg(f) = d και το μέγεθος του LFSR είναι N, τότε η μέγιστη δυνατή τιμή της γραμμικής πολυπλοκότητας της παραγόμενης κλειδοροής είναι:

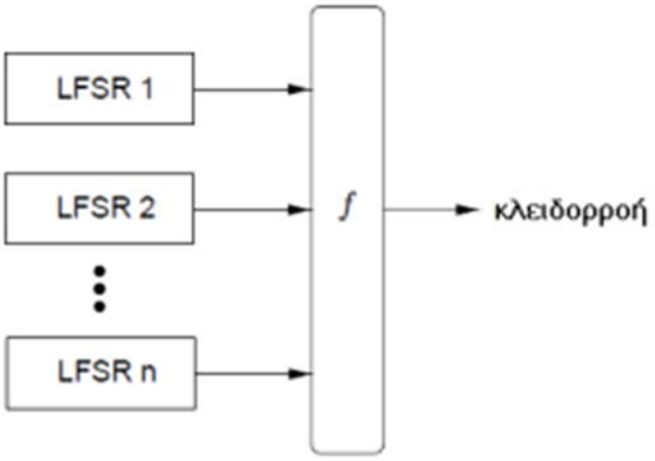
$$\sum_{i=1}^{d} \binom{N}{i}$$

Από την παραπάνω σχέση καταλαβαίνουμε ότι για να έχουμε υψηλή τιμή γραμμικής πολυπλοκότητας θα πρέπει ο βαθμός d της f να είναι υψηλός και το μέγεθος του LFSR αρκετά μεγάλο.

Για μια συνάρτηση f δεν μπορεί να προσδιοριστεί η ακριβής τιμή της γραμμική πολυπλοκότητας της κλειδοροής αλλά υπάρχουν συγκεκριμένες κατασκευές συναρτήσεων που αν χρησιμοποιηθούν ως μη γραμμικά φίλτρα παράγουν εγγυημένα πολύ υψηλής γραμμικής πολυπλοκότητας κλειδοροή [10,17].

2.7.2 Μη γραμμικοί συνδυαστές (nonlinear combination generators)

Σε αυτήν την περίπτωση, οι έξοδοι πολλών LFSRs τροφοδοτούν μια μη γραμμική λογική συνάρτηση και η παραγόμενη κλειδοροή προκύπτει από την έξοδο αυτής της συνάρτησης. Και πάλι σκοπός μας είναι να πετύχουμε υψηλή γραμμική πολυπλοκότητα.

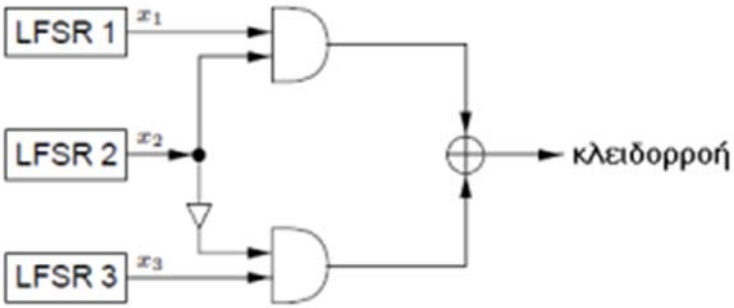


Εικόνα 2.4: Γεννήτρια μη γραμμικού συνδιαστή

Με κατάλληλη επιλογή της f (ιδανικά ίση με το ελάχιστο κοινό πολλαπλάσιο των περιόδων των ακολουθιών που παράγουν οι LFSRs) και των LFSRs, μπορεί να διασφαλιστεί μεγάλη περίοδος της κλειδοροής. Ένα κλασσικό παράδειγμα μη γραμμικού συνδιαστή είναι η γεννήτρια Geffe.

Η γεννήτρια Geffe (Εικόνα 2.5) αποτελείται από 3 πρωταρχικούς LFSR μεγίστου μήκους L_1, L_2, L_3. όπου για τους αριθμούς ισχύει ότι είναι ανά δύο πρώτοι μεταξύ τους, οι οποίοι «τροφοδοτούν» μια γραμμική συνδυαστική συνάρτηση f:

$f(x_1, x_1, x_3) = x_1x_2 \oplus (x_2+1)x_3 = x_1x_2 \oplus x_2x_3 \oplus x_3$



Εικόνα 2.5: Γεννήτρια μη Geffe

Η παραγόμενη κλειδοροή έχει περίοδο $(2^{L_1} -1)(2^{L_2} -1)(2^{L_3} -1)$, ενώ αποδεικνύεται ότι η γραμμική της πολυπλοκότητα ισούται με $L = L_1 L_2 + L_2 L_3 + L_3$

Η γεννήτρια Geffe είναι ωστόσο κρυπτογραφικά ασθενής επειδή διαρρέει πληροφορία για τις καταστάσεις των LFSR 1 και LFSR 2 μέσα στην ακολουθία εξόδου. Γι' αυτό, παρότι έχει υψηλή περίοδο και σχετικά υψηλή γραμμική πολυπλοκότητα, είναι ευάλωτη σε επιθέσεις συσχέτισης τις οποίες θα δούμε στην επόμενη παράγραφο.

2.8 Κρυπτογραφικές επιθέσεις και κρυπτογραφικά κριτήρια

Παρόλο που με τα ανωτέρω μοντέλα μπορούν να παραχθούν κρυπτογραφικές ακολουθίες με καλά χαρακτηριστικά τυχαιότητας, εφόσον χρησιμοποιούνται κατάλληλες ισοβαρείς μη γραμμικές συναρτήσεις μεγάλου βαθμού, εν τούτοις οι ιδιότητες των λογικών συναρτήσεων που υπεισέρχονται μπορούν να δημιουργήσουν κάποια ευπάθεια στον κρυπτογραφικό αλγόριθμο. Χαρακτηριστικές τέτοιες περιπτώσεις περιγράφονται παρακάτω:

Επιθέσεις συσχέτισης (correlation attack)

Η επίθεση συσχέτισης σε γεννήτριες μη γραμμικού συνδυασμού αναπτύχθηκε αρχικά από τον Siegenthaler [25] όπου και αναλύει μια μέθοδο κρυπτανάλυσης που βασίζεται σε ιδιότητες της συνάρτησης ετεροσυσχέτισης (cross correlation function) ανάμεσα στην ακολουθία μεγίστου μήκους (m-sequence) που παράγεται από τον LFSR και την ακολουθία εξόδου. Ουσιαστικά έδειξε ότι εάν η έξοδος κάπου LFSR ταυτίζεται με πιθανότητα $p > ½$ και εάν ένα τμήμα της

κλειδοροής είναι γνωστό τότε μπορεί να βρεθεί η αρχική κατάσταση του LFSR. Προκειμένου λοιπόν τα κρυπτοσυστήματα να είναι ανθεκτικά σε μια τέτοιου είδος επίθεση θα πρέπει να πληρούνται οι εξής ιδιότητες:

Η κρυπτογραφική συνάρτηση πρέπει να είναι ισοβαρής (balanced) για να αποφεύγεται στατιστική εξάρτηση μεταξύ της εισόδου και της εξόδου της ακολουθίας. Ειδικότερα, οποιοσδήποτε μη γραμμικός συνδιαστής f(x) οποίος χρησιμοποιήθηκε για να παράγει την ψευδοτυχαία ακολουθία στον κρυπταλγόριθμο ροής θα πρέπει να μείνει ισοβαρής εάν κρατήσουμε σταθερές κάποιες μεταβλητές εισόδου x_i έτσι ώστε η συνάρτηση f(x) να είναι **ανθεκτική σε συσχετίσεις** (**correlation immune**)[3] . Για παράδειγμα εάν θεωρήσουμε μια μεταβλητή , έστω την x1 ότι έχει σταθερή τιμή (είτε «0» είτε «1») , ανατρέχουμε στις γραμμές του πίνακα αληθείας που αντιστοιχούν σε αυτήν την τιμή και οι οποίες θα είναι σε πλήθος οι μισές, κοιτάζουμε εάν η έξοδος των γραμμών αυτών έχει ισομοιρασμένο αριθμό από «0» και «1». Εάν ναι και αυτό ισχύει για οποιαδήποτε μεταβλητή τότε η ανθεκτικότητα στις συσχετίσεις είναι $1^{ης}$ τάξης. Αν η παραπάνω ιδιότητα ισχύει για κάθε ζεύγος μεταβλητών στις οποίες «σταθεροποιούμε» την τιμή (00 ή 01 ή 10 ή 11) τότε έχουμε ανθεκτικότητα στις συσχετίσεις $2^{ης}$ τάξης κ.ο.κ.

Επιθέσεις προσεγγίσεων (**approximation attack**)

Η επίθεση προσέγγισης εφαρμόζεται όταν η συνάρτηση f της κλειδοροής μπορεί να προσεγγιστεί ικανοποιητικά από μια συνάρτηση χαμηλότερου βαθμού. Για να το εξηγήσουμε, ας αναλογιστούμε μια συνάρτηση f που παράγει κλειδοροή k, και η οποία διαφέρει από μια γραμμική συνάρτηση g σε λίγες θέσεις στον πίνακα αληθείας τους. Εάν αντικαταστήσουμε την f με την γραμμική συνάρτηση g που παράγει μία άλλη κλειδοροή k' τότε χρησιμοποιώντας τον αλγόριθμο Berlekamp-Massey θα βρούμε τον LFSR που την παράγει. Όμως έτσι θα μπορέσουμε να έχουμε βρει και το μεγαλύτερο τμήμα της αρχικής κλειδοροής k μιας και η k με την k' διαφέρουν σε πολύ λίγα bit, λόγω της «εγγύτητας» της g με την f..

Για να μπορέσουμε να αποτρέψουμε επιθέσεις προσέγγισης θα πρέπει η κρυπτογραφική συνάρτηση να είναι **υψηλής μη γραμμικότητας** (non linearity), δηλαδή η ελάχιστη απόσταση Hamming μεταξύ της λογικής συνάρτησης που παράγει την κλειδοροή και των υπόλοιπων γραμμικών συναρτήσεων θα πρέπει να είναι μεγάλη.

Αλγεβρικές επιθέσεις (algebraic attacks)

Οι αλγεβρικές επιθέσεις βασίζονται στην χρήση τεχνικών έτσι ώστε να απλοποιηθούν οι σύνθετες μαθηματικές εκφράσεις που περιγράφουν τα συμμετρικά κρυπτογραφικά συστήματα και να είναι εφικτή η ανάκτηση του μυστικού κλειδιού. Προτάθηκαν από τον Courtois [06] το 2003 και από τότε μελετήθηκαν από πολλούς ερευνητές.

Θεωρούμε γεννήτρια μη γραμμικού φίλτρου f και $k_{0...}k_n$ το κλειδί που είναι η αρχική κατάσταση του LFSR. Η μετάβαση στην επόμενη κατάσταση μπορεί να περιγραφεί μαθηματικά ως μια γραμμική έκφραση L αλλά οι τελικές καταστάσεις και άρα οι μαθηματικές εκφράσεις είναι μη γραμμικές και άρα πολύ σύνθετες λόγω της μη γραμμικής συνάρτησης f.

Για να βρεθεί το μυστικό κλειδί καλείται ο επιτιθέμενος να λύσει ένα σύνθετο σύστημα εξισώσεων και, μάλιστα, όσο πιο μεγάλος ο βαθμός της f τόσο πιο σύνθετες εξισώσεις. Σκοπός της επίθεσης είναι να μειωθεί ο βαθμός των μη γραμμικών εξισώσεων. Αυτό γίνεται όταν υπάρχει τέτοια συνάρτηση φίλτρου g που να ικανοποιεί την εξής ιδιότητα

$$f*g = 0 \text{ ή } (f \oplus 1)*g = 0$$

Κάθε τέτοια συνάρτηση g ονομάζεται **εκμηδενιστής** (**annihilator**) της g.

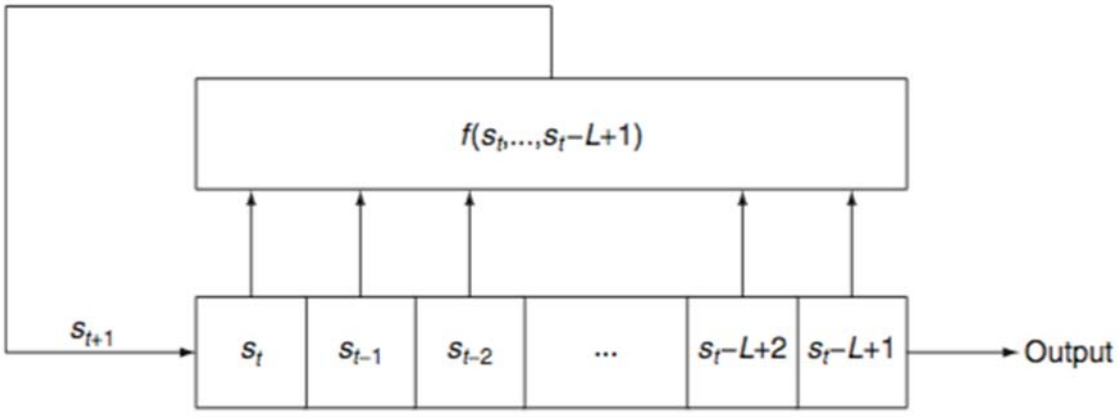
Κατά συνέπεια για να είναι μια συνάρτηση ανθεκτική στις αλγεβρικές επιθέσεις θα πρέπει να μην υπάρχει εκμηδενιστής χαμηλού βαθμού. Έτσι οδηγηθήκαμε στην δημιουργία του κρυπτογραφικού κριτηρίου της **αλγεβρικής ανθεκτικότητας** (**algebraic immunity**) και ορίζεται ως ο ελάχιστος βαθμός από όλους τους μη μηδενικούς εκμηδενιστές της f ή της $f \oplus 1$ (συμπληρωματική). Η αλγεβρική ανθεκτικότητα για κάθε λογική συνάρτηση n μεταβλητών είναι μεγαλύτερη ή ίση από $\lceil n/2 \rceil$.

Κατά συνέπεια, η αντίσταση διαφόρων κρυπτοσυστημάτων που περιέχουν λογικές συναρτήσεις έναντι διαφόρων γνωστών επιθέσεων μπορούν να ποσοτικοποιηθούν μέσω βασικών ιδιοτήτων των λογικών αυτών συναρτήσεων. Από τις παραπάνω κρυπτογραφικές επιθέσεις που είδαμε, μπορούμε να συμπεράνουμε ότι η σχεδίαση των κρυπτογραφικών συναρτήσεων χρειάζεται να λάβει υπ' όψιν πολλά διαφορετικά χαρακτηριστικά ταυτόχρονα όπως είναι η υψηλή μη γραμμικότητα, η ανθεκτικότητα σε συσχετίσεις καθώς και η αλγεβρική ανθεκτικότητα.

2.9 Μη γραμμικοί καταχωρητές ολίσθησης με ανάδραση

Οι καταχωρητές ολίσθησης με ανάδραση (FSR) είναι βασικά συστατικά στοιχεία πολλών γεννητριών κλειδοροής οι οποίες χρησιμοποιούνται στους κρυπταλγόριθμους ροής. Όπως ήδη είδαμε για τη γραμμική περίπτωση (LFSR), κάθε χρονική στιγμή του ρολογιού του συστήματος, η εσωτερική κατάσταση του καταχωρητή ολισθαίνει προς τα δεξιά, έχοντας σαν έξοδο το δεξιότερο bit και το νέο bit από τα αριστερά υπολογίζεται από την προηγούμενη κατάσταση και τη συνάρτηση ανάδρασης f. Οι γραμμικοί καταχωρητές ανάδρασης με ολίσθηση (LFSR) χρησιμοποιούν μια γραμμική συνάρτηση f, οι οποίοι έχουν μελετηθεί επαρκώς και χρησιμοποιηθεί εκτεταμένα. Οι μη γραμμικοί καταχωρητές ανάδρασης με ολίσθηση (NLFSR) χρησιμοποιούν μια μη γραμμική συνάρτηση f.

Οι NLFSR ανήκουν και αυτοί στην κατηγορία των μηχανών πεπερασμένης κατάστασης (finite state machines) όπου μπορούμε να δούμε στην εικόνα 3.1 την λειτουργία τους.



Εικόνα 2.6: Ένας μη γραμμικός καταχωρητής ολίσθησης με ανάδραση

Το bit της ανάδρασης υπολογίζεται από $S_{t+1} = f(S_t,\ldots,S_{t-L+1})$, όπου η f μπορεί να είναι οποιαδήποτε μη γραμμική συνάρτηση με L αριθμό μεταβλητών. Κάθε φορά που ο παλμός του εσωτερικού ρολογιού μεταβάλλεται , τα bit ολισθαίνουν κατά μια θέση προς τα δεξιά , με έξοδο το bit που περιέχεται στην θέση S_t-L+1

Στην συνέχεια θα δούμε παράδειγμα για καλύτερη κατανόηση της διαδικασίας που ακολουθείται.

Θεωρούμε έναν απλό NLFSR με αριθμό μεταβλητών L = 3 και συνάρτηση ανάδρασης

$$f(s_t, s_{t-1}, s_{t-2}) = s_t s_{t-1} \oplus s_t \oplus s_{t-2} .$$

Επίσης ακολουθεί ο πίνακας 3.2 όπου μπορούμε να δούμε τον πίνακα αληθείας της συνάρτησης f. Θεωρούμε τα τρία αρχικά bit της ακολουθίας , τα οποία αποτελούν και την αρχική κατάσταση του NLFSR τα παρακάτω : $s_0 = 0$, $s_1 = 1$, $s_2 = 0$. Έτσι η παραγόμενη ακολουθία είναι οι εξής 01001101…

Ακολουθεί ο πίνακας αληθείας της $f(s_t, s_{t-1}, s_{t-2}) = s_t s_{t-1} \oplus s_t \oplus s_{t-2}$

s_t	**s_{t-1}**	**s_{t-2}**	**$f(s_t, s_{t-1}, s_{t-2})$**
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Πίνακας 2.4: Πίνακας αληθείας NLFSR 3 βαθμίδων

Η αλγεβρική κανονική μορφή της συνάρτησης ανάδρασης είναι $f(x1, x2, x3)=x1x2 \oplus x1 \oplus x3$.

Είναι γνωστό ότι οι παραγόμενες δυαδικές ακολουθίες από καταχωρητές ανάδρασης με ολίσθηση μήκους L, είτε είναι γραμμικοί είτε μη γραμμικοί, έχουν περίοδο το πολύ ίση με $2^L -1$ (αν θεωρήσουμε ότι δεν διέρχονται ποτέ από τη μηδενική κατάσταση). Ωστόσο, μπορούμε να θεωρήσουμε και την περίπτωση της μηδενικής κατάστασης, εφόσον ο καταχωρητής δεν θα μείνει για πάντα σε αυτή – το οποίο σημαίνει ότι, πρακτικά, θα πρέπει να ισχύει $f(0,0,…,0)=1$..

Αυτές λοιπόν οι ακολουθίες που πετυχαίνουν την μέγιστη περίοδο 2^L ονομάζονται de Bruijn και θα τις μελετήσουμε στην συνέχεια.

Γενικότερα οι NLFSR μπορούν να χρησιμοποιηθούν σε οποιαδήποτε μοντέλο γεννήτριας (μη γραμμικού φίλτρου/συνδιαστή ή και σε άλλο μοντέλο). Δείχνουν να παρέχουν ασφάλεια έναντι και των αλγεβρικών επιθέσεων. Οι NLFSR μήκους L παράγουν ακολουθίες που η γραμμική πολυπλοκότητα έχει μέγιστο άνω όριο 2^L έναντι των LFSR ίδιου μήκους που παράγουν ακολουθίες με μέγιστη γραμμική πολυπλοκότητα L. Πολλά ερωτήματα υπάρχουν ακόμα όπως το πώς κατασκευάζονται μεγάλοι NLFSR με εγγυημένη μέγιστη περίοδο ή αν είναι ευάλωτοι σε άλλες επιθέσεις. Γενικότερα δεν υπάρχουν ασφαλείς μέθοδοι για την αποτίμηση της ασφάλειας σε κατασκευές που εμπεριέχουν NLFSR.

