Πανεπιστήμιο Πειραιώς
Σχολή Τεχνολογιών, Πληροφορικής και Επικοινωνιών
Τμήμα Πληροφοριακών Συστημάτων
Π.Μ.Σ. Πληροφορικά Συστήματα και Υπηρεσίες
Κατεύθυνση Μεγάλων Δεδομένων και Αναλυτικής

Παράλληλοι Υπολογισμοί στην Κρυπτογράφηση Εικόνων

Χρήστος Πηλιχός

Επιβλέπων: Μιχαήλ Φιλιππάκης
Αναπληρωτής Καθηγητής

Φεβρουάριος 2021

Αφιερωμένη στους γονείς μου
και σε ένα ακόμη πολύτιμο μέλος της οικογενείας!

Παράλληλοι Υπολογισμοί στην Κρυπτογράφηση Εικόνων

Χρήστος Πηλιχός
(me1820)

Φεβρουάριος 2021

Τριμελής Εξεταστική Επιτροπή

Κυριαζής Δημοσθένης	Αναπληρωτής Καθηγητής	
Φιλιππάκης Μιχαήλ	Αναπληρωτής Καθηγητής	(επιβλέπων)
Τελέλης Ορέστης	Επίκουρος Καθηγητής	

Περιεχόμενα

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

Κάποιες φορές η επικοινωνία ανάμεσα σε οντότητες καλείται να συμβεί σε ένα ανασφαλές περιβάλλον. Μπορεί να λάβει χώρα σε ένα δια ζώσης περιβάλλον ή σε ένα εικονικό περιβάλλον. Γι αυτό αναπτύχθηκαν αλγόριθμοι οι οποίοι επιτρέπουν την επικοινωνία οντοτήτων παρουσία αντιπάλων*. Στην ολότητά τους οι εν λόγω αλγόριθμοι συνθέτουν τον κλάδο των Μαθηματικών, καλούμενο Κρυπτογραφία. Ειδικά, στον χώρο του διαδικτύου, πολλές φορές σε ένα δευτερόλεπτο θα χρειαστεί η επαλήθευση πως δεν έχει παρεισφρήσει κάποιος αντίπαλος σε μια επικοινωνία. Επίσης, ένα μήνυμα, το οποίο είναι εμφανές από όλους, θα πρέπει να γίνει κατανοητό μόνον από τον νόμιμο παραλήπτη του. Άρα, η τάση απόκρυψης του περιεχομένου ενός μηνύματος (κι όχι του του γεγονότος ότι επρόκειται για ένα μήνυμα), εγείρει την ανάγκη ποσοτικοποίησης του επιπέδου της παρεχόμενης ασφάλειας. Επιπλέον, οι αλγόριθμοι θα πρέπει να γίνονται περισσότερο ανεκτικοί σε προσπάθειες παραβίασης και ταυτόχρονα πιο γρήγορα υλοποίησιμοι (καθόσον δεν θα πρέπει να κωλυσιεργούν την πραγματική ανάγκη, όπως την αποστολή ενός μηνύματος). Συνεπώς, ανακύπτει μια διαρκής αναζήτηση νέων, γρηγορότερων και αποτελεσματικότερων τεχνικών που να διασφαλίζουν την ασφαλή (διαδικτακή) επικοινωνία.

Μία ρήση υπαγορεύει πως:

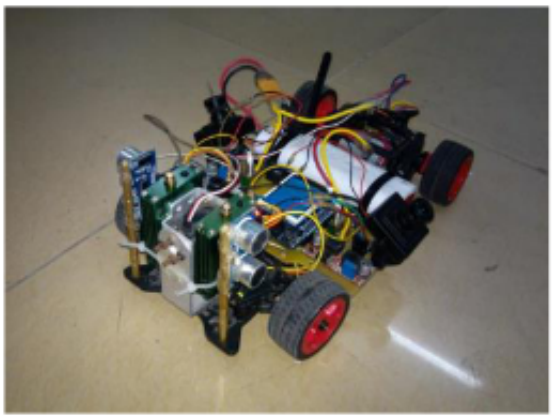
> "Μία εικόνα ισοδυναμεί με χίλιες λέξεις"
>
> -Κινέζικο ρητό