2.10 Εφαρμογή NLFSRs σε κρυπταλγόριθμους ροής: Αλγόριθμος Trivium

Οι NLFSR χρησιμοποιούνται σαν δομικά στοιχεία σε πολλούς σύγχρονους αλγόριθμους ροής όπως ο Grain , Trivium , Mickey , Keeloq , VEST κ.α.

Το 2005 οι Christophe De Canniere και Bart Preneel ανέπτυξαν έναν κρυπταλγόριθμο ροής που ονομάστηκε Trivium. Συμπεριλαμβάνεται στο κρυπταλγορίθμους του προγράμματος eSTREAM [29] και έχει σχεδιαστεί για περιβάλλοντα με περιορισμένους πόρους και απαίτησης υψηλών επιδόσεων. Εκτός από την αποδοτικότητα στην ταχύτητα, σημαντικό ρόλο έπαιξε και η απλή αρχιτεκτονική στην κατασκευή του. Ο αλγόριθμος αποτελείται από φάσεις. Η πρώτη φάση σχετίζεται με την αρχικοποίηση του αλγορίθμου όπου και προσδιορίζεται η μυστική κατάστασή του με βάση ένα μυστικό κλειδί και ένα διάνυσμα αρχικοποίησης (initialization vector – IV). Στη δεύτερη φάση η διαδικασία της παραγωγή της κλειδοροής εκτελείται χρησιμοποιώντας όλα τα δεδομένα της πρώτης φάσης. Όλη η διαδικασία παράγει κλειδοροές μεγίστου μήκους 2^{64} bits.

Πρώτη φάση (φάση ρύθμισης): Σε αυτήν την φάση η μυστική κατάσταση (S) του κρυπταλγόριθμου ορίζεται και αποθηκεύεται σε τρεις NLFSR διαφορετικών μεγεθών (93, 84, 111 bit). Ο NLFSR μεγέθους 93 bits κρατάει το μυστικό κλειδί (SK) του αλγόριθμου. Τα υπόλοιπα

13 bits του καταχωρητή αρχικοποιούνται με μηδενικά και προστίθενται με τα bits από το μυστικό κλειδί:

$(S_1,S_2,...,S_{93}) \leftarrow (SK_1,SK_2,...,SK_{80},0,...,0)$

Στην συνέχεια ένα δεύτερο μέρος της μυστικής κατάστασης , το διάνυσμα αρχικοποίησης (IV), είναι αυτό που χρησιμοποιείται. Συγκεκριμένα, το IV αποθηκεύεται στον NLFSR μεγέθους 84 bit και αντιπροσωπεύει τις θέσεις της μυστικής κατάστασης από τις S_{94} έως την S_{173}. Τα υπόλοιπα 4 bits τίθενται ίσα με μηδέν.

$(S_{94}, S_{95}, ..., S_{177}) \leftarrow (IV_1, IV_2, ..., IV_{80}, 0, ..., 0)$

Το υπόλοιπο μέρος της μυστικής κατάστασης χρησιμοποιείται από το τρίτο NLFSR μεγέθους 111 bit. Αυτός ο NLFSR αρχικοποιείται με όλα τα bit 0, εκτός από τα τρία τελευταία που παίρνουν τιμή 1.

$(S_{178}, S_{179}, ..., S_{286}, S_{287}, S_{288}) \leftarrow (0, 0, ..., 1, 1, 1)$

Στην συνέχεια στη μυστική κατάσταση εκτελούνται διάφορες διαδικασίες όπως προσθέσεις AND, XOR και αντιμεταθέσεις bits. Αυτές οι διαδικασίες επαναλαμβάνονται για 1152 φορές με σκοπό να παραχθεί η μυστική κατάσταση η οποία χρησιμοποιείται στην συνέχεια για να παραχθεί η κλειδοροή. Στην συνέχεια μπορούμε να δούμε τις διαδικασίες που αναφέραμε σε μορφή ψευδοκώδικα όπου το σύμβολο $\oplus$ αναφέρεται στην πράξη XOR και το σύμβολο $\wedge$ στην πράξη AND. Τα αποτελέσματα αποθηκεύονται σε 3 συγκεκριμένες μεταβλητές T_1 T_2 T_3 :

1: for i=1 to 1152 do

2: T1 = S66 $\oplus$S91 $\wedge$S92 $\oplus$S93 $\oplus$S171

3: T2 = S162 $\oplus$S175 $\wedge$S176 $\oplus$S177 $\oplus$S264

4: T3 = S243 $\oplus$S286 $\wedge$S287 $\oplus$S288 $\oplus$S69

5:

6: (S1,S2,...,S93) = (T3,S1,...,S92)

7: (S94,S95,...,S177) = (T1,S94,...,S176)

8: (S178,S179,...,S288) = (T2,S178,...,S287)

9: end for

Δεύτερη φάση (φάση παραγωγής κλειδοροής): Η φάση παραγωγής της κλειδοροής είναι παρόμοια με την φάση που περιγράψαμε παραπάνω. Βασιζόμενη στην τρέχουσα

κατάσταση (S), η κλειδοροή μήκους N ($N \leq 2^{64}$) παράγεται σύμφωνα με εντολές που επαναλαμβάνονται N φορές. Το πρώτο σετ εντολών αποτελείται από τρεις προσθέσεις XOR που εκτελούνται σε τρία ζευγάρια bit η καθεμιά. Το κάθε αποτέλεσμα αποθηκεύεται σε μια μεταβλητή T_1 ,T_2 ,T_3 . Στην συνέχεια ένα bit της κλειδοροής (που συμβολίζεται από το Z_i) παράγεται από κάθε επανάληψη i ως αποτέλεσμα γραμμικού συνδιασμού των T_1 ,T_2 ,T_3 . Στην συνέχεια ακολουθεί ο ψευδοκώδικας των εντολέων που ακολουθήθηκαν.

1: for i=1 to N do

2: T1=S66⊕S93

3: T2=S162⊕S177

4: T3=S243⊕S288

5:

6: Zi=T1⊕T2⊕T3

7: T1 = T1 ⊕S91 ∧S92 ⊕S171

8: T2 = T2 ⊕S175 ∧S176 ⊕S264

9: T3 = T3 ⊕S286 ∧S287 ⊕S69

10: (S1,S2,...,S93) = (T3,S1,...,S92)

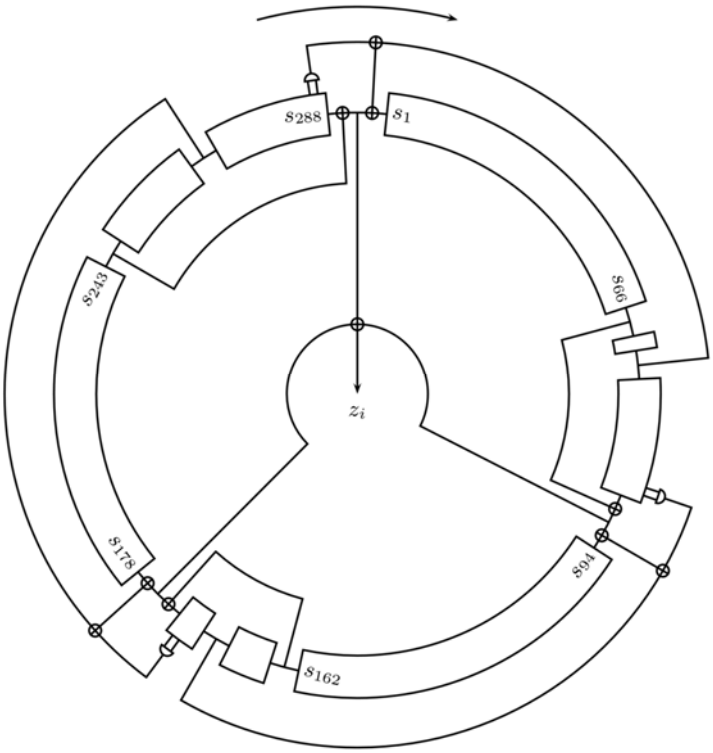
11: (S94,S95,...,S177) = (T1,S94,...,S176)

12: (S178,S179,...,S288) = (T2,S178,...,S287)

13: end for

Στην συνέχεια μετά την παραγωγή της κλειδοροής Z, μπορεί να χρησιμοποιηθεί για διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης.

Στην συνέχεια ακολουθεί η εικόνα με την δομή του Trivium. [33]



Εικόνα 2.7: Δομή του κρυπταλγόριθμου Trivium

Η ασφάλεια του κρυπταλγόριθμου Trivium ακόμα μελετάται από την ερευνητική κοινότητα. Παρά την απλή δομή του υπάρχει μεγάλη δυσκολία στο να αποκαλυφθούν πληροφορίες για την εσωτερική δομή του, όπως για παράδειγμα της μυστικής κατάστασής του. Σύμφωνα με όλες τις τελευταίες έρευνες καμιά κρυπτογραφική επίθεση πέραν της επίθεσης εξαντλητικής αναζήτησης (brute force) δεν είχε καλύτερα αποτελέσματα. Βέβαια έχουν αναπτυχθεί επιθέσεις που τα αποτελέσματά τους είναι αρκετά κοντά με αυτά της εξαντλητικής αναζήτησης όπως για παράδειγμα η επίθεση κύβου (cube attack) [08] και άλλες που δεν θα μελετηθούν στην παρούσα διατριβή.

Κεφάλαιο 3
Ακολουθίες De Bruijn

3.1 Εισαγωγή

Τα τελευταία χρόνια έχουν αρχίσει και χρησιμοποιούνται, αντί για τους LFSRs, μη γραμμικοί καταχωρητές ολίσθησης με ανάδραση (NLFSRs) στις γεννήτριες κλειδοροής. Αυτό γίνεται για περαιτέρω ενίσχυση της ασφάλειας: για παράδειγμα, στις αλγεβρικές επιθέσεις, η επίλυση των μαθηματικών εξισώσεων που θα κληθεί να επιλύσει ο κρυπταναλυτής είναι ακόμα πιο δύσκολη όταν έχουμε NLFSR αντί για LFSR. Προφανώς, και στις περιπτώσεις αυτές, επιθυμούμε να έχουμε NLFSR που να παράγει ακολουθίες μεγίστης περιόδου. Ωστόσο, για τους NLFSR δεν έχουμε συγκεκριμένη μεθοδολογία κατασκευής που να εγγυάται τη μέγιστη περίοδο, όπως ισχύει για τους LFSR (για τους οποίους γνωρίζουμε πώς να επιλέξουμε τη συνάρτηση ανάδρασης έτσι ώστε να παράγεται ακολουθία μεγίστου μήκους). Καταλαβαίνουμε ότι θέλουμε μέγιστη περίοδο στην παραγόμενη ακολουθία έτσι ώστε να πετύχουμε όσο το δυνατόν καλύτερα κρυπτογραφικά χαρακτηριστικά. Τέτοια χαρακτηριστικά επιτυγχάνουν οι λεγόμενες ακολουθίες De Bruijn, οι οποίες έχουν μέγιστη περίοδο, ισοκατανεμημένο αριθμό από bits «0» και «1» και καλά

χαρακτηριστικά τυχαιότητας. Οι ακολουθίες αυτές χρησιμοποιούνται σε πλήθος εφαρμογών όπως στους κρυπταλγόριθμους ροής, στην παραγωγή ψευδοτυχαίων ακολουθιών, στην αναγνώριση τρισδιάστατων μοτίβων, στην μοντελοποίηση δικτύων και πολλά άλλα καθώς επίσης έχουν υπάρξει πολλές μαθηματικές κατασκευές τους[26,18]. Σε αυτό το κεφάλαιο θα δούμε τρόπους με τους οποίους παράγουμε de Bruijn ακολουθίες, καθώς επίσης και πώς κατασκευάζουμε νέες de Bruijn ακολουθίες από ήδη υπάρχουσες.

3.2 Ορισμός των ακολουθιών de Bruijn

Μια de Bruijn ακολουθία που αποτελείται από ένα σύνολο τιμών k-όρων και τάξης n είναι μια ακολουθία με περίοδο k^n που περιέχει κάθε n-αδα του συνόλου τιμών ακριβώς μια φόρα σε όλη την περίοδο της ακολουθίας.

Οι ακολουθίες de Bruijn πήραν το όνομα τους από τον Ολλανδό μαθηματικό Nicholas de Bruijn. Το 1946 ανακάλυψε τον τύπο ο οποίος δίνει όλες τις πιθανές de Bruijn ακολουθίες που υπάρχουν σε σύνολο τιμών k-όρων και τάξης n και είναι ο ακόλουθος :

$$((k-1)!)^{k^{n-1}} k^{k^{n-1}-n} \quad [02]$$

Στην κρυπτογραφία μας ενδιαφέρουν οι δυαδικές ακολουθίες de Bruijn και με αυτές θα ασχοληθούμε στην παρούσα διατριβή.

Ο αριθμός των δυαδικών ακολουθιών de Bruijn περιόδου 2^n με βάση τον παραπάνω τύπο θα είναι $2^{2^{n-1}-n}$. Ένα παράδειγμα δυαδικής ακολουθίας de Bruijn με περίοδο $2^4 = 16$ είναι το παρακάτω:

$\{S_t\}$ = 0000111101100101 όπου όλες οι δυνατές 4-άδες εμφανίζονται μια φορά στην περίοδο της ακολουθίας. Ακολουθούν όλες οι παραγόμενες 4-άδες. Οι τρεις τελευταίες 4-άδες δημιουργούνται με «περιτύλιξη» της ακολουθίας.

0000, 0001, 0011, 0111, 1111, 1110, 1101, 1011, 0110, 1100, 1001, 0010, 0101, 1010, 0100, 1000

Γενικά οι δυαδικές ακολουθίες de Bruijn έχουν περίοδο 2^n (όπου n η τάξη), ισοκατανεμημένο αριθμό «0» και «1» σε μια περίοδο, ικανοποιούν αρκετά κριτήρια τυχαιότητας εάν και

παράγονται με χρήση ντετερμινιστικών μεθόδων. Έχουν χρησιμοποιηθεί ως πηγή για παραγωγή ψευδοτυχαίων αριθμών καθώς επίσης και στην παραγωγή κλειδοροών σε κρυπταλγόριθμους ροής καθώς επίσης και σε πολλές άλλες εφαρμογές των μαθηματικών.

3.3 Παραγωγή δυαδικών ακολουθιών de Bruijn με χρήση NLFSR

Οι ακολουθίες de Bruijn μπορούν να παραχθούν από μη γραμμικές συναρτήσεις ανάδρασης n-μεταβλητών. Με αρχική κατάσταση (s_0 , s_1 ,… s_{n-1}) και μια μη γραμμική λογική συνάρτηση f(z_0, z_1, …, z_{n-1}) μπορεί να παραχθεί η ακολουθία $s_{t+n} = f(s_t , s_{t+1} , \dots , s_{t+n-1})$ για t = 0, 1, 2… Για την κατασκευή τους μπορεί να χρησιμοποιηθεί NLFSR n-βαθμίδων.

Για να κατανοήσουμε καλύτερα τη δομή των δυαδικών ακολουθιών de Bruijn θα ακολουθήσει παράδειγμα. Για να δούμε πόσες δυαδικές ακολουθιών de Bruijn τάξης 3 υπάρχουν θα χρησιμοποιήσουμε τον τύπο $2^{2^{n-1}-n}$ και όπου n=3 . Έτσι γίνεται $2^{2^2-3} \Rightarrow 2$. ‘Αρα βλέπουμε ότι οι δυαδικές ακολουθίες de Bruijn τάξης 3 είναι συνολικά 2.

Θεωρούμε NLFSR μήκους 3 με την ακόλουθη μη γραμμική συνάρτηση $\mathbf{f_1(x_0, x_1, x_2) = x_0 \oplus x_1 x_2 \oplus x_1 \oplus 1}$. Στην συνέχεια παραθέτουμε τον πίνακα που περιέχει τις καταστάσεις των 3 καταχωρητών του NLSFR σε κάθε παλμό του ρολογιού , με αρχική κατάσταση [0,0,0]

Στάδιο 0	**Στάδιο 1**	**Στάδιο 2**	**T - Παλμός**
0	0	0	0
0	0	1	1
0	1	1	2
1	1	1	3
1	1	0	4

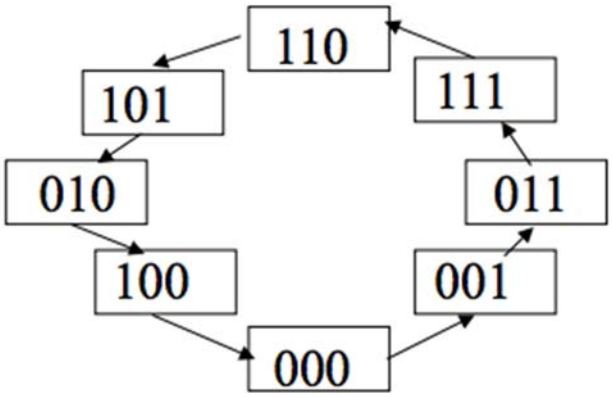
1	0	1	5
0	1	0	6
1	0	0	7

Πίνακας 3.1: Πίνακας σταδίων NLFSR μη γραμμικής f_1

Η παραγόμενη ακολουθία είναι η «00011101». Άρα ο NLFSR μήκους 3 με την μη γραμμική συνάρτηση f_1 που έχει αλγεβρική κανονική μορφή $f_1(x_0, x_1, x_2) = x_0 \oplus x_1 x_2 \oplus x_1 + \oplus 1$ παράγει ακολουθία de Bruijn και αυτό το επιβεβαιώνουμε γιατί διέρχεται από όλες τις καταστάσεις (εικόνα 3.2) με μέγιστη περίοδο $8 = 2^3$.

Στη συνέχεια μπορούμε να δούμε το διάγραμμα καταστάσεων του NLFSR (εικόνα 3.2). Το διάγραμμα αυτό μας δείχνει όλες τις καταστάσεις (3-άδες) που πέρασε ο NLFSR. Μάλιστα αυτό μπορούμε να το δούμε και από την ακολουθία de Bruijn που δημιουργήθηκε «00011101». Έτσι εάν πάρουμε σειριακά όλες τις 3-άδες που υπάρχουν στην προαναφερθείσα ακολουθία θα δούμε ότι είναι αυτές.

Σημείωση: για να μπορέσουμε να πάρουμε όλες τις 3-αδες από την ακολουθία , θα πρέπει να την θεωρήσουμε σαν έναν κυκλικό βρόχο την ακολουθία, δηλαδή για να μπορέσουμε για παράδειγμα να πάρουμε την 3-άδα 010 θα πρέπει να πάρουμε τα 2 δεξιότερα bit της ακολουθίας «00011101» καθώς και το πρώτο αριστερότερο έτσι ώστε να δημιουργηθεί ο κυκλικός βρόχος και να πάρουμε την 3-άδα που αναφέραμε. 'Ετσι παρατηρούμε ότι για να δημιουργηθούν όλες οι πιθανές n-άδες (υπακολουθίες) θα πρέπει να ενώνουμε τα n-1 αρχικά ψηφία με τα τελικά.



Εικόνα 3.1: Διάγραμμα καταστάσεων του NLFSR

x_2	x_1	x_0	f_1
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

Πίνακας 3.2: Πίνακας αληθείας συνάρτησης f_1

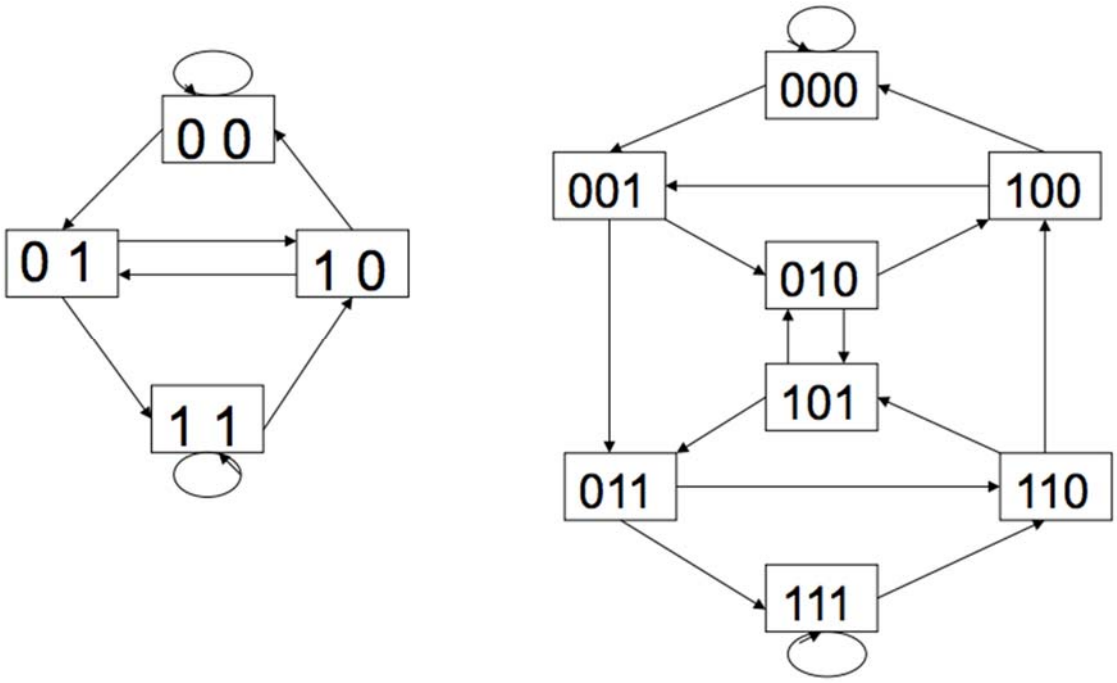
Παρατηρώντας τον πίνακα αληθείας της συνάρτησης f_1, καθώς επίσης τον πίνακα καταστάσεων της ακολουθίας αλλά και την ίδια την παραγόμενη de Bruijn ακολουθία, μπορούμε να

αντλήσουμε πληροφορίες για το πώς, γνωρίζοντας την ακολουθία de Bruijn, μπορούμε να κατασκευάσουμε τον πίνακα αληθείας της. Αυτό γίνεται εάν από μια ακολουθία de Bruijn τάξης n , επιλέξουμε τις αντίστοιχες n-άδες καθώς επίσης και το αμέσως δεξιότερο ψηφίο της κάθε n-άδας. Έτσι θα έχουμε αντιστοιχίσει κάθε μία από τις n-άδες με το δεξιότερο ψηφίο της. Στη συνέχεια θα πρέπει να αντιστρέψουμε το κάθε στοιχείο της n-άδας από τα δεξιά προς τα αριστερά και έτσι θα έχουμε ουσιαστικά κάθε σειρά του πίνακα αληθείας με την αντίστοιχη έξοδό του.

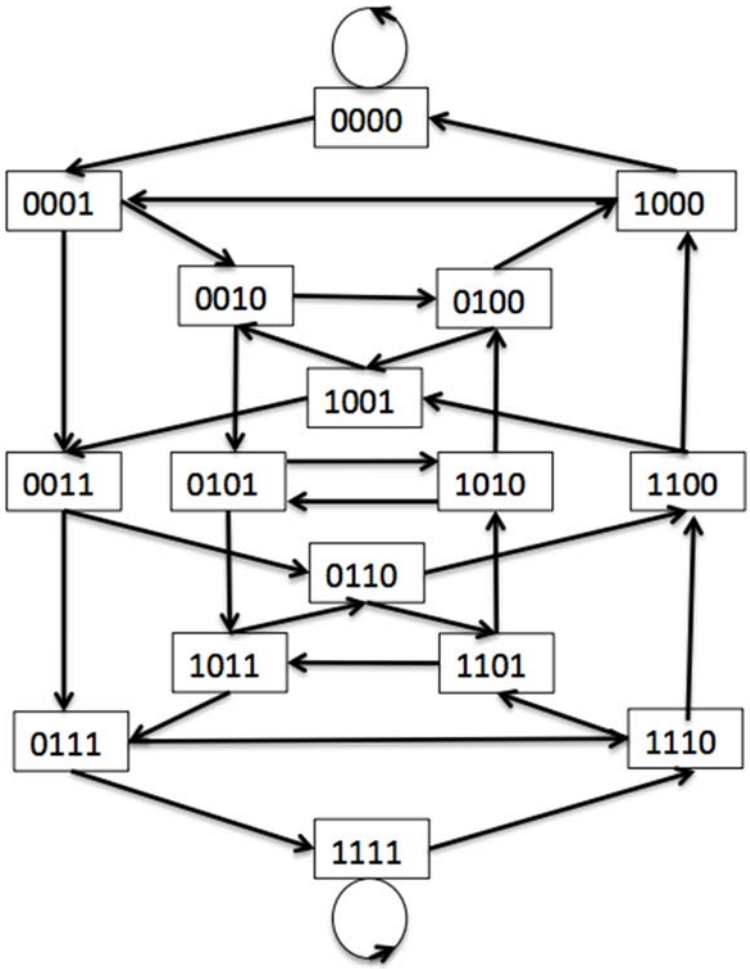
Για να το κατανοήσουμε καλύτερα θα χρησιμοποιήσουμε για παράδειγμα την προηγούμενη δυαδική ακολουθία de Bruijn «00011101». Έτσι για παράδειγμα η πρώτη 3-άδα (γιατί είναι τάξης 3) θα είναι η 000 με αντιστοίχιση το αμέσως δεξιότερο ψηφίο της που είναι το 1 (ουσιαστικά, η έξοδος του πίνακα αληθείας), η δεύτερη 3άδα θα είναι η 001 με το επόμενο ψηφίο 1, η τρίτη 3άδα θα είναι η 011 με επόμενο ψηφίο το 1 , η τέταρτη 3άδα θα είναι η 111 με το επόμενο ψηφίο το 0 κλπ.. Στη συνέχεια θα αντιστρέψουμε τα ψηφία της κάθε τριάδας από τα αριστερά προς τα δεξιά των ζευγαριών και έτσι θα έχουμε πλέον την πρώτη 3άδα 000 με το επόμενο ψηφίο 1, τη δεύτερη 3άδα θα είναι 100 με επόμενο ψηφίο 1, την τρίτη 3άδα που θα είναι 110 με επόμενο ψηφίο το 1 κλπ. Έτσι ουσιαστικά έχουμε δημιουργήσει την κάθε σειρά του πίνακα αληθείας. Για επιβεβαίωση μπορούμε να συγκρίνουμε τα αποτελέσματα που πήρα με αυτήν την μέθοδο και τον πίνακα 3.3. Έτσι επιβεβαιώνουμε ότι η μέθοδος είναι σωστή. Αυτή η μέθοδος θα χρησιμοποιηθεί στην συνέχεια από το πρόγραμμα που έχουμε κατασκευάσει.

3.4 Γράφος De Bruijn (De Bruijn Graph)

Γράφος de Bruijn (ονομάζεται και ως καλός γράφος – good graph) ορίζεται ένα γράφημα που περιέχει 2^n κόμβους (vertexes), όπου κάθε κόμβος αναπαριστά μια από τις πιθανές καταστάσεις ενός καταχωρητή ολίσθησης με ανάδραση (FSR) τάξης n. Όλοι οι πιθανοί κόμβοι έχουν δύο εξερχόμενα άκρα και δύο εισερχόμενα. Τα εξερχόμενα άκρα αναπαριστούν την μεταβολή της κατάστασης του καταχωρητή σε μια από τις δύο πιθανές επόμενες καταστάσεις του και τα εισερχόμενα άκρα αναπαριστούν τις δύο πιθανές προηγούμενες καταστάσεις του. Ακολουθούν οι γράφοι de Bruijn για FSR τάξης 2 , 3 και 4.



Εικόνα 3.2: Γράφοι de Bruijn τάξης n = 2 και n = 3



Εικόνα 3.3: Γράφος de Bruijn τάξης n =4

Η διαδρομή που διατρέχει κάθε κόμβο του γράφου de Bruijn μια φορά παράγει ακολουθία de Bruijn και είναι γνωστή ως «κύκλος Hamilton». Σε έναν de Bruijn γράφο που περιέχει 2^n κόμβους υπάρχουν $2^{2^{n-1}-n}$ κύκλοι Hamilton, όπου ουσιαστικά μας δίνει τον συνολικό αριθμό των ακολουθιών de Bruijn που υπάρχουν για τάξη n.

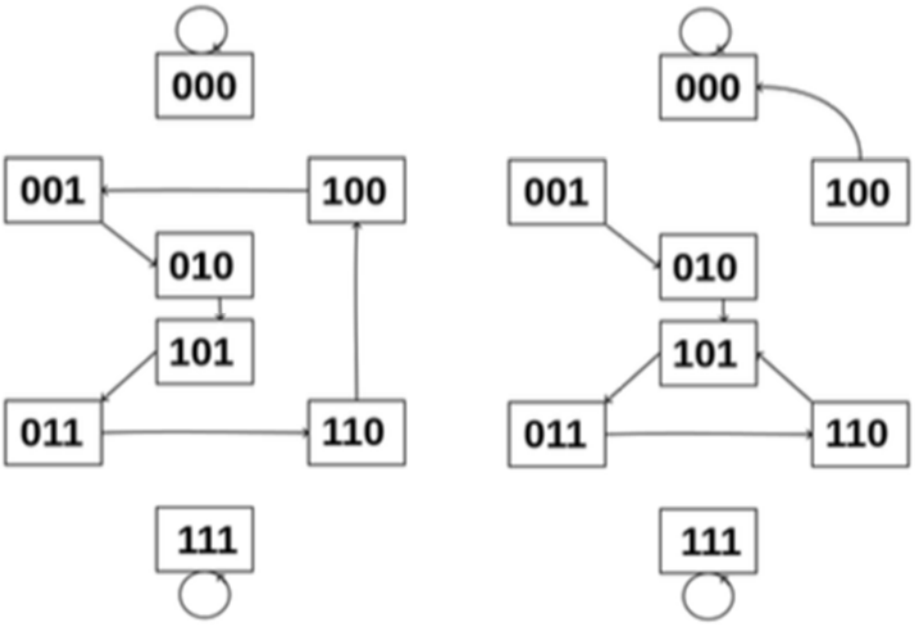
3.5 Ιδιάζων – Μη ιδιάζων (N)LFSR

Ένας (γραμμικός ή μη γραμμικός) καταχωρητής ανάδρασης με ολίσθηση (N)LFSR με συνάρτηση ανάδρασης f καλείται ιδιάζων (singular) εάν υπάρχει τουλάχιστον ένα ζευγάρι συζυγών καταστάσεων του S = (s_0 , s_1 , … s_{n-1}) και Con(S) = ($\bar{s}_0$, s_1 , … s_{n-1}) τέτοιο ώστε f((S)) = f (Con(S)). Έτσι ώστε να υπάρχει τουλάχιστον μία κατάσταση (s_1 , … s_{n-1}, f(S)) που να μπορεί να προκύψει από τις 2 προηγούμενες.

Αυτό για να το κατανοήσουμε καλύτερα μπορούμε να το δούμε από τον δεξιά γράφο της εικόνας 3.5 όπου η κατάσταση (1,0,1) έχει δύο προηγούμενες, την (0,1,0) και την (1,1,0). Εάν **δεν** υπάρχει κάποιο τέτοιο ζευγάρι τότε ο (N)LFSR ονομάζεται **μη ιδιάζων** (non singular). Όλες οι καταστάσεις τότε θα έχουν μία μοναδική προηγούμενη και επόμενη κατάσταση.

Έτσι εάν η συνάρτηση είναι **μη ιδιάζουσα** (non singular) τότε όλες οι αρχικές καταστάσεις του (N)LFSR **θα παράγουν κύκλο**.

Για παράδειγμα, στην εικόνα 3.5 που ακολουθεί μπορούμε να δούμε τον αριστερά γράφο de Bruijn. Αυτός ο γράφος δημιουργείται από τη συνάρτηση ανάδρασης f (x_0, x_1 , x_2) = $x_0 \oplus x_1 \oplus x_1$ x_2 ο οποίος παράγει τους κύκλους (0,0,0), (0,1,0,1,1,0) και (1,1,1). Όλοι οι «υπο»-γράφοι που ικανοποιούν την συνθήκη ότι όλοι οι κόμβοι που υπάρχουν σε αυτούς τους «υπο»-γράφους έχουν μόνο μια εξερχόμενη και εισερχόμενη κατάσταση τότε ανταποκρίνονται σε ένα **μη ιδιάζων** καταχωρητή ολίσθησης με ανάδραση (FSR).



Εικόνα 3.4: Γράφοι de Bruijn τάξης n =3(αριστ. μη ιδιάζον ,δεξιά ιδιάζον)

Για να είναι ένας καταχωρητής ολίσθησης με ανάδραση (FSR) με συνάρτηση ανάδρασης $f(s_0, s_1, \ldots s_{n-1})$, μη ιδιάζων θα πρέπει κάθε ζευγάρι συζυγών καταστάσεων να έχει μοναδική διάδοχη κατάσταση. Έτσι η συνάρτηση ανάδρασης θα πρέπει να εξαρτάται από το $\mathbf{s_0}$ για να είναι μη ιδιάζουσα. Όλες οι συζυγείς ακόλουθες καταστάσεις θα πρέπει να είναι μοναδικές έτσι ώστε ο όρος s_0 να είναι ανεξάρτητος από τους υπόλοιπους και να έχουμε πλέον την συνάρτηση ανάδρασης να είναι ίση με $f(s_0, s_1, \ldots s_{n-1}) = \mathbf{s_0} \oplus g(s_1, s_2, \ldots s_{n-1})$. Η περίοδος που παράγεται από έναν μη ιδιάζωντα FSR μήκους n είναι το πολύ 2^n. Αν όμως είναι ακριβώς 2^n τότε ο FSR είναι NLFSR και η ακολουθία που παράγεται είναι de Bruijn.

3.6 Τροποποιημένη ακολουθία De Bruijn

Μια ακολουθία de Bruijn που έχει παραχθεί από έναν NLFSR n τάξης, διαθέτει γνωστές ιδιότητες τυχαιότητας όπως μεγάλη περίοδο, ισοκατενεμημένο αριθμό «0» και «1» και κάθε n-άδα θα εμφανίζεται ακριβώς μια φορά σε διάστημα μιας περιόδου (ιδιότητα span-n)[11,12,04]. Η γραμμική πολυπλοκότητα της ακολουθίας ορίζεται όπως ξέρουμε ως το μήκος του μικρότερου LFSR που παράγει την ακολουθία. Οι ακολουθίες de Bruijn έχουν υψηλή γραμμική πολυπλοκότητα, και είναι μεγαλύτερη από το ήμισυ της περιόδου τους [16].

Μια **τροποποιημένη ακολουθία de Bruijn** ή **span-n ακολουθία** μπορεί να παραχθεί εάν αφαιρέσουμε ένα μηδενικό από το τμήμα με τα συνεχόμενα n «0» της n τάξης της ακολουθίας de

Bruijn που έχει περίοδο 2^n. Η τροποποιημένη de Bruijn ακολουθία έχει τα ίδια χαρακτηριστικά με την de Bruijn που παράχθηκε , δηλαδή ισοκατανεμημένο αριθμό «0» και «1» , διατηρεί την span – n ιδιότητα αλλά έχει διαφορετική γραμμική πολυπλοκότητα και η οποία μπορεί να μειωθεί δραστικά σε σχέση με την αρχική.

Επίσης μια απλή τεχνική παραγωγής ακολουθίας de Bruijn είναι από ακολουθία που παράγεται από πρωταρχικό LFSR (m – sequence) προσθέτοντας ένα «0» στο τμήμα με τα συνεχόμενα (n -1) μηδενικά της. Η ελάχιστη γραμμική πολυπλοκότητα της παραγόμενη αυτής ακολουθίας de Bruijn είναι $(2^{n-1} + n)$ [13].

Από τα παραπάνω μπορούμε να συμπεράνουμε ότι η γραμμική πολυπλοκότητα διαφοροποιείται δραστικά με την αφαίρεση μόλις ενός μηδενικού από το τμήμα με τα συνεχόμενα μηδενικά και αυτό μας δείχνει πως δεν είναι ικανοποιητικό κριτήριο για την μέτρηση της τυχαιότητας της ακολουθίας. Έτσι η τυχαιότητα θα πρέπει να μετράται με βάση την γραμμική πολυπλοκότητα της τροποποιημένης ακολουθίας de Bruijn γιατί έχουν μόλις ένα bit διαφορά.

3.7 Παραγωγή δυαδικών ακολουθιών de Bruijn

Έχουν μελετηθεί διάφορες τεχνικές και έχουν δημιουργηθεί διάφοροι αλγόριθμοι για την παραγωγή ακολουθιών de Bruijn[08,18].

Ο αλγόριθμος που θα αναφέρουμε στην συνέχεια είναι ο απλούστερος στην περιγραφή του. Έτσι εάν θέλουμε να κατασκευάσουμε μια ακολουθία de Bruijn τάξης n τότε ξεκινάμε με n μηδενικά «0» και προσθέτουμε ένα άσσο «1» κάθε φορά όταν η αντίστοιχη n-αδα που σχηματίστηκε δεν έχει εμφανιστεί αλλιώς προσθέτουμε μηδέν «0». Για παράδειγμα βλέπουμε την ακολουθία de Bruijn τάξης n = 4 που κατασκευάστηκε με αυτόν τον τρόπο [13].

S_r = 0000111101100101

Το μειονέκτημα αυτής της μεθόδου είναι ότι κάθε φορά που εκτελούμε τον συγκεκριμένο αλγόριθμο θα παίρνουμε σαν έξοδο την ίδια de Bruijn ακολουθία.

Ένας ακόμα αλγόριθμος που χρησιμοποιείται για την κατασκευή de Bruijn ακολουθιών αξιοποιεί την τεχνική της **υπαναχώρησης** (**backtracking**). Έστω ότι θέλουμε να κατασκευάσουμε de Bruijn ακολουθία τάξης n: ξεκινάμε με n μηδενικά «0» στο επόμενο βήμα προσθέτουμε «1» και στην συνέχεια προσθέτουμε με τυχαία επιλογή «0» ή «1» αρκεί όμως το νέο bit που εισήγαμε τυχαία να μην έχει αποτέλεσμα η τρέχουσα n-άδα να έχει εμφανιστεί νωρίτερα. Αν κάποια στιγμή φτάσουμε σε τέτοιο σημείο ώστε εάν εισάγουμε είτε «0» είτε «1» η n-άδα που προκύπτει έχει εμφανιστεί νωρίτερα , τότε γυρίζουμε ένα βήμα πίσω και αλλάζουμε την επιλογή που είχαμε κάνει στο αντίστοιχο βήμα κ.ο.κ. Και έτσι προχωρούμε μέχρι να παραχθεί η ακολουθία de Bruijn [19].

Το πλεονέκτημα της εφαρμογής αυτού του αλγορίθμου είναι ότι κάθε φορά που εκτελείται αναμένεται να προκύπτει και μια διαφορετική ακολουθία de Bruijn, λόγω της τυχαίας κάθε φορά επιλογής του επόμενου bit.

Πέρα όμως από την κατασκευή de Bruijn ακολουθιών θα δούμε στην συνέχεια δύο ακόμα τρόπους με τους οποίους μπορούμε να κατασκευάσουμε νέες ακολουθίες de Bruijn από ήδη υπάρχουσες de Bruijn ακολουθίες.