Το παραπάνω ρητό βρίσκει εφαρμογή και στην αναπαράσταση των εικόνων στον ηλεκτρονικό υπολογιστή. Όπως κάθε λέξη έχει το δικό της νόημα μέσα

*Κάθε μη εμπλεκόμενη οντότητα η οποία επιθυμεί να γίνει κοινωνός μιας πληροφορίας, η οποία δεν την αφορά.

στην πρόταση, έτσι και το εκάστοτε εικονοστοιχείο (pixel) συνεισφέρει τη δική του πληροφορία στην εικόνα. Η αναπαράσταση των εικόνων στους αλγορίθμους Μηχανικής Μάθησης γίνεται εικονοστοιχείο προς εικονοστοιχείο. Στην Μηχανική Μάθηση έχουν αναπτυχθεί αλγόριθμοι οι οποίοι προσανατολίζονται στο να μεταχειρίζονται καταλλήλως τα δεδομένα στην πλήρη τους έκταση. Επί παραδείγματι, στην Επεξεργασία Φυσικής Γλώσσας (Natural Language Process - NLP) κάθε λέξη συνιστά μια έννοια, μέσα από τα συμφραζόμενα εντός της πρότασης στην οποία ανήκει. Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNN) με την εφαρμογή φίλτρων δύνανται να εκπαιδευτούν στο να αναγνωρίζουν πρότυπα σε εικόνες. Ωστόσο, όσο πιο εμπεριστατωμένη είναι η προσέγγιση, τόσο πιο απαιτητική σε χρόνο βαίνει να είναι. Συνεπώς, οι αλγόριθμοι των NLP και CNN απαιτούν περισσότερο χρόνο. Γίνεται αντιληπτό πως μία εικόνα, ως κωδικοποίηση εκονοστοιχείο προς εικονοστοιχείο, συνιστά ένα αντικείμενο χρονοβόρο στην επεξεργασία του. Επομένως, ανακύπτει το ερώτημα εάν μπορούν κάποιοι υπολογισμοί οι οποίοι απαιτείται να γίνουν, να διεξαχθούν με μια πιο αποτελεσματική οργάνωση χρόνου. Στα πλαίσια της Εργασίας, κάθε εικόνα θεωρείται ως ένα χρονοβόρα διαχειρίσιμο δεδομένο και κατά μείζονα λόγο το αυτό ισχύει και για μια ροή εικόνων. Συνεπώς, σκοπόντας τις εικόνες -και κατ᾽ επέκτασην μια ροή τους- ως μεγάλο δεδομένο, χρησιμοποιούνται τεχνικές και εργαλεία με σκοπό να ανακύψει μια πιο αποτελεσματική διαχείρησή τους. Όσο ο όγκος της εισρεόμενης πληροφορίας αυξάνει, τόσο πιο απαιτητική γίνεται η διαχείρηση της, δηλαδή η διαχείρηση μεγάλων δεδομένων.

Μία εικόνα μπορεί να εμπεριέχει ευαίσθητα δεδομένα τα οποία δεν πρέπει να αποκαλυφθούν σε μη-εξουσιοδοτημένες οντότητες (όπως για παράδειγμα, ιατρικά δεδομένα [ASKC03]). Ένα έξυπνο αμαξίδιο ασφαλείας δύναται να φέρει μια κάμερα η οποία μαγνητοσκοπεί τα τεκτενόμενα τριγύρω του. Αποτελεί μια λύση σε μέρη όπου είναι επισφαλής η ανθρώπινη παρουσία, ενώ έχει τη δυνατότητα να ελίσσεται περισσότερο απ᾽ όσο ένας άνθρωπος. Τα αμαξίδιο, πέραν της κάμερας και του δέκτη του (από τον οποίον λαμβάνει οδηγίες), δύναται να είναι εξοπλισμένο με έναν αναμεταδότη ο οποίος αποστέλλει την καταγραφόμενη ροή. Η ροή διαδίδεται



Σχήμα 1.1: Το έξυπνο αμαξίδιο ασφαλείας [WSGYJ13]

ασύρματα, άρα οποιαδήποτε τρίτη -μη εξουσιοδοτημένη- οντότητα δύναται μέσω ενός δέκτη να λάβει τη ροή επίσης. Κατά συνέπεια, ανακύπτει το θέμα εξεύρεσης μιας αλγοριθμικής τεχνικής κατά την οποία η μεταδιδόμενη ροή αν ληφθεί από μια μη εξουσιοδοτημένη οντότητα δεν θα μπορέσει να αξιοποιηθεί. Ωστόσο, η ροή θα πρέπει να συνεχίσει να είναι εν εξελίξει, δηλαδή η παραλλαγή της να μην κοστίζει χρόνο ο οποίος θα καθυστερεί τη μετάδοσή της, καθιστόντας την ανακριβή την αναπαράσταση των εν εξελίξει γεγονότων.

Σκοπός της Εργασίας

Ο σκοπός της Εργασίας έγκειται στην (απο)κρυπτογράφηση εικόνων σε μία συσκευή με χρήση παράλληλων υπολογισμών. Κατά την Εργασία, επιχειρείται ο συγκερασμός της Κρυπτογραφίας με τα Μεγάλα Δεδομένα, με σκοπό την πλήρωση των δύο απαιτήσεων οι οποίες ετέθησαν ανωτέρω:

- Τα δεδομένα να μην είναι ανακτήσιμα από τρίτες οντότητες.
- Τα δεδομένα να αναπαριστούν την τρέχουσα επικρατούσα κατάσταση.

Τόσο ο τομέας της Κρυπτογραφίας (εν συνόλω, κι όχι μόνον η πτυχή της η οποία εμπεριέχεται στην παρούσα Εργασία), όσο και ο τομέας διαχείρησης των μεγάλων δεδομένων αποτελούν δύο διαρκώς εξελισσόμενους τόμεις, οι οποίοι καλούνται σε κάθε βήμα τους να ικανοποιούν "βασικές ανάγκες" στον κόσμο του διαδικτύου. Αφ' ενός, η Κρυπτογραφία θα πρέπει να διασφαλίζει τη μη διαρροή δεδομένων προς τρίτους καθώς αυτά ανταλλάζονται δια μέσω του διαδικτύου. Αφ' ετέρου, η ποσότητα εισρεόμενης πληροφορίας αυξάνει με την εξέλιξη του διαδικτύου και των ηλεκτρονικών υπολογιστών. Άρα, η διαχείρησή της καθίσταται αναγκαία.

Διάρθρωση της Εργασίας

Η εργασία αποτελείται από 8 Κεφάλαια, τα εξής:

Κεφάλαιο 1: Πρόκειται για την παράθεση του σκοπού της Εργασίας καθώς τη χρησιμότητα της λύση την οποίο η Εργασία υλοποιεί.

Κεφάλαιο 2: Σταχυολογούνται μερικές παρόμοιες εφαρμογές παράλληλων υπολογισμών στην Κρυπτογραφία (σε ευρύτερο πλαίσιο κι όχι μόνον στα περιεχόμενα της Εργασίας).

Κεφάλαιο 3: Παρατίθενται μερικοί από τους βασικούς ορισμούς οι οποίοι αφορούν την Κρυπτογραφία και θα απαντηθούν στην Εργασία.

Κεφάλαιο 4: Παρουσιάζονται αλγόριθμοι οι οποίοι προκρίθηκαν στη τελικής φάση διεξαγωγής του διαγωνισμού AES του NIST: ο Rijndael και ο Serpent.

Κεφάλαιο 5: Παρουσιάζονται τα χαοτικά κρυπτοσυστήματα για τα οποία υπάρχουν ενδείξεις πως μπορούν να αποδόσουν αποτελεσματικούς αλγορίθμους κρυπτογράφησης εικόνων.