3.7.1 Ένωση και διαχωρισμός κύκλων (cycle joining and splitting)

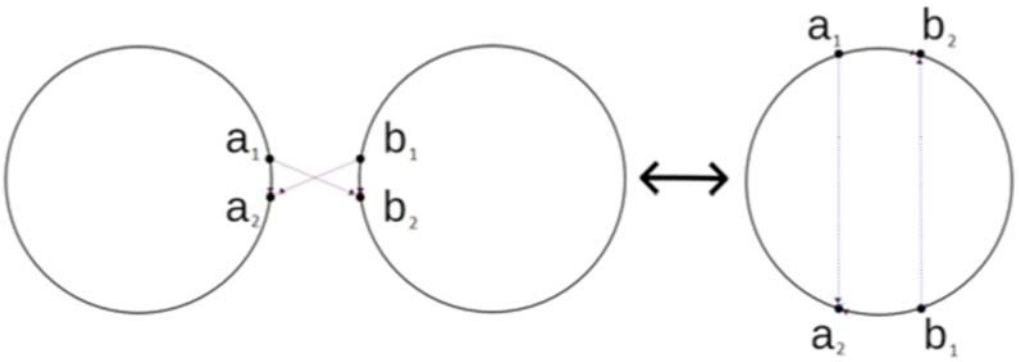
Δεδομένης μιας μη ιδιάζουσας συνάρτησης ανάδρασης $f(s_0 , s_1 , \dots s_{n-1}) = s_0 \oplus g(s_1 , s_2 , \dots s_{n-1})$, εάν αλλάξουμε μια καταχώρηση από τον πίνακα αληθείας της g τότε ουσιαστικά μεταβάλλουμε τη διάδοχη μιας συγκεκριμένης κατάστασης του FSR που παράγει την εν λόγω ακολουθία. Πριν προχωρήσουμε στην μεθοδολογία, θα ορίσουμε κάποιες βασικές έννοιες.

Για μια κατάσταση $S_i = (s_{i+n-1}, \dots , s_{i+1}, s_i)$ ορίζουμε την **συζυγή** (**conjugate**) κατάσταση $\overline{S_i} = (s_{i+n-1}, \dots , s_{i+1}, s_i')$, όπου s_i' είναι η **συμπληρωματική** (**complement**) κατάσταση της s_i.

Δύο κύκλοι C1 και C2 λέγονται **γειτονικοί** (**adjacent**) όταν η κατάσταση Si ανήκει στον κύκλο C1 και η συζυγής κατάσταση $\overline{S_i}$ ανήκει στον κύκλο C2. Εάν έχουμε δύο γειτονικούς κύκλους και αντιμεταθέσουμε τις διάδοχες καταστάσεις των Si και $\overline{S_i}$ θα έχουμε σαν αποτέλεσμα να πάρουμε

έναν νέο μεγαλύτερο κύκλο C που θα περιέχει όλες τις καταστάσεις των κύκλων C1 και C2 . Αυτή η μέθοδος λέγεται ένωση κύκλων [19]. Στην συνέχεια θα δούμε παράδειγμα για να το κατανοήσουμε καλύτερα.

Για μια κατάσταση a1 = (1, k) – όπου ως k υποδηλώνεται μία ομάδα από n-1 bits - και διάδοχης κατάστασης a2 = (k, c) – όπου ως c υποδηλώνεται ένα bit - που βρίσκεται στον κύκλο C1 καθώς επίσης για κατάσταση b1 = (0, k) με διάδοχη κατάσταση b2 = (κ, $\bar{c}$) που βρίσκεται στον γειτονικό κύκλο C2 , αλλάζουμε τα επόμενα bit από τις a1 και b1: άρα, μετά την a1 θέτουμε το $\bar{c}$ (αντί για το c) και μετά την b1 θέτουμε το c (αντί για το $\bar{c}$) . Αυτή η αλλαγή ισοδυναμεί με αλλαγές σε δύο συγκεκριμένες γραμμές του πίνακα αληθείας της αντίστοιχης συνάρτησης. Με αυτόν τον τρόπο ουσιαστικά ενώνουμε τους δύο κύκλους, αφού η διάδοχος κατάσταση της a1 αλλάζει σε b2 και η διάδοχος της b1 σε a2. Έτσι αυτή η μέθοδος μπορεί να χρησιμοποιηθεί για να ενώσει δύο κύκλους εάν οι δύο καταστάσεις a1 και b1 είναι σε διαφορετικούς κύκλους. Εάν είναι στον ίδιο κύκλο τότε ο αρχικός κύκλος θα χωριστεί σε δύο. Ακολουθεί η εικόνα 3.5 που μας δίνει μια οπτική αναπαράσταση της διαδικασίας.



Εικόνα 3.5: Ένωση και διαχωρισμός κύκλων

3.7.2 Cross join pairs

Μια άλλη προσέγγιση για την κατασκευή ακολουθιών de Bruijn βασίζεται στον διαχωρισμό ενός κύκλου C σε δύο κύκλους C_1 και C_2 και στην συνέχεια στην κατάλληλη συνένωση τους σε ένα νέο κύκλο D [23].

Τα cross join pairs είναι 2 ζευγάρια συζυγών καταστάσεων $\{a , \bar{a}\}$ και $\{b , \bar{b}\}$ σε έναν κύκλο C. Εάν αντιμεταθέσουμε τις διάδοχες καταστάσεις του $\{a , \bar{a}\}$ ή του $\{b , \bar{b}\}$ χωρίζεται ο κύκλος C σε δύο νέους κύκλους C_1 και C_2 . Στην συνέχεια αντιμεταθέτουμε τις διάδοχες καταστάσεις του

ζευγαριού που δεν χρησιμοποιήθηκε για να χωρίσει το κύκλο C και έτσι ουσιαστικά θα ενώσει τους δύο κύκλους C_1 και C_2 σε έναν νέο κύκλο D.

Αποδεικνύεται ότι 2 συζυγή ζεύγη $\{a\,,\bar{a}\}$ και $\{b\,,\overline{b}\,\}$ της ακολουθίας z μπορούν να χρησιμοποιηθούν στην ως άνω κατασκευή – και αποτελούν ένα «cross join pair» - εάν υπάρχει κυκλική μετατόπιση της z τέτοια ώστε να έχει την εξής μορφή:

$$\dots\ a, \dots b, \dots\ \bar{\alpha}, \dots \overline{b}, \dots$$

Από το παραπάνω βλέπουμε ότι αντιμεταθέτοντας τις διάδοχες καταστάσεις του $a, \bar{\alpha}$ έχει σαν αποτέλεσμα να χωρίζεται η ακολουθία z και στην συνέχεια συνενώνουμε τους δύο νέους κύκλους σε έναν νέο αντιμεταθέτοντας τις διάδοχες καταστάσεις των $b\,,\overline{b}$.

Ο αριθμός των cross join pairs για μια ακολουθία μεγίστου μήκους (m-sequence) έχει αποδειχθεί ότι είναι ίσος με $\frac{(2^{n-1}-1)(2^{n-1}-2)}{6}$ [14].

Κεφάλαιο 4
Κρυπτογραφικές ιδιότητες γεννητριών De Bruijn

4.1 Εισαγωγή

Οι NLFSR έχουν προσελκύσει το ερευνητικό ενδιαφέρον για τον σχεδιασμό κρυπτοσυστημάτων όπως για παράδειγμα οι γεννήτριες κλειδοροών στους αλγόριθμους ροής. Δυστυχώς όμως, λόγω της εγγενούς δυσκολίας τους, οι NLFSR δεν έχουν μελετηθεί τόσο διεξοδικά ακόμα από την ερευνητική κοινότητα (σε σχέση με τους LFSR). Δεν γνωρίζουμε πώς να παράγουμε εγγυημένα ψευδοτυχαίες ακολουθίες με την χρήση NLFSR οι οποίες να έχουν τη μέγιστη δυνατή περίοδο – δηλαδή, να είναι ακολουθίες De Bruijn. Αυτό που γνωρίζουμε είναι ότι παράγουν ακολουθίες με μεγάλη γραμμική πολυπλοκότητα. Έχει αποδειχθεί ότι οι de Bruijn ακολουθίες έχουν γενικά μεγάλη γραμμική πολυπλοκότητα, η οποία είναι μεγαλύτερη από το ήμισυ της περιόδου της ακολουθίας. Εάν όμως αφαιρέσουμε ένα μηδενικό από το τμήμα με τα συνεχόμενα μηδενικά, τότε παίρνουμε την τροποποιημένη ακολουθία de Bruijn. Το παράδοξο είναι ότι η

τροποποιημένη de Bruijn έχει τα ίδια χαρακτηριστικά με την αρχική, είναι δηλαδή ισοβαρής, κάθε n-άδα εμφανίζεται ακριβώς μια φορά μέσα στην περίοδο αλλά πλέον δεν έχει την ίδια γραμμική πολυπλοκότητα με την αρχική.

Στο παρόν κεφάλαιο θα μελετήσουμε τις κρυπτογραφικές ιδιότητες που εμφανίζουν οι συναρτήσεις ανάδρασης εκείνων των NLFSR οι οποίοι παράγουν De Bruijn ακολουθίες. Συγκεκριμένα, αρχικά θα υπολογιστούν τέτοιες συναρτήσεις και στη συνέχεια θα εξεταστούν οι κρυπτογραφικές τους ιδιότητες, όπως ο βαθμός, η μη γραμμικότητα, η ανθεκτικότητα σε συσχετίσεις κ.α. Περαιτέρω, βασιζόμενοι στις κατασκευές De Bruijn συναρτήσεων που περιεγράφηκαν στο Κεφάλαιο 3, θα δούμε τη σχέση που υπάρχει, ως προς τις προαναφερθείσες κρυπτογραφικές ιδιότητες, μεταξύ των συναρτήσεων ανάδρασης των αντίστοιχων NLFSR.

4.2 Κριτήρια αποτίμησης ασφάλειας ακολουθιών de Bruijn.

Όπως έχουμε δει και στα προηγούμενα κεφάλαια , στους καταχωρητές ανάδρασης με ολίσθηση (είτε γραμμικούς είτε μη γραμμικούς) μπορούμε να περιγράψουμε την ανάδρασή τους με μια λογική συνάρτηση f . Η αναπαράσταση της συνάρτησης f γίνεται με την αλγεβρική κανονική μορφή της (Algebraic Normal Form - ANF). Εάν η ακολουθία που παράγεται από NLFSR n μεγέθους έχει περίοδο 2^n και περιέχει κάθε n-αδα του συνόλου των τιμών της τότε πρόκειται, όπως έχουμε ήδη δει, για de Bruijn ακολουθία.

Σκοπός μας στην εργασία αυτή είναι εξετάσουμε διάφορα **κρυπτογραφικά χαρακτηριστικά** λογικών συναρτήσεων οι οποίες παράγουν ακολουθίες de Bruijn. Οι κρυπτογραφικές συναρτήσεις, πέραν του υψηλού βαθμού, πρέπει να ικανοποιούν και κάποια άλλα κρυπτογραφικά χαρακτηριστικά.

Τα κρυπτογραφικά χαρακτηριστικά που θα εξετάσουμε για να αποτιμήσουμε την ασφάλεια των γεννητριών των ακολουθιών είναι τα εξής: η μη γραμμικότητα (nonlinearity), αλγεβρική ανθεκτικότητα (algebraic immunity), ανθεκτικότητα στις συσχετίσεις (correlation immunity) και η ύπαρξη γραμμικών δομών (linear structure).

Με την **μη γραμμικότητα** (**nonlinearity**) μιας συνάρτησης f ορίζουμε το μικρότερο πλήθος θέσεων στον πίνακα αληθείας της, οι οποίες εάν μεταβληθούν θα μετατρέψουν τη συνάρτηση σε

γραμμική (linear). Γενικά αποζητούμε υψηλή μη γραμμικότητα ώστε να καθίστανται δύσκολες προσπάθειες κρυπτανάλυσης που βασίζονται σε προσεγγίσεις της f με γραμμικές συναρτήσεις (επιθέσεις γραμμικών προσεγγίσεων – βλ. Κεφάλαιο 3).

Με την **αλγεβρική ανθεκτικότητα** (algebraic immunity) μπορούμε να αποτιμήσουμε το κατά πόσο μια λογική συνάρτηση f που παράγει ακολουθία de Bruijn είναι ανθεκτική στις αλγεβρικές επιθέσεις. Ειδικότερα θα αναζητήσουμε την συνάρτηση φίλτρου g (εκμηδενιστής) με τον μικρότερο βαθμό που εάν πολλαπλασιαστεί με την συνάρτηση f (ή την συμπληρωματική της) την μηδενίζει.

$$f*g = 0 \text{ ή } (f \oplus 1)*g = 0$$

Σκοπός των αλγεβρικών επιθέσεων είναι να μειωθεί ο βαθμός των μη γραμμικών εξισώσεων που έχουν ως άγνωστη μεταβλητή το μυστικό κλειδί. Όσο πιο μεγάλου βαθμού είναι ο μικρότερος εκμηδενιστής g τόσο πιο ανθεκτική είναι η συνάρτηση f σε τέτοιου είδους επίθεση. Η αλγεβρική ανθεκτικότητα για κάθε λογική συνάρτηση n μεταβλητών είναι μικρότερη ή ίση του $\lceil n/2 \rceil$ [06] και κατ' επέκταση δεν πρέπει ο εκμηδενιστής g, στην ιδανική περιπτωση, να έχει τιμή χαμηλότερη από αυτή (βλ. Κεφάλαιο 3).

Μια λογική συνάρτηση είναι **ανθεκτική στις συσχετίσεις** (correlation immune) t βαθμού όταν η κατανομή της εξόδου της (δηλαδή η πιθανότητα να είναι 0 ή 1 στην έξοδό της) μένει αμετάβλητη ακόμα και αν θέσουμε συγκεκριμένες τιμές σε οποιεσδήποτε t μεταβλητές της εισόδου της.

Εάν ο βαθμός t της ανθεκτικότητας σε συσχετίσεις είναι αρκετά μικρός (στη χειρότερη περίπτωση, μηδενικός), τότε η λογική συνάρτηση είναι ευάλωτη σε επίθεση τύπου «divide-and-conquer» όπου εκμεταλλεύεται αυτή την ευπάθεια η οποία έχει αποδειχθεί από τον Siegenthaler [25].

Επίσης η **γραμμική δομή** (**linear structure**) στις λογικές συναρτήσεις παίζει σημαντικό ρόλο ώστε να αποτιμηθεί η αδυναμία αυτών των συναρτήσεων ως προς τις κρυπτογραφικές τους εφαρμογές[07]. Οι γραμμικές δομές περιλαμβάνουν μια πολύ ευρεία έννοια της μη γραμμικότητας, αφού μια συνάρτηση f χωρίς γραμμικές δομές δεν μπορεί να αντιστοιχηθεί με μια μερικώς γραμμική συνάρτηση μέσω ενός γραμμικού μετασχηματισμού[40]. Επίσης η

παρουσία των γραμμικών δομών μπορεί να αξιοποιηθεί σε επιθέσεις που βασίζονται στην διαφορική κρυπτανάλυση[24].

4.3 Μεθοδολογία που ακολουθήθηκε

Η μεθοδολογία που ακολουθήσαμε για την διενέργεια των πειραμάτων μας είναι η εξής: Κατ' αρχάς, χρησιμοποιήθηκε η μέθοδος υπαναχώρησης (backtracking) για την κατασκευή των αρχικών ακολουθιών de Bruijn τάξης N =4, 5, 6, 7, 8, 9. Υπολογίσθηκε η αλγεβρική κανονική μορφή των εκάστοτε λογικών συναρτήσεων που παρήγαγαν τις ακολουθίες de Bruijn και στην συνέχεια έγινε αποτίμηση των κρυπτογραφικών χαρακτηριστικών τους με βάση την μη γραμμικότητα (non linearity), αλγεβρική ανθεκτικότητα (algebraic immunity), ανθεκτικότητα στις συσχετίσεις (correlation immunity) και την ύπαρξη γραμμικής δομής (linear structure).

Στην συνέχεια έγινε χρήση των μεθόδων κατασκευής νέων de Bruijn ακολουθιών από ήδη υπάρχουσες de Bruijn με την μεθοδολογία των cross join pairs και cycle joining & splitting. Ακολούθως υπολογίσθηκε η αλγεβρική κανονική μορφή των λογικών συναρτήσεων που τις παράγουν και στην συνέχεια έγινε αποτίμηση των ίδιων κρυπτογραφικών χαρακτηριστικών τους, όπως έγινε και με την μεθοδολογία της υπαναχώρισης (backtracking).

4.3.1 SageMath

Το SageMath [30] είναι αλγεβρικό υπολογιστικό σύστημα ανοιχτού κώδικα το οποίο καλύπτει πολλές πτυχές των μαθηματικών όπως είναι η άλγεβρα, συνδυαστική, αριθμητική ανάλυση, θεωρία γράφων και πολλά άλλα. Βασίζεται στην γλώσσα προγραμματισμού Python και χρησιμοποιεί πολλά πακέτα ανοικτού κώδικα όπως το NumPy, SciPy, matplotlib, GAP, R, FLINT και αρκετά ακόμα. Η πρώτη έκδοση δημοσιεύτηκε στις 24 Φεβρουαρίου 2005 από τον καθηγητή μαθηματικών William Stein του πανεπιστημίου της Washington. Σκοπός του SageMath είναι να αποτελέσει μια εναλλακτική δωρεάν επιλογή των προγραμμάτων Mathematica, Maple, MATLAB κ.α.

Στην παρούσα διατριβή θα γίνει εκτενής χρήση του προγράμματος SageMath για να υπολογιστούν διάφορα κρυπτογραφικά χαρακτηριστικά των συναρτήσεων οι οποίες παράγουν de Bruijn ακολουθίες, όπως είναι ο βαθμός, η μη γραμμικότητα, αλγεβρική ανθεκτικότητα, ανθεκτικότητα στις συσχετίσεις καθώς και η γραμμική δομή. Για να γίνει αυτό μελετήθηκε εις

βάθος ο τρόπος λειτουργίας του προγράμματος ώστε να αναπτυχθεί ο αντίστοιχος κώδικας για τον υπολογισμό των διαφόρων κρυπτογραφικών χαρακτηριστικών των λογικών συναρτήσεων.

4.3.2 Περιγραφή αλγορίθμων

Για να μπορέσουμε να εξετάσουμε τα κρυπτογραφικά χαρακτηριστικά λογικών συναρτήσεων που παράγουν ακολουθίες de Bruijn θα ακολουθήσουμε την εξής μεθοδολογία :

Αρχικά, δεδομένου ότι δεν υπάρχουν γνωστές τεχνικές κατασκευής NLFSR που να παράγουν De Bruijn ακολουθίες, χρειαζόμαστε μία διαδικασία παραγωγής ακολουθιών De Bruijn. Για το σκοπό αυτό, θα χρησιμοποιήσουμε έναν αλγόριθμο που αξιοποιεί την τεχνική της υπαναχώρησης (backtracking), αξιοποιώντας ένα πρόγραμμα που αναπτύχθηκε σε γλώσσα προγραμματισμού C που υλοποιεί τη σχετική ιδέα που περιγράφεται στο[19] και αναφέρθηκε στην παράγραφο 3.6, έτσι ώστε να παράγουμε ακολουθίες de Bruijn τάξης n =4, 5, 6, 7, 8 και 9. Ο λόγος επιλογής αυτής της τεχνικής είναι ότι είναι πιθανοτικός αλγόριθμος, δηλαδή κάθε φορά που εκτελείται παράγει και διαφορετική ακολουθία De Bruijn.