Κεφάλαιο 6: Συνθέται μια εισαγωγή στο πως οι ηλεκτρονικοί υπολογιστές προσπελαύνουν διεργασίες παράλληλα, αλλα και μια εισαγωγή στο `Spark`, το πρόγραμμα διαχείρησης μεγάλων δεδομένων το οποίο προσφέρει τη δυνατότητα διεξαγωγής πράλληλων υπολογισμών.

Κεφάλαιο 7: Εμπεριέχεται η υλοποίηση η οποία έγινε στα πλαίσια της Εργασίας, η οποία αποτελεί την έμπρακτη εφαρμογή ενός μέρους από τα όσα παρουσιάσθηκαν.

Κεφάλαιο 8: Συνοψίζει τα όσα παρουσιάστηκαν κατά την Εργασία, διενεργεί μία ανασκόπηση στα εργαλεία τα οποία χρησιμοποιήθηκαν καθώς και προτείνει μια επιπλέον υλοποίηση (παραλλαγή) της εφαρμογής που προηγήθηκε.

Κεφάλαιο 2

Προηγούμενα αποτελέσματα

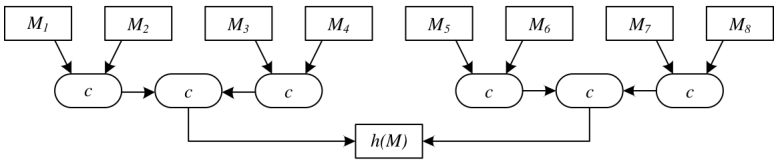
Υπάρχουν διάφοροι τρόποι σύμφωνα με τους οποίους μπορούν να διαμοιραστούν οι υπολογισμοί εν προκειμένω να επιτευχθεί η παραλληλία (δηλαδή η υλοποίηση διαφορετικών υπολογισμών στο ίδιο χρονικό διάστημα):

1. Χρήση πόρων της ίδιας συσκευής (χρησιμοποιώντας τις πολλαπλές μονάδες επεξεργασίας (CPUs) τις οποίες ενδέχεται να διαθέτει).
2. Εκχώρηση υπολογισμών σε ένα υπολογιστικό νέφος [Ba17].

Η Εργασία ασχολείται με τον πρώτο τρόπο αξιοποίησης των πόρων.

Παράλληλοι υπολογισμοί στην Κρυπτογραφία

Οι παράλληλοι υπολογισμοί μπορούν να μην εξαντλούνται στην απλή εφαρμογή μιας συνάρτησης, αλλά δύναται να συνθέτουν μια αλληλουχία υπολογισμών οι οποίοι πρέπει να πραγματοποιηθούν στο ίδιο χρονικό διάστημα και κατόπιν να χρησιμοποιηθούν από άλλες παράλληλες διεργασίες. Η εν λόγω αλληλουχία δύναται να αναπαρασταθεί μέσω μιας δενδρικής δομής, η οποία υποδηλώνει την εξάρτηση των παράλληλων διεργασιών.



Σχήμα 2.1: Ένα παράδειγμα δενδρικών υπολογισμών σε συναρτήσεις κατακερματισμού (όπου η είσοδος έχει τμηματικά διαμερισθεί) [KDB10]

Μία χρήση των παράλληλων υπολογισμών στην Κρυπτογραφία συνίσταται στον υπολογισμό συναρτήσεων κατακερματισμού* (hash function). Μια αρχική προσπάθεια παράλληλης υλοποίησης συνάρτησεων κατακερματισμού επιχειρήθηκε από τον Ivan Damgård το 1989 στο [Da89], η οποία βελτιώθηκε αργότερα από τους Palash Sarkar και Paul Scellenberg το 2001 στο [PS01] και από τους Pinakpani Pal και Palash Sarkar το 2003 στο [PS03]. Η υλοποίηση των Mihir Bellare και Daniele Micciancio στο [BM97] το 1997, μπορεί να θεωρηθεί ως ένα δένδρο υπολογισμών 2 επιπέδων, παρ' όλο που για ένα μέρος τον υπολογισμών έχει βρεθεί μία επιτυχής επίθεση από τους Stephane Manuel και Nicolas Sendrier, στο [MS07]. Επίσης, με χρήση παράλληλων υπολογισμών μπορούν να υλοποιηθούν και κάποιες από τις συναρτήσεις οι οποίες συμμετείχαν διαγωνισμό SHA-3 [FIPS202]:

- Skein ([Skein] συμμετείχε στη τελική φάση του διαγωνισμού) [AEMR10].
- MD6 ([MD6] δεν προκρίθηκε στον δεύτερο γύρο του διαγωνισμού) [BD09].

Συγκλίνοντας προς τον σκοπό της Εργασίας

Ο σκοπός της Εργασίας θέλει τη χρήση πόρων της ιδίας συσκευής (καθόσον θεωρείται πως το έξυπνο αμαξίδιο της Εικόνας 1.1 θα πρέπει είναι αποκλειστικά υπεύθυνο για την κρυπτογράφηση των εικόνων). Ακολουθούν μερικές, εγγύτερες στον σκοπό, παρόμοιες μελέτες.

Ένα πανόραμα στην παράλληλη υλοποίηση κρυπτοσυστημάτων (στα οποία συγκαταλέγονται και εκείνα του Κεφαλαίου 4) περιέχεται στο [Al18]. Τα κρυπτοσυστήματα υλοποιούνται σε προγραμματιστικό περιβάλλον το οποίο υποστηρίζει παράλληλους βρόγχους, πολυνηματική διαχείρηση διεργασιών, ασύγχρονους υπολογισμούς και χρήση πολλαπλών επεξεργαστών.

Μια τεχνική παράλληλων υλοποιήσεων μπορεί να επιτευχθεί μεσω της χρήσης Γραφικών Μοναδων Επεξεργασίας (Graphics Process Unit - GPU) στην ίδια συσκευή. Κατά την εν λόγω υλοποίηση οι φυσικές μονάδες επεξεργασίας (CPUs) διαμοιράζονται ώστε να εξυπηρετούν περισσότερες διεργασίες εν παραλλήλω. Μια υλοποίηση του κρυπτοσυτήματος Serpent (§4.2) με χρήση GPUs εμπεριέχεται στο [NHA09].

*Σε ένα αφηρημένο επίπεδο θεώρησης, η συνάρτηση κατακερματισμού λαμβάνει μία είσοδο αυτθαίρετου μήκους, την οποία απεικονίζει σε μια σταθερού μήκους έξοδο.

Μέρος Ι

Αλγόριθμοι

Κεφάλαιο 3

Κρυπτογραφία

Η Κρυπτογραφία αποτελείται από αλγορίθμους οι οποίοι μπορούν να ταξινομηθούν αναλόγως τη χρήση των τεχνικών και εργαλίων που χρησιμοποιούν. Η Κρυπτογραφία πραγματεύεται τον μετασχηματισμό των δεδομένων εν προκειμένω το περιοχόμενο αυτών [RfC2828]:

1. Να είναι σε μια μη κατανοητή μορφή.
2. Να μην είναι δυνατό να υποστεί μη ανιχνεύσιμη αλλοίωση.
3. Να εμποδιστεί η μη εξουσιοδοτημένη χρήση του.

3.1 Κρυπτοσυστήματα

Ως **κρυπτοσύστημα** ορίζεται η πλειάδα

$$\langle \mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D} \rangle$$

όπου:

- $\mathcal{M}$ είναι ο χώρος των **απλών κειμένων**.
- $\mathcal{C}$ είναι ο χώρος των **κρυπτοκειμένων**.
- $\mathcal{K}$ είναι ο χώρος των **κλειδιών**.
- $\mathcal{E} : \mathcal{K} \times \mathcal{M} \longrightarrow \mathcal{C}$, συνιστά μια εύκολα υπολογίσιμη συνάρτηση, η οποία καλείται **συνάρτηση κρυπτογράφησης**. Συμβολίζεται δε $\mathcal{E}_K(\cdot) \equiv \mathcal{E}(K, \cdot)$, για $K \in \mathcal{K}$.

- $\mathcal{D} : \mathcal{K} \times \mathcal{C} \longrightarrow \mathcal{M}$, όπου $\mathcal{D} \equiv \mathcal{E}^{-1}$, συνιστά μια συνάρτηση η οποία είναι
  - εύκολα υπολογίσιμη δεδομένου ενός $K \in \mathcal{K}$,
  - ο υπολογισμός -σε λογικά χρονικά πλαίσια της προ-εικόνας- $M \in \mathcal{M}$ από το $\mathcal{E}_K(M) \in \mathcal{C}$, μέσω της $\mathcal{D}$, δίχως τη γνώση του $K \in \mathcal{K}$) είναι ανέφικτος.

  Η εν λόγω συνάρτηση καλείται **συνάρτηση αποκρυπτογράφησης**. Συμβολίζεται δε, ως $\mathcal{D}_K(\cdot) \equiv \mathcal{D}(K, \cdot)$, για $K \in \mathcal{K}$.

Συνεπώς, ένα $M \in \mathcal{M}$ καλείται **απλό κείμενο** (plain text και γι αυτό συμβολίζεται και ως P), ή **μήνυμα** (message) κι ένα $C \in \mathcal{C}$ καλείται **κρυπτοκείμενο** (ciphertext).

Από το γεγονός ότι $\mathcal{D} \equiv \mathcal{E}^{-1}$, προκύπτει πως το κρυπτοσύστημα θα πρέπει να διέπεται από την κατωτέρω ιδιότητα:

$$(\forall K_e \in \mathcal{K})(\forall M \in \mathcal{M})\Big[M = \mathcal{D}_{K_d}\big(\mathcal{E}_{K_e}(M)\big)\Big] \quad \text{(ορθότητα κρυπτοσυστήματος)}$$

όπου $K_d \equiv K_d(K_e)$ (δηλαδή το κλειδί αποκρυπτογράφησης διέπεται από κάποια σχέση με το κλειδί κρυπτογράφησης). Αν στο κρυπτοσύστημα ισχύει ότι:

- πάντοτε $K_e \neq K_d$ (δηλαδή διαφορετικό κλειδί χρησιμοποιείται κατά την κρυπτογράφηση/αποκρυπτογράφηση), τότε το K_e καλείται **κλειδί κρυπτογράφησης**, ενώ το K_d **κλειδί αποκρυπτογράφησης** και τα εν λόγω κρυπτοσυστήματα καλούνται **κρυπτοσυστήματα δημοσίου κλειδιού**.
- πάντοτε $K_e = K_d$ (δηλαδή το κλειδί είναι κοινό για την κρυπτογράφηση/αποκρυπτογράφηση), τότε το $K_e \equiv K_d$ καλείται **συμμετρικό (κοινό) κλειδί** και τα εν λόγω κρυπτοσυστήματα καλούνται **συμμετρικά κρυπτοσυστήματα**.

Η ασφάλεια του κρυπτοσυστήματος επαφίεται στη δυσκολία υπολογισμού του M δεδομένων:

- του κρυπτοσυστήματος (δηλαδή του αλγορίθμου),
- του $\mathcal{E}_{K_e}(M)$ (του κρυπτοκειμένου)
- *μόνο για τα κρυπτοσυστήματα δημοσίου κλειδιού*: και του K_e (του δημοσίου κλειδιού).