Στην συνέχεια θα κάνουμε χρήση του προγράμματος SageMath έτσι ώστε από τις de Bruijn ακολουθίες που έχουν παραχθεί να γίνει υπολογισμός της λογικής συνάρτησης που τις παράγει (δηλαδή, με άλλα λόγια, να υπολογίσουμε τη συνάρτηση ανάδρασης που αντιστοιχεί σε εκείνον τον NLFSR που παράγει την εν λόγω De Bruijn ακολουθία). Επειδή όμως το SageMath δέχεται σαν είσοδο ακολουθία με σειρά των ψηφίων όπως ορίζεται από τον πίνακα αληθείας της, γι' αυτό το λόγο θα πρέπει να μετασχηματίσουμε τις ακολουθίες που έχουν παραχθεί από το πρόγραμμα που αναφέραμε σύμφωνα με τον πίνακα αληθείας τους. Για παράδειγμα το SageMath ορίζει για πίνακα αληθείας τριών μεταβλητών x_0, x_1, x_2 τον ακόλουθο πίνακα της εικόνας 4.1.

x_2	x_1	x_0
0	0	0
0	0	1
0	1	0

0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Εικόνα 4.1: Πίνακας αληθείας λογικής συνάρτησης 3 μεταβλητών

Είναι προφανές λοιπόν ότι θέτοντας απλά την De Bruijn ακολουθία ως σειρά εξόδου του πίνακα αληθείας της λογικής συνάρτησης, δεν θα υπολογιστεί από το sage η λογική συνάρτηση εκείνη που παράγει πραγματικά την ακολουθία.. Έτσι δημιουργήσαμε πρόγραμμα σε γλώσσα προγραμματισμού python με την ονομασία sorting_truth_table.py που δέχεται ως είσοδο την de Bruijn ακολουθία που έχει παραχθεί και παράγει ως έξοδο τη σειρά των ψηφίων με βάση την δομή του πίνακα αληθείας που ορίζει το πρόγραμμα SageMath.

Η αλγοριθμική λογική που ακολουθεί το πρόγραμμα sorting_truth_table.py είναι η ακόλουθη:

Έστω ακολουθία de Bruijn που παράγεται από λογική συνάρτηση n μεταβλητών. Χωρίζουμε την ακολουθία σε n-άδες με περιτύλιξη έτσι ώστε να πάρουμε όλες τις πιθανές n-άδες που δημιουργούνται. Σε κάθε παραγόμενη n-άδα αντιστοιχίζουμε το αμέσως δεξιότερο bit της ακολουθίας. Έτσι έχουμε αντιστοίχιση n-άδας με το επόμενο δεξιότερο ψηφίο. Στην συνέχεια αντιστρέφουμε τα ψηφία της κάθε n-άδας (από τα αριστερά προς τα δεξιά), διατηρώντας την υπάρχουσα αντιστοίχιση. Έτσι έχουμε δημιουργήσει την κάθε γραμμή του πίνακα αληθείας που αναζητούσαμε. Στην συνέχεια την κάθε σειρά την τοποθετούμε σε αύξουσα σειρά με βάση την τιμή n-άδα (από την μικρότερη προς την μεγαλύτερη) και έτσι παίρνουμε τον πίνακα αληθείας με την σειρά όπου μπορεί να επεξεργασθεί σωστά από το πρόγραμμα SageMath.

Για παράδειγμα, ας θεωρήσουμε την ακολουθία de Bruiijn 00010111 τάξης 3 (δηλαδή παράγεται από συνάρτηση 3 μεταβλητών). Με βάση τη μεθοδολογία που αναφέραμε, θα πάρουμε όλες τις πιθανές n-άδες με αντιστοίχιση το αμέσως δεξιότερο ψηφίο. Έτσι θα έχουμε τα εξής παραγόμενα ζευγάρια:

000-1 , 001-0, 010 -1, 101-1, 011-1, 111-0, 110-0, 100-0.

Στην συνέχεια θα αντιστρέψουμε τα ψηφία της κάθε 3άδας και θα έχουμε τα παρακάτω ζευγάρια :

000-1, 100-0, 010-1, 101-1, 110-1, 111-0, 011-0, 001-0

Με αυτά τα ζευγάρια έχουμε ουσιαστικά περιγράψει την κάθε γραμμή του πίνακα αληθείας που θέλουμε και το μόνο που μένει είναι να ταξινομήσουμε το κάθε ζευγάρι με βάση αύξουσα σειρά της κάθε 3άδας. Έτσι παίρνουμε τα ζευγάρια με την ακόλουθη σειρά:

000-1 , 001-0 , 010-1 , 011-0 , 100-0 , 101-1 ,110-1, 111-0

Ουσιαστικά η νέα ταξινομημένη σειρά που παίρνουμε θα είναι η ακόλουθη:

«10100110»

η οποία είναι, προφανώς, η έξοδος του πίνακα αληθείας της συνάρτησης εκείνης η οποία, αν τεθεί ως συνάρτηση ανάδρασης ενός NLFSR, θα παραχθεί η αρχική De Bruijn ακολουθία.

Και το οποίο το επιβεβαιώνουμε με το πρόγραμμα sorting_truth_table.py που δημιουργήσαμε και η παραγόμενη ακολουθία εισόδου θα είναι «0010111» .

```
C:\Users\johnk\OneDrive\Desktop\Python - C Program>python sorting_truth_table.py
['1', '0', '1', '0', '0', '1', '1', '0']
1,0,1,0,0,1,1,0
```

Εικόνα 4.1: Έξοδος του προγράμματος sorting_truth_table.py για ακολουθία εισόδου 0010111

Πλέον εφόσον έχουμε τις de Bruijn ακολουθίες σε μορφή που μπορεί να επεξεργασθεί κατάλληλα το SageMath, έχουμε τη δυνατότητα να υπολογίσουμε την αλγεβρική κανονική μορφή της λογικής συνάρτησης που παράγει την εκάστοτε ακολουθία. Αυτό το πετύχαμε με τη χρησιμοποίηση του εξής κώδικα στο πρόγραμμα SageMath:

```
In [3]: sage: from sage.crypto.boolean_function import BooleanFunction
        sage: B = BooleanFunction([1,0,1,0,0,1,1,0])
        sage: P = B.algebraic_normal_form()
        sage: P
Out[3]: x0 + x1*x2 + x2 + 1
```

Εικόνα 4.2: Κώδικας από SageMath για υπολογισμό αλγεβρικής κανονικής μορφής

Έτσι από την εικόνα 4.2 βλέπουμε ότι με την χρήση απλού κώδικα στο SageMath δίνοντας σαν είσοδο την «κανονικοποιημένη» - υπό την έννοια που περιγράφηκε ανωτέρω - de Bruijn ακολουθία «10100110» παίρνουμε σαν έξοδο την ακόλουθη αλγεβρική κανονική μορφή της συνάρτησης f, η οποία είναι η «x0+x1*x2+x2+1» που παράγει την ακολουθία «0010111».

Για να γίνει εξέταση των κρυπτογραφικών χαρακτηριστικών των λογικών συναρτήσεων θα χρησιμοποιήσουμε και πάλι το SageMath όπου και θα εξετάσουμε την **μη γραμμικότητα** (non linearity), **αλγεβρική ανθεκτικότητα** (algebraic immunity), **ανθεκτικότητα στις συσχετίσεις** (correlation immunity) και **γραμμική δομή** (linear structure) τους. Θα ακολουθήσουν παραδείγματα από τους κώδικες που χρησιμοποιήθηκαν στο SageMath για τον υπολογισμό όλων των παραπάνω κρυπτογραφικών χαρακτηριστικών των λογικών συναρτήσεων.

Σε συνέχεια του προηγούμενου παραδείγματος, για τον υπολογισμό της αλγεβρικής ανθεκτικότητας χρησιμοποιήθηκε ο εξής κώδικας:

```
In [8]: sage: from sage.crypto.boolean_function import BooleanFunction
        sage: R.<x0,x1,x2> = BooleanPolynomialRing(3)
        sage: B = BooleanFunction(x0 + x1*x2 + x2 + 1)
        sage: B.algebraic_immunity(annihilator=True)
Out[8]: (2, x0*x2 + x0)
```

Εικόνα 4.3: Κώδικας από SageMath για υπολογισμό αλγεβρικής ανθεκτικότητας

Εδώ βλέπουμε ότι αλγεβρική ανθεκτικότητα είναι βαθμού 2 και ο εκμηδενιστής g της λογικής συνάρτησης f βαθμού 2 είναι ο ακόλουθος : g = x0*x2 + x0

Ενώ για τον υπολογισμό της ανθεκτικότητας στις συσχετίσεις χρησιμοποιήσαμε τον παρακάτω κώδικα:

```
In [9]: sage: from sage.crypto.boolean_function import BooleanFunction
        sage: R.<x0,x1,x2> = BooleanPolynomialRing(3)
        sage: B = BooleanFunction(x0 + x1*x2 + x2 + 1)
        sage: B.correlation_immunity()

Out[9]: 0
```

Εικόνα 4.4: Κώδικας από SageMath για υπολογισμό ανθεκτικότητας στις συσχετίσεις

Έτσι βλέπουμε ότι η ανθεκτικότητα στις συσχετίσεις της λογικής συνάρτησης x0 + x1*x2 + x2 + 1 είναι μηδενική, το οποίο σημαίνει ότι έστω και 1 bit να κρατήσουμε σταθερό από την λογική συνάρτηση εισόδου τότε θα αλλάξει η πιθανότητα ισοκατανομής των bit εξόδου.

Επίσης ο υπολογισμός της γραμμικής δομής έγινε με βάση τον κώδικα:

```
In [10]: sage: from sage.crypto.boolean_function import BooleanFunction
         sage: R.<x0,x1,x2> = BooleanPolynomialRing(3)
         sage: B = BooleanFunction(x0 + x1*x2 + x2 + 1)
         sage: B.has_linear_structure()

Out[10]: True
```

Εικόνα 4.5: Κώδικας από SageMath για υπολογισμό ύπαρξης γραμμικής δομής

Από την έξοδο του προγράμματος της εικόνας 4.5 βλέπουμε ότι η συγκεκριμένη λογική συνάρτηση περιέχει τουλάχιστον μια γραμμική δομή.

Συνολικά με την μέθοδο της υπαναχώρησης έχουμε παράγει 43 ακολουθίες de Bruijn, εκ των οποίων 7 ακολουθίες είναι τάξης n= 4 , 8 είναι τάξης n=5 , 6 είναι τάξης n=6 , 7 είναι τάξης n=7 , 8 είναι τάξης n=8 και 8 είναι τάξης n=9.

Όλες οι παραγόμενες ακολουθίες είχαν τα εξής χαρακτηριστικά:

α) Υπάρχει τουλάχιστον μια γραμμική δομή (linear structure).

β) Η ανθεκτικότητα στις συσχετίσεις (correlation immunity) ήταν μηδενικού βαθμού.

γ) Η παραγόμενη αλγεβρική κανονική μορφή (algebraic normal form) έχει τον μέγιστο βαθμό n-1 για κάθε τάξη n=4, 5, 6, 7, 8 και 9.

Όλες οι ακολουθίες που παράγονται από συναρτήσεις τάξης n=4 έχουν μη γραμμικότητα βαθμού 2 και αλγεβρική ανθεκτικότητα τάξης 2 (δηλαδή τη μέγιστη δυνατή).

Για τάξη n = 5, οι 3 ακολουθίες έχουν μη γραμμικότητα βαθμού 10 και οι 5 βαθμού 6. Επίσης η αλγεβρική ανθεκτικότητα των 5 ακολουθιών είναι τάξης 2 και 3 έχουν τάξη 3 (μέγιστη).

Για τάξη n = 6, η 1 ακολουθία έχει μη γραμμικότητα βαθμού 14, οι 4 ακολουθίες μη γραμμικότητα 18 και 1 έχει μη γραμμικότητα 22. Οι 5 ακολουθίες έχουν αλγεβρική ανθεκτικότητα τάξης 3 (όπου είναι και η μέγιστη δυνατή) και η 1 μια είναι τάξης 2.

Για τάξη n = 7, οι 4 ακολουθίες έχουν μη γραμμικότητα βαθμού 46 και οι 3 έχουν βαθμού 42. Όλες οι ακολουθίες είχαν αλγεβρική ανθεκτικότητα τάξης 3 (δηλαδή καμία δεν είχε τη μέγιστη δυνατή).

Για τάξη n = 8, 3 ακολουθίες έχουν μη γραμμικότητα βαθμού 94 , 4 έχουν βαθμού 98 και 1 έχει βαθμού 102. Όλες οι ακολουθίες έχουν αλγεβρική ανθεκτικότητα τάξης 4 όπου και είναι η μέγιστη δυνατή.

Για τάξη n = 9, 2 ακολουθίες έχουν μη γραμμικότητα βαθμού 202, 4 έχουν μη γραμμικότητα 206 και 2 έχουν μη γραμμικότητα 210. Οι 5 έχουν αλγεβρική ανθεκτικότητα τάξης 4 και οι 3 έχουν την μέγιστη που είναι τάξης 5.

Πλέον αφού έχει γίνει υπολογισμός των κρυπτογραφικών χαρακτηριστικών των λογικών συναρτήσεων τάξης n=4, 5, 6, 7, 8 και 9 που παράγουν de Bruijn ακολουθίες, θα χρησιμοποιήσουμε τους 2 τρόπους που αναφέραμε στο κεφάλαιο 3.7 όπου δημιουργούμε νέα de Bruijn ακολουθία βασιζόμενοι είτε σε μία άλλη υπάρχουσα ακολουθία De Bruijn είτε μετά από συνένωση δύο υπαρχουσών de Bruijn ακολουθιών. Αυτό θα γίνει με σκοπό να υπολογισθούν οι νέες λογικές συναρτήσεις που παράγουν τις νέες de Bruijn και να μελετηθούν τα κρυπτογραφικά χαρακτηριστικά τους, συγκριτικά με τα κρυπτογραφικά χαρακτηριστικά των αρχικών συναρτήσεων. Απώτερος στόχος είναι να γίνει αποτίμηση της αποτελεσματικότητας των δύο μεθόδων παραγωγής νέων de Bruijn με βασικά κριτήρια τα κρυπτογραφικά χαρακτηριστικά των λογικών συναρτήσεων που τις παράγουν.

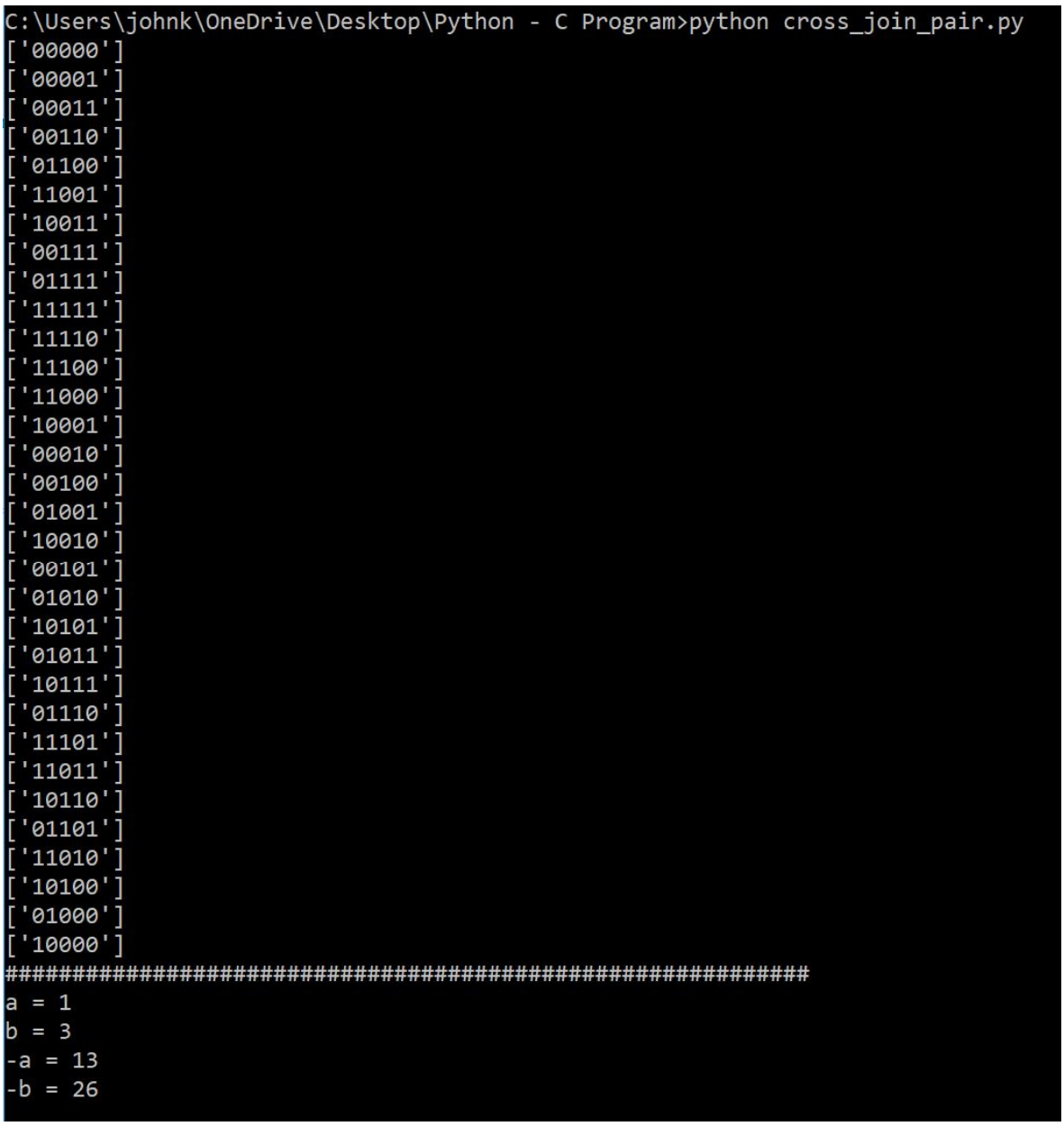
4.3.2.1 Παραγωγή νέας de Bruijn ακολουθίας με μεθοδολογία cross join pair

Όπως αναφέραμε και στην παράγραφο 3.7.2 για τα cross join pairs, εφόσον έχουμε την de Bruijn ακολουθία τάξης n και άρα γνωρίζουμε τον κύκλο των καταστάσεων που την απαρτίζουν – δηλαδή όλες τις n-αδες που υπάρχουν με την σειρά που εμφανίζονται στην ακολουθία - επιλέγουμε δύο ζευγάρια συζυγών καταστάσεων $\{a, \bar{a}\}$ και $\{b, \overline{b}\}$ τέτοια ώστε να εκπληρώνουν την παρακάτω απαίτηση $a \dots b \dots \bar{a} \dots \overline{b}$. Εφόσον επιλέξουμε τα ζευγάρια κατάλληλα, θα τα εντοπίσουμε στην υπάρχουσα ακολουθία και θα ξεκινήσουμε να δημιουργούμε την νέα ακολουθία de Bruijn τοποθετώντας τις υπάρχουσες καταστάσεις σε σειρά όπως στην αρχική έως ότου εντοπίσουμε μια από τις 4 ακολουθίες που έχουμε επιλέξει. Μόλις την εντοπίσουμε αντιστρέφουμε το δεξιότερο bit της επόμενης κατάστασης και συνεχίζουμε τοποθετώντας τις επόμενες καταστάσεις της αρχικής από το επόμενο σημείο της αλλαγής έως ότου γίνουν όλες οι αλλαγές και έτσι θα έχουμε πάρει την νέα de Bruijn.

Για να υλοποιήσουμε την τεχνική αυτή δημιουργήσαμε πρόγραμμα σε γλώσσα Python τέτοιο ώστε για δοθείσα ακολουθία na υπολογίζει όλες της καταστάσεις της, καθώς επίσης και τις καταστάσεις των συζυγών ζευγών που έχουμε επιλέξει. Επίσης υπολογίζει τις θέσεις τους στον κύκλο καταστάσεων. Το πρόγραμμα ονομάζεται cross_join_pair.py και μπορούμε να βρούμε τον κώδικα στο παράρτημα 3.