3.2 Μια ταξινόμηση

Δεδομένου του διαχωρισμού που προηγήθηκε, μία ευρεία ταξινόμηση της Κρυπτογραφίας μπορεί να αποτελέσει η ακόλουθη:

Κρυπτογραφία Δημοσίου Κλειδιού: Κάθε οντότητα έχει δύο κλειδιά: ένα ιδιωτικό το οποίο χρησιμοποιείται μόνο από την οντότητα και ένα δημόσιο το οποίο είναι κοινοποιημένο ώστε να το χρησιμοποιήσει κάθε άλλη ενδιαφερόμενη οντότητα η οποία επιθυμεί να συνάψει επικοινωνία με την οντότητα-κάτοχο του δημοσίου κλειδιού. Συνεπώς, $K_e \neq K_d$ στον ορισμό του κρυπτοσυστήματος που προηγήθηκε.

Αλγόριθμοι κρυπτογράφησης δημοσίου κλειδιού: Το δημόσιο κλειδί χρησιμοποιείται για την κρυπτογράφηση των μηνυμάτων από μια άλλη οντότητα με στόχο μόνο η οντότητα κάτοχος του δημοσίου κλειδιού να μπορεί να το αποκρυπτογραφήσει, με χρήση του ιδιωτικού της κλειδιού.

Ψηφιακές Υπογραφές: Μία οντότητα κρυπτογραφεί το ένα μύνυμα με το ιδιωτικό της κλειδί. Κάθε άλλη ενδιαφερόμενη οντότητα μπορεί να χρησιμοποιήσει το δημόσιο κλειδί της αρχικής οντότητας ώστε να αποκρυπτογραφήσει το μήνυμα. Η χρήση μιας ψηφιακής υπογραφής σε ένα μήνυμα, διασφαλίζει την [DS]:

πιστοποίηση: παρέχει στον παραλήπτη την ισχυρή πεποίθηση πως το μήνυμα εστάλλει από τον πραγματικό αποστολέα κι όχι κάποιον που προσπαθεί να τον μιμηθεί.

ακεραιότητα: το περιεχόμενο του μηνύματος δεν παραλλάχθηκε από την αποστολή, μέχρι την παραλαβή του.

Συμμετρική Κρυπτογραφία: Οι επικοινωνούσες οντότητες μοιράζονται την ίδια πληροφορία (κλειδί) η οποία και πρέπει να διατηρηθεί μυστική καθ' όσον χρησιμοποιείται και για την κρυπτογράφηση, αλλά και για την αποκρυπτογράφηση μηνυμάτων. Συνεπώς, $K_e = K_d$ στον ορισμό του κρυπτοσυστήματος που προηγήθηκε.

Αλγόριθμοι Ροής: Το μήνυμα προσπελαύνεται χαρακτήρα προς χαρακτήρα. Σε κάθε χαρακτήρα εφαρμόζεται ένας μαθηματικός τύπος. Τα κρυπτοσυστήματα της κατηγορίας διακρίνονται σε μονοαλφαβητικούς (ο ίδιος μετασχηματισμός εφαρμόζεται σε όλους τους χαρακτήρες του μηνύματος) και πολυαλφαβητικούς (οι οποίοι χρησιμοποιούν διαφορετικούς μετασχηματισμούς ανά χαρακτήρα) μετασχηματισμούς.

Αλγόριθμοι τμήματος: Το μήνυμα διαμερίζεται σε ισομήκεις ακολουθίες διαδοχικών χαρακτήρων. Κάθε μία ακολουθία προσπελαύνεται από τον αλγόριθμο (αποτελεί το αντικείμενο του Κεφαλαίου 4).

3.3 Συμμετρικά κλειδιά

Από την Ενότητα που προηγήθηκε, κατέστει σαφές πως οι οντότητες οι οποίες προτίθενται να συνάψουν επικοινωνία, θα πρέπει να συμφωνήσουν σε μια κοινή πληροφορία (συμμετρικό κλειδί) το οποίο θα παρέχει σε αμφότερες της δυνατότητα κρυπτογράφησης και αποκρυπτογράφησης των μηνυμάτων που ανταλλάζουν. Ο σχεδιασμός των κρυπτοσυστημάτων θα πρέπει (μεταξύ άλλων) να διέπεται από την αρχή του Kerchoff [MOV01]: *το επίπεδο ασφαλείας το οποίο παρέχει το κρυπτοσύστημα θα πρέπει αποκλειστικά να εξαρτάται από το κλειδί το οποίο χρησιμοποιείται.*