Για λόγους καλύτερης κατανόησης ακολουθεί παράδειγμα με de Bruijn ακολουθία τάξης n = 5 και η οποία είναι η ακόλουθη : «00000110011111000100101011101101»

Χρησιμοποιούμε το πρόγραμμα που δημιουργήθηκε (cross_join_pair.py) και λαμβάνουμε σαν έξοδο όλες τις καταστάσεις του κύκλου καθώς και τις θέσεις των 2 συζυγών ζευγαριών $\{a, \bar{a}\}$ & $\{b, \overline{b}\}$ που έχουμε επιλέξει όπου και ικανοποιούν την απαίτηση $a \dots b \dots \bar{a} \dots \overline{b}$

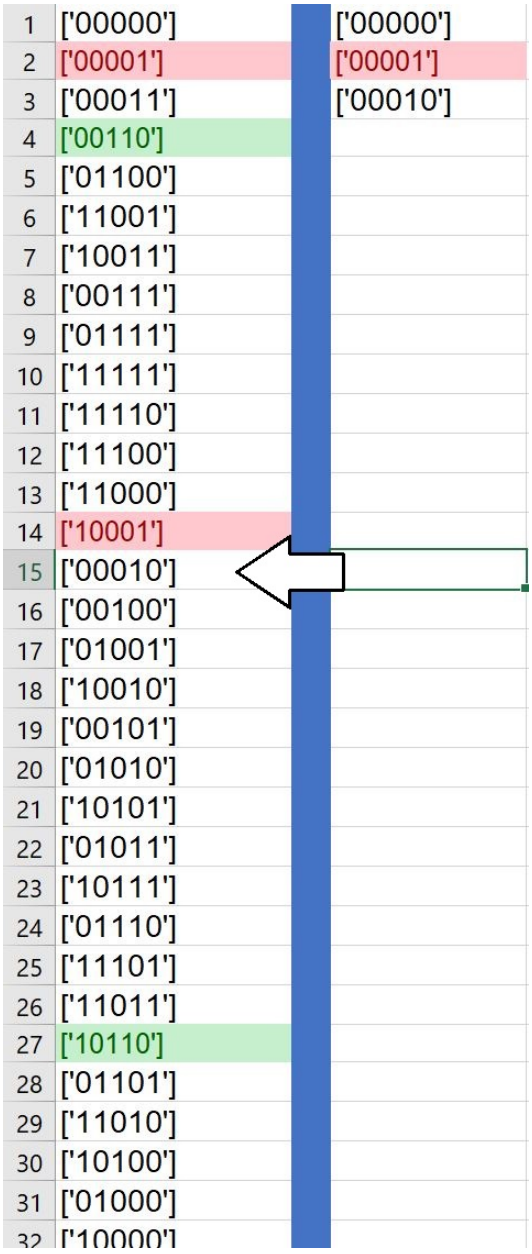


Εικόνα 4.6: Έξοδος του προγράμματος cross_join_pair.py για την ακολουθία «00000110011111000100101011101101»

Τα ζεύγη των συζυγών καταστάσεων όπως βλέπουμε και στην εικόνα 4.6 είναι τα ακόλουθα a(00001) b (00110) $\bar{a}$ (10001) και $\overline{b}$ (10110).

Στην συνέχεια εφόσον έχουμε εντοπίσει τις θέσεις των συζυγών καταστάσεων ξεκινούμε την δημιουργία της νέα ακολουθίας. Αρχίζουμε από την πρώτη κατάσταση της αρχικής ακολουθίας (αριστερή στήλη της εικόνας 4.7) και συνεχίζουμε την αντιστοίχιση με βάση την υπάρχουσα σειρά των καταστάσεων. Μόλις εντοπίσουμε την πρώτη κατάσταση από τα δύο συζυγή ζευγάρια (η οποία είναι η δεύτερη από την αριστερή στήλη της εικόνας 4.7) τότε αλλάζουμε το

δεξιότερο bit της επόμενης κατάστασης . Δηλαδή η τρίτη 5άδα της αρχικής ακολουθίας «00011» θα γίνει «00010» και θα τοποθετηθεί στην νέα ακολουθία (δεξιά στήλη).

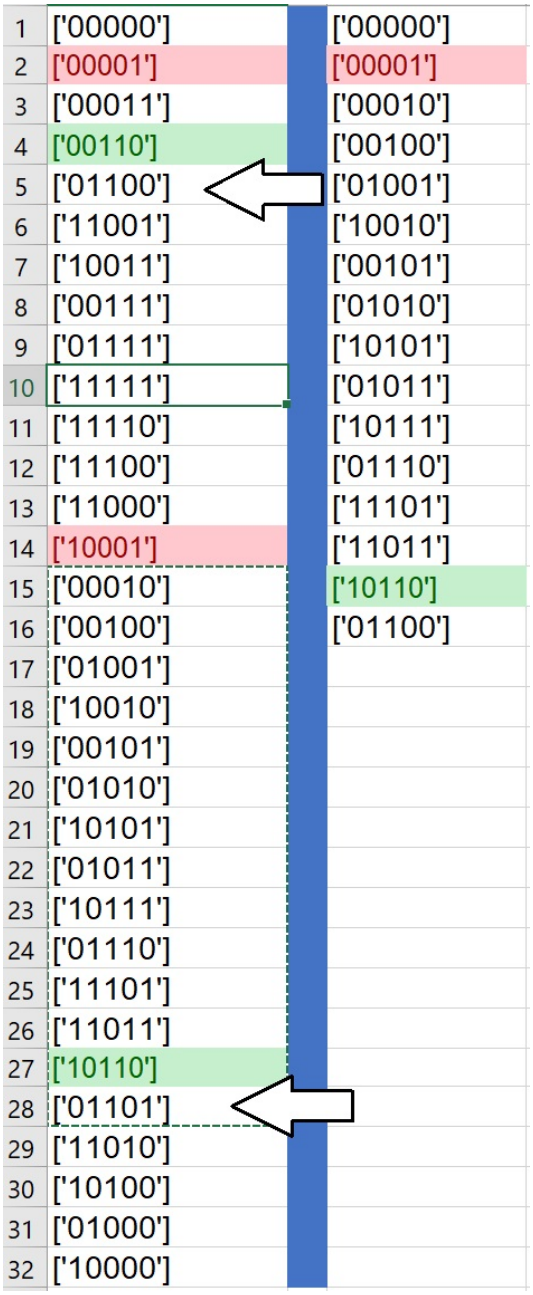


Εικόνα 4.7: Δημιουργία νέας de Bruijn – βήμα 1

Όπως βλέπουμε από την εικόνα 4.7, ξεκινούμε με την 5άδα που βρίσκεται στην θέση 1 και συνεχίζουμε με την θέση 2 όπου και έχουμε σημειώσει ότι βρίσκεται η μια 5άδα από τα 2 ζευγάρια 5άδων που έχουμε επιλέξει, έτσι στην συνέχεια θα τοποθετήσουμε την επόμενη κατάσταση αλλά με αλλαγή του δεξιότερου ψηφίου από 1 σε 0.

Στην συνέχεια θα αναζητήσουμε στον αρχικό μας κύκλο την κατάσταση «00010» και θα συνεχίσουμε από εκείνο το σημείο να τοποθετούμε τις επόμενες καταστάσεις. Έτσι σαν επόμενο

βήμα θα συνεχίσουμε από την σειρά 3 του δεξιού μέρους να τοποθετήσουμε όλες τις καταστάσεις του αριστερό μέρος που ξεκινούν από την θέση 15 μέχρι να βρούμε την επόμενη κατάσταση από τα ζευγάρια που επιλέξαμε. Στην συγκεκριμένη περίπτωση η επόμενη επιλεγμένη 5άδα βρίσκεται στη θέση 27 και άρα θα αλλάξουμε το δεξιότερο ψηφίο της επόμενης 5άδας, δηλαδή της θέσης 28 από «01101» σε «01100» και θα βρίσκεται στην θέση 16 της δεξιάς στήλης. Και έτσι θα συνεχίσουμε και θα συμπληρώσουμε όλες τις υπόλοιπες καταστάσεις.



Εικόνα 4.8: Δημιουργία νέας de Bruijn – βήμα 2

Αυτό θα έχει σαν αποτέλεσμα την συμπλήρωση όλων των καταστάσεων στον νέο κύκλο που δημιουργήθηκε (δεξιά στήλη).

1	['00000']		['00000']
2	['00001']		['00001']
3	['00011']		['00010']
4	['00110']		['00100']
5	['01100']		['01001']
6	['11001']		['10010']
7	['10011']		['00101']
8	['00111']		['01010']
9	['01111']		['10101']
10	['11111']		['01011']
11	['11110']		['10111']
12	['11100']		['01110']
13	['11000']		['11101']
14	['10001']		['11011']
15	['00010']		['10110']
16	['00100']		['01100']
17	['01001']		['11001']
18	['10010']		['10011']
19	['00101']		['00111']
20	['01010']		['01111']
21	['10101']		['11111']
22	['01011']		['11110']
23	['10111']		['11100']
24	['01110']		['11000']
25	['11101']		['10001']
26	['11011']		['00011']
27	['10110']		['00110']
28	['01101']		['01101']
29	['11010']		['11010']
30	['10100']		['10100']
31	['01000']		['01000']
32	['10000']		['10000']

Εικόνα 4.9: Δημιουργία νέας de Bruijn – βήμα 3

Η νέα ακολουθία είναι η «01001010111011001111100011010000» και η οποία προέκυψε από το δεξιότερο ψηφίο της κάθε 5άδας της δεξιά στήλης. Έτσι από τα παραπάνω μπορούμε να δούμε ότι αλλάζοντας μια de Bruijn ακολουθία σε 4 σημεία της, δημιουργήσαμε μια νέα de Bruijn ακολουθία με την τεχνική των cross join pairs.

Συνολικά με την μέθοδο κατασκευής **cross join pairs** χρησιμοποιήθηκαν 37 de Bruijn ακολουθίες τάξης n = 5, 6, 7, 8, 9.

Εδώ πρέπει να αναφέρουμε ότι όλα τα cross join pairs επιλέχθηκαν με τέτοιο τρόπο ώστε οι **νέες** ακολουθίες να μην έχουν τον ίδιο εκμηδενιστή (annihilator) με την αρχική ακολουθία έτσι ώστε

να εξετάσουμε εάν οι νέες de Bruijn θα έχουν την μέγιστη αλγεβρική ανθεκτικότητα. Αυτό πραγματοποιήθηκε με την κατάλληλη επιλογή των συζυγών ζευγαριών τέτοια ώστε όταν επιλέγαμε να αλλάξουμε την έξοδο πχ από «0» σε «1» σε μια από τις 4 συζυγές n-άδες θα έπρεπε η συγκεκριμένη n-άδα να δίνει έξοδο 1 στην συγκεκριμένη n-άδα (σειρά) του πίνακα αληθείας του εκμηδενιστή έτσι ώστε να είμαστε βέβαιοι ότι ο νέος εκμηδενιστής θα είναι διαφορετικός.

Στις 7 νέες de Bruijn (18,9%) ακολουθίες η αλγεβρική ανθεκτικότητα (algebraic immunity) βελτιώθηκε , στις 29 παρέμεινε το ίδιο (78,4%) και στη 1 μειώθηκε (2,7%).

4.3.2.2 Παραγωγή νέας de Bruijn ακολουθίας με μεθοδολογία cycle joining and splitting

Στην συγκεκριμένη μεθοδολογία έχουμε αρχικά 2 διαφορετικές ακολουθίες de Bruijn και τις ενώνουμε κατάλληλα έτσι ώστε να παραχθεί μια νέα ακολουθία που θα είναι και αυτή de Bruijn.

Για οποιαδήποτε de Bruijn ακολουθία (s_0 , s_1, … , s_p) όπου p είναι της μορφής $2^n - 1$ για κάποιον ακέραιο n, ορίζουμε την ακολουθία :

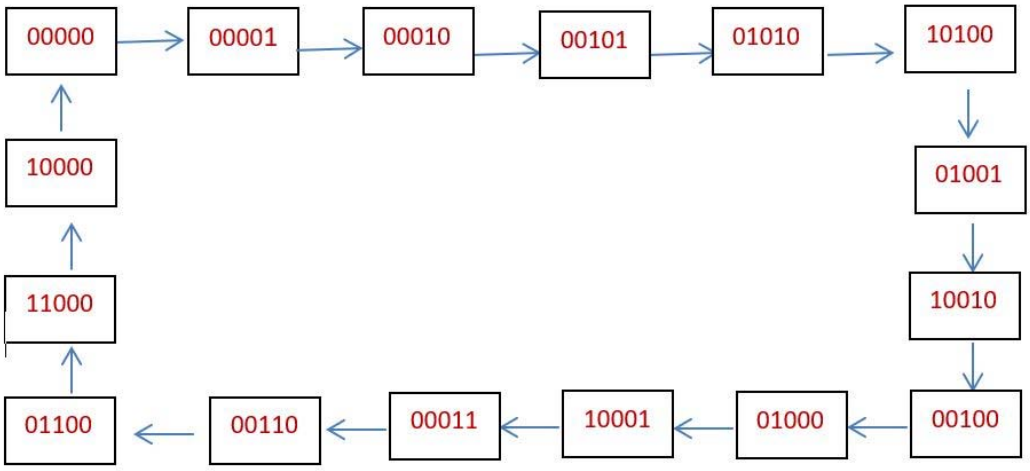
$D_0^{-1}(s) = (0, s_0 , s_0 \oplus s_1 , s_0 \oplus s_1 \oplus s_2 , \dots , s_0 \oplus s_1 \oplus s_2 \oplus \dots s_{p-1})$

Δηλαδή το πρώτο bit της ακολουθίας $D_0^{-1}(s)$ θα είναι το 0 , το δεύτερο bit θα είναι το s_0 , το τρίτο bit θα είναι το $s_0 \oplus s_1$ κλπ. . Το τελευταίο bit θα είναι το άθροισμα όλων εκτός του τελευταίου bit $s_0 \oplus s_1 \oplus s_2 \oplus \dots \oplus s_{p-1}$

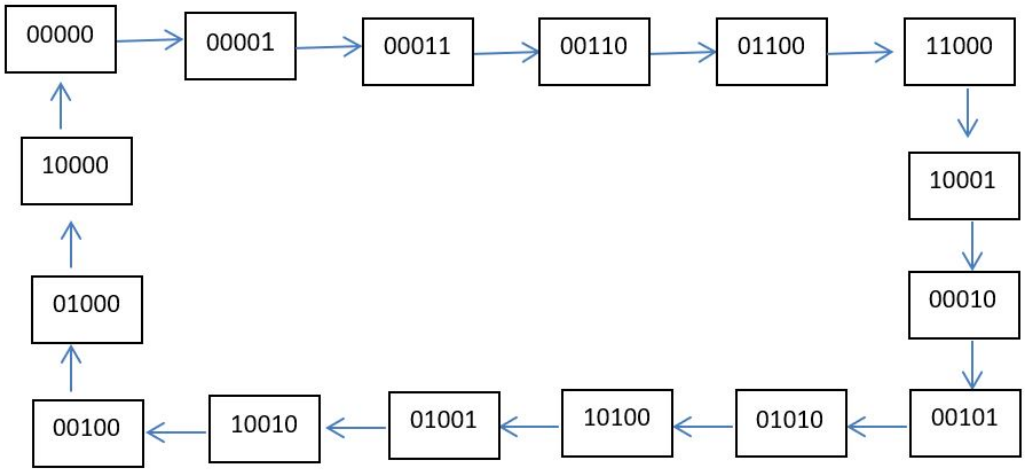
Επίσης θα ορίσουμε και την ακολουθία $D_1^{-1}(s)$ όπου θα είναι η συμπληρωματική της $D_0^{-1}(s)$. Πρέπει να αναφέρουμε πως οι δύο αυτές ακολουθίες που ορίσαμε <u>δεν</u> είναι de Bruijn. Για παράδειγμα έστω ότι έχουμε την de Bruijn ακολουθία s = «11010001» τότε έχουμε $D_0^{-1}(s)$ = «01001111» και $D_1^{-1}(s)$ = «10110000».

Για να κατασκευάσουμε λοιπόν την νέα ακολουθία de Bruijn θα πρέπει να αναζητήσουμε δύο de Bruijn ακολουθίες s και r τάξης n τέτοιες ώστε οι ακολουθίες **$D_0^{-1}(s)$** και **$D_1^{-1}(r)$** να περιέχουν ακριβώς τις ίδιες (n+1)άδες. Έτσι εάν βρούμε δύο τέτοιες ακολουθίες , συνδυάζοντας κατάλληλα τις $D_0^{-1}(s)$ και $D_1^{-1}(r)$ σε ένα συζυγές ζεύγος παράγουμε την νέα de Bruijn ακολουθία που αναζητούμε και θα είναι τάξης n+1.

Για καλύτερη κατανόηση ακολουθεί παράδειγμα με δύο de Bruijn ακολουθίες s = 0000111101100101 και r = 0000101001111011 τάξης n = 4 και οι δύο. Για κάθε μια εκ των $D_0^{-1}(r)$ = 0000010100100011 και $D_0^{-1}(s)$ = 0000011000101001 εξετάζουμε τις (n+1)άδες τους , δηλαδή τις 5άδες τους κατασκευάζοντας τον κατάλληλο γράφο όπως βλέπουμε και στις εικόνες 4.10 και 4.11 παρακάτω. Θα πρέπει και οι δύο γράφοι να περιέχουν τις ίδιες 5άδες για να μπορέσουμε να συνενώσουμε κατάλληλα τις δύο de Bruijn.

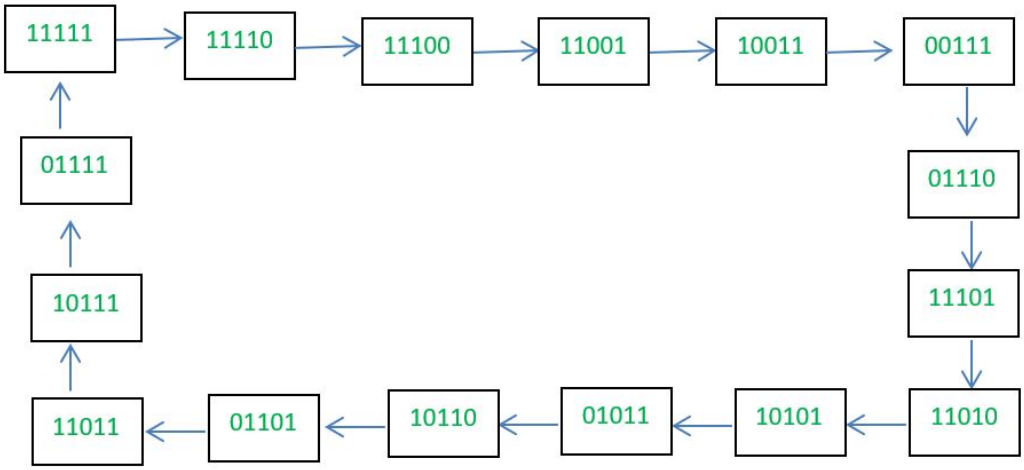


Εικόνα 4.10: Γράφος $D_0^{-1}(r)$



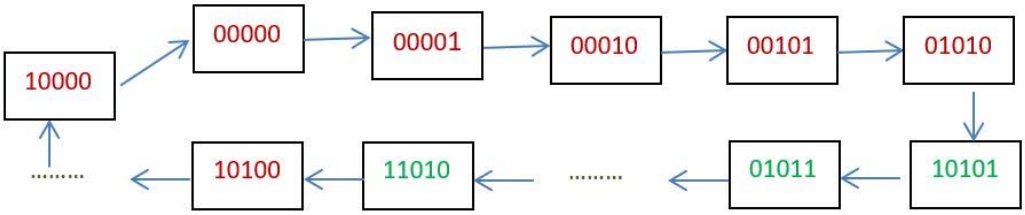
Εικόνα 4.11: Γράφος της $D_0^{-1}(s)$

Μπορούμε να διαπιστώσουμε ότι οι ίδιες 16 5άδες εμφανίζονται και στους δύο γράφους των $D_0^{-1}(r)$ και $D_0^{-1}(s)$. Επίσης μπορούμε να συμπεράνουμε πως η συμπληρωματική $D_1^{-1}(s)$ = 1111100111010110 περιέχει τις υπόλοιπες 16 5άδες που δεν εμφανίζονται στους παραπάνω γράφους. Στη εικόνα 4.12 βλέπουμε τον γράφο της $D_1^{-1}(s)$



Εικόνα 4.12: Γράφος της $D_1^{-1}(s)$

Αναζητούμε ζευγάρι συζυγών 5άδων που η μια εξ αυτών να βρίσκεται στην $D_0^{-1}(r)$ και η άλλη στην $D_1^{-1}(s)$. Τέτοιο ζευγάρι είναι το **01010** (βρίσκεται στην $\mathbf{D_0^{-1}(r)}$) και το **11010** (βρίσκεται στην $\mathbf{D_1^{-1}(s)}$). Τα ενώνουμε εναλλάσσοντας , για κάθε μια από αυτές τις 5άδες την επόμενη της , όπως φαίνεται και στην εικόνα 4.13 .



Εικόνα 4.13: Νέος γράφος από ένωση των δύο de Bruijn

Οπότε παρεμβάλαμε την $D_1^{-1}(s)$ (πράσινα) εντός της $D_0^{-1}(r)$ (κόκκινα) δημιουργώντας έναν νέο γράφο με 32 διαφορετικές καταστάσεις και έτσι ουσιαστικά δημιουργήσαμε την νέα de Bruijn ακολουθία 00000101011011111001110100100011.

Για να μπορέσουμε να κατασκευάσουμε τις νέες de Bruijn με την μεθοδολογία που αναλύσαμε παραπάνω αναπτύξαμε κώδικα σε Python. Το πρόγραμμα το ονομάσαμε **joining_splitting.py** και μπορούμε να βρούμε στο παράρτημα Α4. Ο συγκεκριμένος κώδικας για 2 υπάρχουσες ακολουθίες de Bruijn s, r τάξης n υπολογίζει όλες τις (n+1)-άδες για την κάθε μία εκ των $D_0^{-1}(r)$, $D_1^{-1}(r)$, $D_0^{-1}(s)$, $D_1^{-1}(s)$, καθώς επίσης ελέγχει εάν οι $\mathbf{D_0^{-1}(s)}$ και $\mathbf{D_1^{-1}(r)}$ περιέχουν τις ίδιες (n+1)-άδες και ταυτόχρονα βρίσκει όλα τα συζυγή ζευγάρια που η μια εξ αυτών βρίσκεται στην $D_0^{-1}(r)$ και η άλλη στην $D_1^{-1}(s)$.

```
C:\Users\johnk\OneDrive\Desktop\Python - C Program>python joining_spliting.py
Ayth einai h D0(s):  [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1]
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 1, 1], [0, 0, 0, 1, 1, 0], [0, 0, 1, 1, 0, 0], [0, 1, 1, 0, 0, 1], [1, 1, 0, 0, 1, 0], [1, 0, 0, 1, 0, 1], [0, 0, 1, 0, 1, 0], [0, 1, 0,
 1, 0, 1], [1, 0, 1, 0, 1, 1], [0, 1, 0, 1, 1, 0], [1, 0, 1, 1, 0, 1], [0, 1, 1, 0, 1, 1], [1, 1, 0, 1, 1, 1], [1, 0, 1, 1, 1, 1], [0, 1, 1, 1, 1, 0], [1, 1, 1, 1, 0, 1], [1, 1, 1, 0, 1, 1],
 [1, 1, 0, 1, 1, 0], [1, 0, 1, 1, 0, 0], [0, 1, 1, 0, 0, 0], [1, 1, 0, 0, 0, 1], [1, 0, 0, 0, 1, 0], [0, 0, 0, 1, 0, 1], [0, 0, 1, 0, 1, 1], [0, 1, 0, 1, 1, 1], [1, 0, 1, 1, 1, 0], [0, 1, 1,
 1, 0, 0], [1, 1, 1, 0, 0, 0], [1, 1, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0]]
--------------
Ayth einai h D1(s): [1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0]
[[1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 0], [1, 1, 1, 1, 0, 0], [1, 1, 1, 0, 0, 1], [1, 1, 0, 0, 1, 1], [1, 0, 0, 1, 1, 0], [0, 0, 1, 1, 0, 1], [0, 1, 1, 0, 1, 0], [1, 1, 0, 1, 0, 1], [1, 0, 1,
 0, 1, 0], [0, 1, 0, 1, 0, 0], [1, 0, 1, 0, 0, 1], [0, 1, 0, 0, 1, 0], [1, 0, 0, 1, 0, 0], [0, 0, 1, 0, 0, 0], [0, 1, 0, 0, 0, 0], [1, 0, 0, 0, 0, 1], [0, 0, 0, 0, 1, 0], [0, 0, 0, 1, 0, 0],
 [0, 0, 1, 0, 0, 1], [0, 1, 0, 0, 1, 1], [1, 0, 0, 1, 1, 1], [0, 0, 1, 1, 1, 0], [0, 1, 1, 1, 0, 1], [1, 1, 1, 0, 1, 0], [1, 1, 0, 1, 0, 0], [1, 0, 1, 0, 0, 0], [0, 1, 0, 0, 0, 1], [1, 0, 0,
 0, 1, 1], [0, 0, 0, 1, 1, 1], [0, 0, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1]]


Ayth einai h D0(r):  [0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0]
[[0, 0, 1, 1, 0, 0], [0, 1, 1, 0, 0, 0], [1, 1, 0, 0, 0, 1], [1, 0, 0, 0, 1, 0], [0, 0, 0, 1, 0, 1], [0, 0, 1, 0, 1, 1], [0, 1, 0, 1, 1, 0], [1, 0, 1, 1, 0, 1], [0, 1, 1, 0, 1, 1], [1, 1, 0,
 1, 1, 1], [1, 0, 1, 1, 1, 1], [0, 1, 1, 1, 1, 0], [1, 1, 1, 1, 0, 1], [1, 1, 1, 0, 1, 1], [1, 1, 0, 1, 1, 0], [1, 0, 1, 1, 0, 0], [0, 1, 1, 0, 0, 1], [1, 1, 0, 0, 1, 0], [1, 0, 0, 1, 0, 1],
 [0, 0, 1, 0, 1, 0], [0, 1, 0, 1, 0, 1], [1, 0, 1, 0, 1, 1], [0, 1, 0, 1, 1, 1], [1, 0, 1, 1, 1, 0], [0, 1, 1, 1, 0, 0], [1, 1, 1, 0, 0, 0], [1, 1, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0], [0, 0, 0,
 0, 0, 0], [0, 0, 0, 0, 0, 1], [0, 0, 0, 0, 1, 1], [0, 0, 0, 1, 1, 0]]
--------------
Ayth einai h D1(r): [1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1]
[[1, 1, 0, 0, 1, 1], [1, 0, 0, 1, 1, 1], [0, 0, 1, 1, 1, 0], [0, 1, 1, 1, 0, 1], [1, 1, 1, 0, 1, 0], [1, 1, 0, 1, 0, 0], [1, 0, 1, 0, 0, 1], [0, 1, 0, 0, 1, 0], [1, 0, 0, 1, 0, 0], [0, 0, 1,
 0, 0, 0], [0, 1, 0, 0, 0, 0], [1, 0, 0, 0, 0, 1], [0, 0, 0, 0, 1, 0], [0, 0, 0, 1, 0, 0], [0, 0, 1, 0, 0, 1], [0, 1, 0, 0, 1, 1], [1, 0, 0, 1, 1, 0], [0, 0, 1, 1, 0, 1], [0, 1, 1, 0, 1, 0],
 [1, 1, 0, 1, 0, 1], [1, 0, 1, 0, 1, 0], [0, 1, 0, 1, 0, 0], [1, 0, 1, 0, 0, 0], [0, 1, 0, 0, 0, 1], [1, 0, 0, 0, 1, 1], [0, 0, 0, 1, 1, 1], [0, 0, 1, 1, 1, 1], [0, 1, 1, 1, 1, 1], [1, 1, 1,
 1, 1, 1], [1, 1, 1, 1, 1, 0], [1, 1, 1, 1, 0, 0], [1, 1, 1, 0, 0, 1]]


Aytes einai oi diafores:  []
Aytes einai oi idies anamesa sthn symplhrwmatikh ths D0(s) kai sthn D1(r):  [[1, 0, 0, 0, 0, 1], [1, 0, 0, 0, 1, 1], [1, 0, 0, 1, 1, 0], [1, 1, 1, 0, 0, 1], [0, 1, 0, 0, 1, 0], [1, 0, 1, 0,
1, 0], [1, 1, 0, 1, 0, 1], [0, 0, 1, 1, 0, 1], [0, 0, 1, 1, 1, 1], [1, 1, 1, 1, 1, 0], [0, 1, 1, 1, 0, 1], [0, 1, 0, 0, 0, 1], [0, 0, 0, 0, 1, 0], [0, 0, 1, 1, 1, 0], [1, 1, 1, 1, 0, 0], [0,
 1, 0, 0, 0, 0]]

C:\Users\johnk\OneDrive\Desktop\Python - C Program>
```

Εικόνα 4.14: Έξοδος του προγράμματος joining_splitting.py

Συνολικά με την μέθοδο κατασκευής **cycle joining & splitting** χρησιμοποιήθηκαν 19 ζευγάρια ακολουθιών de Bruijn για να κατασκευαστούν οι νέες ακολουθίες. ‘Ολα τα αρχικά ζευγάρια περιείχαν ακολουθίες de Bruijn ίδιας τάξης. Από τα αρχικά 19 ζευγάρια τα 7 ήταν de Bruijn τάξης n = 4 , τα 4 τάξης n = 5 και από 2 τάξης n = 6 , 7, 8 , 9.

Η δυσκολία στην μέθοδο αυτή ήταν η εύρεση κατάλληλων ζευγών de Bruijn τάξης n οι οποίες περιείχαν ζευγάρι συζυγών n-άδων έτσι ώστε να δημιουργούνται οι κατάλληλές $D_0^{-1}(s)$ και $D_1^{1}(r)$ για να παραχθεί η νέα de Bruijn n+1 τάξης.

Συνολικά σε όλες τις νέες de Bruijn ακολουθίες που δημιουργήθηκαν από τα επιλεγμένα ζευγάρια, η αλγεβρική κανονική μορφή διατήρησε το μέγιστο δυνατό βαθμό n-1. Επίσης όλες οι ακολουθίες περιείχαν τουλάχιστον μια γραμμική δομή και η ανθεκτικότητα στις συσχετίσεις ήταν μηδενική.

Από τα 15 ζευγάρια de Bruijn ακολουθιών τάξης n που είχαν την ίδια αρχική μη γραμμικότητα (non linearity) και οι 15 (100%) νέες ακολουθίες de Bruijn τάξης n+1 έχουν μη γραμμικότητα (non linearity) που είναι ίση ή μεγαλύτερη της αρχικής διπλάσιας τιμής.

Από τα 4 ζευγάρια de Bruijn ακολουθιών τάξης n που είχαν διαφορετική μη γραμμικότητα (non linearity) οι τιμές όλων των νέων de Bruijn ακολουθιών τάξης n+1 έχουν βαθμό μη γραμμικότητας μικρότερο από τη διπλάσια τιμή της μεγαλύτερης αρχικής τιμής μη γραμμικότητας (non linearity). Ωστόσο, όλες οι νέες de Bruijn έχουν μεγαλύτερη μη γραμμικότητα από τη διπλάσια τιμή της μικρότερης αρχικής τιμής τη μη γραμμικότητας. Με άλλα λόγια, αν L_1, L_2 είναι οι μη γραμμικότητες των συναρτήσεων των αρχικών De Bruijn

ακολουθιών τάξης n, όπου $L_1<L_2$, και L είναι η μη γραμμικότητα της συνάρτησης της νέας De Bruijn ακολουθίας τάξης n+1, τότε τα πειράματά μας σε όλες τις περιπτώσεις καταδεικνύουν ότι ισχύει $2L_1 < L < 2L_2$.

Τα 11 από τα 19 αρχικά ζευγάρια ακολουθιών de Bruijn τάξης n, έχουν μέγιστη αλγεβρική ανθεκτικότητα (algebraic immunity). Οι 5 (45,5%) νέες de Bruijn ακολουθίες τάξης n+1 που κατασκευάστηκαν με την μεθοδολογία cycle joining & splitting διατηρούν την μέγιστη αλγεβρική ανθεκτικότητα (algebraic immunity) σε σύγκριση με τις αρχικές 11.

Επίσης τα 8 από τα 19 ζευγάρια δεν έχουν αρχικά τη μέγιστη αλγεβρική ανθεκτικότητα (algebraic immunity). Όμως οι 7 από τις 8 (87.5%) νέες ακολουθίες που παράγονται έχουν την μέγιστη αλγεβρική ανθεκτικότητα (algebraic immunity).

Συνολικά οι 11 από τις 19 (57,9%) νέες de Bruijn ακολουθίες τάξης n+1 έχουν την μέγιστη αλγεβρική ανθεκτικότητα (algebraic immunity).

4.4 Συνολική αποτίμηση κρυπτογραφικών χαρακτηριστικών των μεθοδολογιών

Έχοντας υπολογίσει τις νέες de Bruijn ακολουθίες που παρήχθησαν από υπάρχουσες de Bruijn με βάση τον τρόπο κατασκευής των cross join pairs και cycle joining & splitting που είδαμε στις παραγράφους 3.7.1 και 3.7.2 , αποτιμήσαμε τα κρυπτογραφικά χαρακτηριστικά (non linearity, algebraic immunity, correlation immunity και linear structure) των λογικών συναρτήσεων που τις παράγουν ακολουθώντας τα ίδια βήματα όπως αναφέρθηκαν στην παράγραφο 4.3.2.

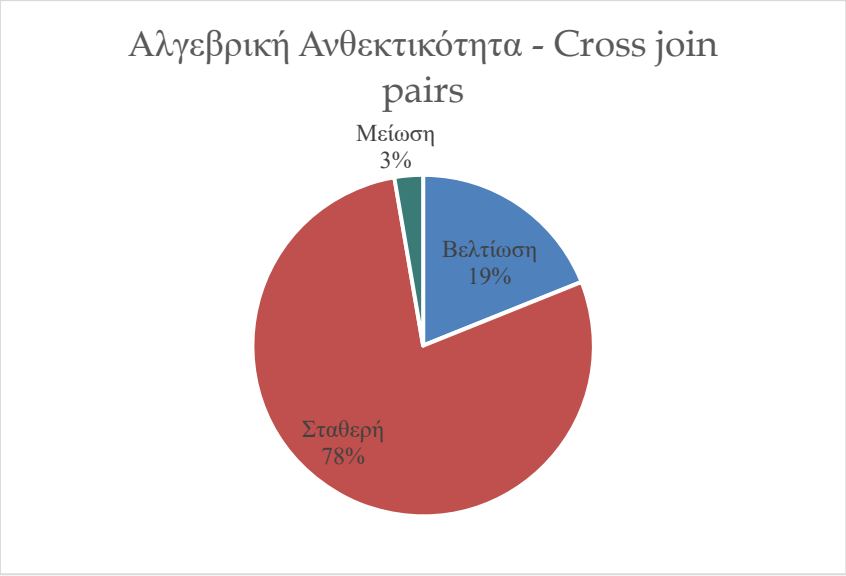
Από τα παραπάνω συμπεραίνουμε πως με την μέθοδο του **cycle joining & splitting** ότι η μη γραμμικότητα (non linearity) των νέων ακολουθιών φαίνεται ότι λαμβάνει τιμές μεταξύ της διπλάσιας τιμής της μικρότερης αρχικής και της διπλάσιας της αρχικής ακολουθίας με την μεγαλύτερη μη γραμμικότητα (non linearity). Έστω ότι L_1, L_2 είναι οι μη γραμμικότητες των συναρτήσεων των αρχικών De Bruijn ακολουθιών τάξης n, όπου $L_1 < L_2$, και L είναι η μη γραμμικότητα της συνάρτησης της νέας De Bruijn ακολουθίας τάξης n+1, τότε τα πειράματά μας σε όλες τις περιπτώσεις καταδεικνύουν ότι ισχύει $2L_1 < L < 2L_2$.

Επίσης μπορούμε να συμπεράνουμε – βάσει των πειραμάτων - ότι όταν οι 2 αρχικές ακολουθίες έχουν ίδιο βαθμό μη γραμμικότητας (non linearity) τότε η νέα ακολουθία θα έχει βαθμό μεγαλύτερο του διπλάσιου των αρχικών.

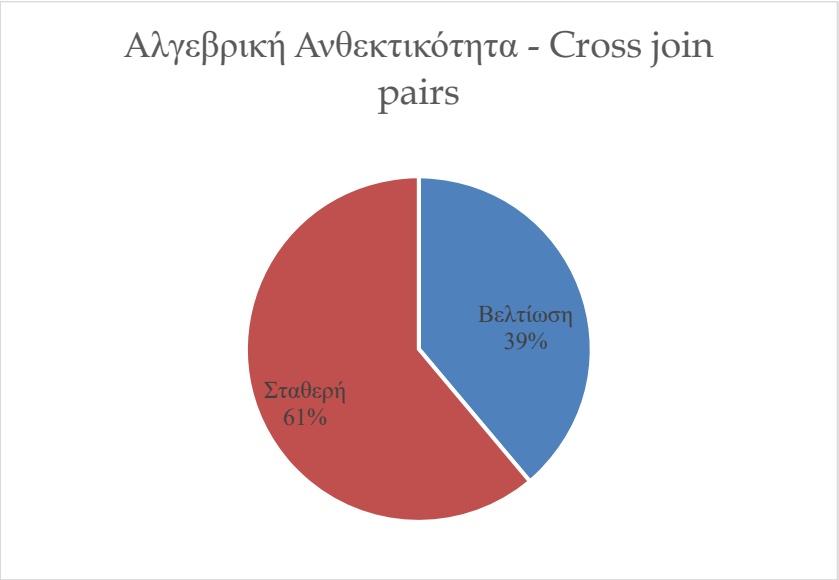
Η πιθανότητα της αλγεβρικής ανθεκτικότητας (algebraic immunity) της νέας de Bruijn ακολουθίας να διατηρηθεί στο μέγιστο βαθμό, από ζευγάρι ακολουθιών οι οποίες και οι δύο δεν είχαν τον μέγιστο βαθμό της αλγεβρικής ανθεκτικότητας με την προαναφερθείσα μεθοδολογία είναι πολύ μεγάλη, και είναι στο 87,5% .

Με την μεθοδολογία των **cross join pairs** γενικά η συντριπτική πλειοψηφία της αλγεβρικής ανθεκτικότητας των νέων ακολουθιών έμεινε αμετάβλητη 78,4% με μια πιθανότητα βελτίωσης 19% . Ωστόσο εάν λάβουμε μόνο υπόψιν τις ακολουθίες που αρχικά δεν είχαν την μέγιστη αλγεβρική ανθεκτικότητα και στην συνέχεια την βελτίωσαν αποκτώντας την μέγιστη δυνατή βλέπουμε ότι το ποσοστό βελτίωσης ανέρχεται στο 39%.

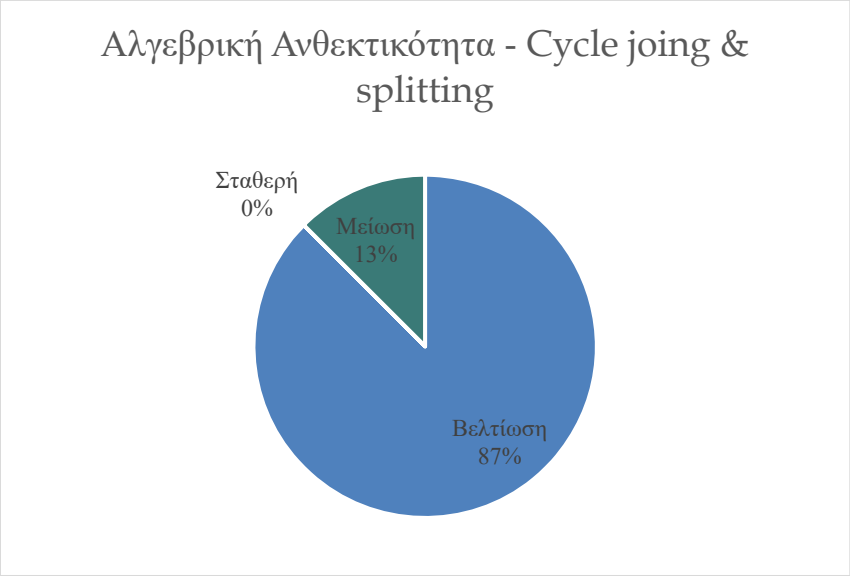
Τα παραπάνω μπορούμε να τα οπτικοποιήσουμε στα παρακάτω 3 γραφήματα:



Γράφημα 4.15: Συνολικά ποσοστά αλγεβρικής ανθεκτικότητας με την μέθοδος των cross join pair



Γράφημα 4.16: Ποσοστά αλγεβρικής ανθεκτικότητας με την μέθοδος των **cross join pairs** όταν οι αρχικές ακολουθίες δεν έχουν την μέγιστη αλγεβρική ανθεκτικότητα



Γράφημα 4.17: Ποσοστά αλγεβρικής ανθεκτικότητας με την μέθοδος των **cycle joining & splitting** όταν τα αρχικά ζευγάρια δεν έχουν την μέγιστη αλγεβρική ανθεκτικότητα

Ανεξαρτήτως της μεθοδολογίας που ακολουθήσαμε για την κατασκευή των de Bruijn ακολουθιών, είδαμε πως η ανθεκτικότητα στις συσχετίσεις των παραγόμενων ακολουθιών είναι μηδενικής τάξης, όλες οι ακολουθίες διατηρούν τουλάχιστον μια γραμμική δομή και η αλγεβρική κανονική μορφή τους διατηρεί τον μέγιστο δυνατό βαθμό n-1.

Κεφάλαιο 5
Επίλογος

5.1 Σύνοψη

Η παρούσα διατριβή εστίασε στο ανοιχτό ερευνητικό πρόβλημα της αποτίμησης κρυπτογραφικών ιδιοτήτων λογικών συναρτήσεων οι οποίες, όταν τεθούν ως συναρτήσεις ανάδρασης σε NLFSRs, παράγουν ακολουθίες De Bruijn. Το ζήτημα αυτό παρουσιάζει εξαιρετικό ενδιαφέρον στην κρυπτογραφία, μια που οι NLFSR αποτελούν βασικό δομικό συστατικό πολλών κρυπταλγορίθμων ροής, οι οποίοι προτιμώνται σε εφαρμογές με χαμηλές απαιτήσεις σε υπολογιστικούς πόρους. Χαρακτηριστικές τέτοιες εφαρμογές ανακύπτουν λόγω της ραγδαίας αύξησης των IoT συσκευών, δηλαδή «έξυπνων» συσκευών που διασυνδέονται μέσω του διαδικτύου και ανταλλάσσουν δεδομένα, δημιουργώντας την ανάγκη για κρυπτογράφηση όλων αυτών των δεδομένων με χρήση «ελαφρών» (lightweight) αλγορίθμων.

Βασικό συστατικό όλων αυτών των αλγορίθμων είναι η χρησιμοποίηση ψευδοτυχαίων ακολουθιών που να προσομοιάζουν όσο το δυνατό της τυχαίες ακολουθίες. Και αυτό γιατί η

ασφάλεια των αλγορίθμων ροής στηρίζεται στην παραγόμενη, από τις αντίστοιχες γεννήτριες κλειδοροή (keystream), δηλαδή στην ακολουθία που παράγεται έτσι ώστε να γίνει πρόσθεση XOR με τα δεδομένα που πρέπει να κρυπτογραφηθούν. Έτσι, βασικός σχεδιαστικός στόχος στους αλγορίθμους ροής είναι η παραγόμενη ακολουθία της κλειδοροής να μην μπορεί να προβλεφθεί. Για να γίνει αυτό θα πρέπει να εκπληρώνει όσο το δυνατόν καλύτερα διάφορα κριτήρια τυχαιότητας όπως μεγάλη περιοδικότητα της ακολουθίας, υψηλός βαθμός γραμμικής πολυπλοκότητας, καθώς επίσης να διαθέτει διάφορα άλλα στατιστικά χαρακτηριστικά.

Για μία γεννήτρια κλειδοροής είναι κρυπτογραφικά επιθυμητό να παράγει ακολουθίες με τη μέγιστη δυνατή περίοδο. Όταν η γεννήτρια βασίζεται σε μη γραμμικό καταχωρητή ολίσθησης με ανάδραση (NLFSR), η ακολουθία μεγίστης περιόδου που παράγεται ονομάζεται ακολουθία De Bruijn. Οι ιδιότητες των ακολουθιών De Bruijn έχουν μελετηθεί μεμονωμένα, αλλά υπάρχουν ακόμα πολλά ερωτήματα ως προς τις ιδιότητες των NLFSR που παράγουν ακολουθίες De Bruijn.

Στο πλαίσιο της διατριβής αφού παρουσιάσαμε τις ιδιότητες των de Bruijn ακολουθιών, αξιοποιήσαμε την μέθοδο υπαναχώρησης (backtracking) για την κατασκευή των de Bruijn ακολουθιών διαφορετικών τάξεων, με απώτερο στόχο να τις αντιστοιχίσουμε με λογικές συναρτήσεις: τις συναρτήσεις αυτές ακολούθως τις μελετήσαμε ως προς τα κρυπτογραφικά χαρακτηριστικά τους με χρήση αφενός του λογισμικού ανοικτού κώδικα SageMath και, αφετέρου, με χρήση κατάλληλων προγραμμάτων σε γλώσσα python που αναπτύξαμε έτσι ώστε να αυτοματοποιήσουμε και να διεκπεραιώσουμε τις διαδικασίες των πειραμάτων.

Στην συνέχεια υλοποιήσαμε 2 τρόπους κατασκευής de Bruijn ακολουθιών που κάνουν χρήση υπαρχουσών de Bruijn ακολουθιών με τελικό στόχο να μελετήσουμε τα κρυπτογραφικά χαρακτηριστικά των συναρτήσεων των νέων De Bruijn ακολουθιών, σε σχέση με τα χαρακτηριστικά των συναρτήσεων των αρχικών De Bruijn ακολουθιών.

5.2 Συμπεράσματα – Μελλοντική έρευνα

Από τα πειράματά μας, τα οποία πραγματοποιήθηκαν επί τυχαίων ακολουθιών De Bruijn τάξης n=5, 6, 7, 8 , 9 και αξιοποιώντας τις τεχνικές «cross join pairs» (παραγωγή μίας ακολουθίας De Bruijn από μία άλλη ίδιας τάξης) και «cycle joining & splitting» (παραγωγή μίας ακολουθίας De Bruijn από δύο άλλες χαμηλότερης τάξης), προκύπτουν τα εξής:

1. Δεν βρέθηκε γεννήτρια ακολουθίας De Bruijn η οποία να έχει κάποια ανθεκτικότητα σε συσχετίσεις (correlation immunity)

2. Δεν βρέθηκε γεννήτρια ακολουθίας De Bruijn η οποία να μην έχει καμία γραμμική δομή (Linear structure)

3. Σε περίπτωση που μία γεννήτρια ακολουθίας de Bruijn δεν έχει μέγιστη αλγεβρική ανθεκτικότητα (algebraic immunity), και γνωρίζουμε έναν εκμηδενιστή (annihilator) g αυτής χαμηλού βαθμού, τότε εφαρμόζοντας κατάλληλα την τεχνική «cross join pairs» έτσι ώστε η συνάρτηση g να μην είναι εκμηδενιστής της νέας γεννήτριας De Bruijn, η πιθανότητα η νέα αυτή γεννήτρια να έχει μέγιστη αλγεβρική ανθεκτικότητα δεν είναι αρκετά υψηλή .

4. Η τεχνική «cycle joining & splitting» φαίνεται ότι είναι μία πολύ καλή προσέγγιση για παραγωγή γεννητριών ακολουθιών De Bruijn με υψηλή μη γραμμικότητα: συγκεκριμένα, αν οι γεννήτριες των αρχικών De Bruijn ακολουθιών τάξης n έχουν μη γραμμικότητα L_1 και L_2 αντίστοιχα, με $L_1 \leq L_2$, τότε η γεννήτρια της νέας De Bruijn ακολουθίας τάξης n+1 έχει μη γραμμικότητα μεγαλύτερη από $2L_1$ (οπότε, ξεκινώντας με υψηλές τιμές μη γραμμικότητας, φαίνεται ότι διασφαλίζονται υψηλές τιμές μη γραμμικότητας και για De Bruijn ακολουθίες μεγαλύτερης τάξης).

Ως μελλοντική έρευνα, θα πρέπει είτε να υπάρξει μία μαθηματική θεμελίωση των ανωτέρω είτε να προσδιοριστούν μαθηματικά οι συνθήκες εκείνες οι οποίες, εφόσον συντρέχουν, τότε τα ανωτέρω εγγυημένα ικανοποιούνται για οποιαδήποτε ακολουθία De Bruijn. Απώτερος ερευνητικός στόχος είναι η μεθοδολογική κατασκευή συναρτήσεων που εγγυημένα παράγουν ακολουθίες De Bruijn και, ταυτόχρονα, ικανοποιούν σύνολο κρυπτογραφικών ιδιοτήτων.

Βιβλιογραφία

[01] S. S. Bedi, N. Rajesh Pillai. «Cubes attacks on Trivium», Scientific Analysis Group, Metcalfe House Complex, 2009.

[02] N.G. de Bruijn. «A Combinatorial Problem». Proc. Koninklijke Nederlandse Akademie v. Wetenschappen, vol 49, pp 758 – 764, 1946.

[03] P. Camion, C. Carlet, P. Charpin, N. Sendrier. «On correlation-immune functions. In: Advances in cryptology: Crypto '91». Lecture notes in computer science, Springer, vol 576, pp 86 – 100, 1991.

[04] A.H. Chan, R.A. Games. «On the Complexities of de Bruijn Sequences». Journal of Combinatorial Theory, vol 33 issue 3, pp 233 – 246, 1982.

[05] S. Claude. «Communication Theory of Secrecy Systems». Bell System Technical Journal, vol 28, pp 662, 1949.

[06] N. Courtois. «Fast algebraic attacks on stream ciphers with linear feedback». Proceedings of CRYPTO 2003, Lecture notes in computer science, vol 2729, pp 177–194, 2003.

[07] E. Dawson, C. Wu. «On the linear structure of symmetric Boolean functions». Australian Journal of Combinatorics, vol 16, pp 239 – 243, 1997.

[08] H. Fredricksen. «A survey of full length nonlinear shift register cycle algorithms». Society for Industrial and Applied Mathematics, vol 24 issue 2, pp 195 – 221, 1982.

[09] B. Francis. «Of the Advancement and Proficience of Learning». Oxford University, pp 257–271, 1640.

[10] J.D. Golić. «On the security of nonlinear filter generators». Fast Software Encryption, Lecture notes in computer science, Springer, vol. 1039, pp 173–188, 1996.

[11] S.W.Golomb,. «On the Classification of Balanced Binary Sequences of Period 2n−1». IEEE Transformation on Information Theory, issue 6, vol 26, pp 730 – 732, 1980.

[12] S.W. Golomb. «Shift Register Sequences». Aegean Park Press, 1981.

[13] T. Helleseth. «De Bruijn sequence». Encyclopedia of Cryptography and Security, Springer, pp 138 – 140, 2005.

[14] T. Helleseth, T. Klove. «The number of cross-join pairs in maximum length linear sequences». IEEE Transactions on Information Theory, vol. 37, no. 6, pp. 1731 - 1733, 1991.

[15] A. Kerckhoffs. «La cryptographie militaire». Journal des sciences militaires, vol 9, pp 161–191, 1883.

[16] D. Knuth. «The art of computer programming». Addison-Wesley Longman Publishing Co., vol. 2, 1997.

[17] Lee S, Chee S, Park S. «Conditional correlation attack on nonlinear filter generators». Advances in cryptology – ASIACRYPT'96. Lecture notes in computer science, Springer, vol 1163, pp 360–367, 1996.

[18] A. Lempel. «On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers». IEEE Transactions on Computers, vol 19, pp 1204 – 1209, 1970.

[19] K. Limniotis, N. Kolokotronis and D. Kotanidis, «De Bruijn sequences and suffix arrays: Analysis and constructions», chapter in Modern Discrete Mathematics and Analysis With Applications in Cryptography, Information Systems and Modeling, Springer, 2018 (to appear).

[20] J.L. Massey, «Shift-register synthesis and BCH decoding». IEEE Transactions on Information Theory, vol 15, pp 122-127, 1969.

[21] G. Marsaglia. «A current view of random number generators». Statistics and Computer Science: XVI Symposium on the Interface, Elsevier Science Publishers, pp 3- 10, 1985.

[22] A. Menezes, P.V. Oorschot, S. Vanstone, «Handbook of Applied Cryptography». CRC Press, 1996.

[23] J. Mykkeltveit, J. Szmidt. «On cross joining de Bruijn sequences». Contemporary Mathematics, AMS, vol 632, pp 333 - 344, 2015.

[24] L. O'Connor, A. Klapper. «Algebraic nonlinearity and its applications to cryptography». Journal of cryptology, vol 7, pp 213 – 228, 1994.

[25] T. Siegenthaler, «Cryptanalysts representation of nonlinearly filtered m-sequences». Advances in Cryptology-Eurocrypt. '85, vol. 219, pp. 103-110, 1986.

[26] A. Ralston. «De Bruijn sequences-a model example of the interaction of discrete mathematics and computer science». Mathematics Magazine, vol 55, pp 131–143, 1982.

[27] https://plato.stanford.edu/entries/della-porta/

[28] https://en.wikipedia.org/wiki/Enigma_machine

[29] https://en.wikipedia.org/wiki/ESTREAM

[30] http://www.crypto-it.net/eng/theory/kerckhoffs.html

[31] https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf

[32] http://www.sagemath.org/

[33] https://en.wikipedia.org/wiki/Trivium_(cipher)

[34] https://en.wikipedia.org/wiki/The_Gold-Bug

[35] http://www.crypto-it.net/eng/simple/trithemius-cipher.html?tab=0

[36] http://www.cryptomuseum.com/crypto/vernam.htm

[37] http://webpages.uncc.edu/yonwang/liltest/

[38] https://www.archives.gov/education/lessons/zimmermann

[39] https://learncryptography.com/classical-encryption/caesar-cipher

[40] https://www.cs.uky.edu/~klapper/pdf/nonlinearity.pdf

Παράρτημα Α
Λογισμικό που χρησιμοποιήθηκε

Α.1 Κώδικας ταξινόμησης πίνακα αληθείας [sorting_truth_table.py]

Κώδικας σε γλώσσα Python δέχεται σαν είσοδο την de Bruijn ακολουθία που έχει παραχθεί και δίνει σαν έξοδο την σειρά των ψηφίων με βάση την δομή του πίνακα αληθείας που ορίζει το πρόγραμμα SageMath [sorting_truth_table.py]

```
#! /usr/bin/env python

sa='00010111' #initial sequence

n=3

s='00010111000'
```

```
list1=[]

newlist=[]

for i in s:

        list1.append(i)    #if I add int() -> transform to integer

for i in range(len(list1)-n):

        newlist.append(list1[i:i+(n+1)])

list2=[]

newlist2=[]

for i in s:

        list2.append(i)

for i in range(len(list2)-n):

        newlist2.append(list2[i:(i+n)])




# reverse raw of each element
```

```
board = []

for i in newlist2:

        board.append(i[::-1])

for i in range(2**n):

        board[i].append(newlist[i][n])

newnewlist=[]

for newitem in board:

        newitem[0:n] = [''.join(newitem[0:n])]

        newnewlist.append(newitem)

def getKey(items):

        return items[0]    #sorting based on first element



zz = sorted(newnewlist, key=getKey)

finalist =[]

for i in range(2**n):

        finalist.append(zz[i][1])

print finalist
```

```
solution = ",".join(finalist)

print solution
```

A.2 Πρόγραμμα [cross_join_pair.py]

Πρόγραμμα σε γλώσσα Python τέτοιο ώστε για δοθείσα ακολουθία υπολογίζει όλες της καταστάσεις της, καθώς επίσης για τις καταστάσεις των συζυγών ζευγών που έχουμε επιλέξει υπολογίζει τις θέσεις τους στον κύκλο [cross_join_pair.py]

```
sa='00000110011111000100101011101101'

n=5

s='000001100111110001001010111010100000'



list1=[]

new_list1=[]

for i in s:

        list1.append(i)    #if I put int() -> integer

for i in range(len(list1)-n):
```

```
        new_list1.append(list1[i:i+(n)])

new_list3 = []

for newitem in new_list1:

        c = [''.join(newitem[0:n])]

        new_list3.append(c)

#for i, j in enumerate(new_list3):

        #print (i,j)

for i in new_list3 :

        print i

print '################################################################'

print "a = %d " % new_list3.index(['00001'])

print "b = %d " % new_list3.index(['00110'])

print "-a = %d " % new_list3.index(['10001'])

print "-b = %d " % new_list3.index(['10110'])
```

A.3 Πρόγραμμα [joining_splitting.py]

Πρόγραμμα σε γλώσσα Python υπολογίζει για υπάρχουσες ακολουθίες de Bruijn s, r τάξης n , όλες τις πιθανές (n+1)άδες τις κάθε $D_0^{-1}(r)$, $D_1^{-1}(r)$, $D_0^{-1}(s)$, $D_1^{-1}(s)$ ακολουθίες που ορίσαμε καθώς επίσης ελέγχει εάν οι $D_0^{-1}(s)$ και $D_1^{-1}(r)$ περιέχουν τις ίδιες (n+1)άδες και ταυτόχρονα βρίσκει όλα τα συζυγή ζευγάρια που η μια εξ αυτών βρίσκεται στην $D_0^{-1}(r)$ και η άλλη στην $D_1^{-1}(s)$.

```
#! /usr/bin/env python

sb = '00000101011111011000110100111001'

sc = '01010011101100011010111110010000'

n=5

list2 = []

for i in sb:

        list2.append(int(i))

list3 = []

for i in range(len(list2)):

        list3.append(sum(list2[0:i]))

list4 = []

for i in list3:
```

```
        if i % 2 == 0 :
                list4.append(0)
        else:
                list4.append(1)
#transformation to n group
for i in range(n):
        list4.append(list4[i])


print "Ayth einai h D0(s): ", list4[0:2**n]
listx1 = []
for i in range(len(list4)-n):
        listx1.append(list4[i:i+(n+1)])


print listx1
print "----------"
#end of transformation
```

```
list5 = []

for item in list4:

        if item == 0 :

                list5.append(1)

        else:

                list5.append(0)

print "Ayth einai h D1(s):", list5[0:2**n]

#start of transformation

listx2 = []

for i in range(len(list5)-n):

        listx2.append(list5[i:i+(n+1)])



print listx2

print "\n"

#end of transformation

list12 = []

for i in sc:
```

```
        list12.append(int(i))

list13 = []

for i in range(len(list12)):

        list13.append(sum(list12[0:i]))

list14 = []

for i in list13:

        if i % 2 == 0 :

                list14.append(0)

        else:

                list14.append(1)

#start of transformation

for i in range(n):

        list14.append(list14[i])

print "Ayth einai h D0(r): ", list14[0:2**n]

listx11 = []

for i in range(len(list14)-n):

        listx11.append(list14[i:i+(n+1)])
```

```
print listx11

print "----------"

#end of transformation

list15 = []

for item in list14:

        if item == 0 :

                list15.append(1)

        else:

                list15.append(0)

print "Ayth einai h D1(r):", list15[0:2**n]

#start of transformation

listx12 = []

for i in range(len(list15)-n):

        listx12.append(list15[i:i+(n+1)])

print listx12

print "\n"

#end of transformation
```

```
diff = []

for i in listx1:

        if i not in listx11:

                diff.append(i)

print "Aytes einai oi diafores: ", diff

for i in range(len(listx1)):



                if listx1[i][0] == 0 :

                        listx1[i][0] = 1

                else :

                        listx1[i][0] = 0

#print "Ayth einai h syzygh ths D0(s): ", listx1

same = []

for i in listx1:

        if i in listx12:

                same.append(i)

print "Aytes einai oi idies anamesa sthn symplhrwmatikh ths D0(s) kai sthn D1(r): ", same
```