Ο τρόπος κατά τον οποίον παράγονται τα κλειδιά, θα πρέπει να είναι (ή τουλάχιστον να φαντάζει) τυχαίος. Σε διαφορετική περίπτωση υπάρχουν επιθέσεις οι οποίες μπορούν να εξάγουν πληροφορία από τα κρυπτοκείμενα και εν τέλει να ανακτήσουν το κλειδί. Από την παραπάνω διατύπωση διακρίνεται η ύπαρξη "ασθενών" κλειδιών στον χώρο των κλειδιών. Ένα κλειδί χαρακτηρίζεται ως *ασθενές* όταν μπορεί να ανακτηθεί το απλό κείμενο μέσα από μια συλλογή κρυπτοκειμένων σε λιγότερο υπολογιστικό κόστος σε σχέση με τα υπόλοιπα κλειδιά (:στοιχεία του χώρου των κλειδιών).

Στα πλαίσια της Εργασίας, ένα συμμετρικό κλειδί συνιστά:

Για τους τμηματικούς αλγορίθμους: Μια ακολουθία από bytes κατάλληλου μήκους.

[Ειδικά, τα κρυπτοσυστήματα του Κεφαλαίου 4, συνοδεύονται από έναν αλγόριθμο παραγωγής κλειδιού (key scheduling). Ο αλγόριθμος αποσκοπεί στη δημιουργία κλειδιών για κάθε γύρο του αλγορίθμου (απο)κρυπτογράφησης σύμφωνα με έναν τρόπο ο οποίος δεν φανερώνει τη συσχέτιση αναμεταξύ των παραγομένων κλειδιών. Μια απαίτηση από τη διαδικασία παραγωγή κλειδιού είναι πως αν πρόκειται να παραχθούν κλειδιά μήκους $n \in \mathbb{N}$ δυφίων, τότε κάθε κλειδί με το εν λόγω μήκος θα πρέπει να είναι ισοπίθανο να επιλεγεί από τη διαδικασία (δηλαδή να έχει πιθανότητα 2^{-n} να χρησιμοποιηθεί από τον αλγόριθμο. Ένας χώρος κλειδιών δίχως αρκετά στοιχεία μπορεί να επιτρέψει μια εξαντλητική αναζήτηση διαρρηγνύοντας το κρυπτοσύστημα (κι άρα και το επίπεδο ασφαλείας το οποίο παρέχει).]

Για τους χαοτικούς αλγορίθμους: Οι παράμετροι οι οποίοι εναρμονίζουν τα χαοτικά συστήματα στον αποστολέα και στον παραλήπτη του μηνύματος.

[Οι αρχικοί όροι των χαοτικών ακολουθιών αποτελούν μεροληπτικές εξαρτήσεις από τον πρώτο όρο της ακολουθίας. Γι αυτό τον λόγο, θα πρέπει να χρησιμοποιούνται αρκετά μεταγενέστεροι όροι της ακολουθίας (όπου η επιρροή των αρχικών συνθηκών θα έχει εξαλειφθεί). Επίσης, η επιλογή των αρχικών συνθηκών (παράμετροι και αρχικός όρος) επηρεάζει και το επίπεδο του χάους το οποίο εμφανίζει η ακολουθία. Συνεπώς, ανακύπτει πως υπάρχουν αρχικές παράμετροι οι οποίες οδηγούν σε πιο προβλέψιμες καταστάσεις (ακόμη και στους μεταγενέστερους όρους τους), δηλαδή ασθενή κλειδιά.]

3.3.1 Αλγόριθμοι ανταλλαγής κλειδιού

Οι αλγόριθμοι οι οποίοι πρόκειται να αναπτυχθούν στο Μέρος Ι κατατάσσονται στους συμμετρικούς αλγορίθμους. Κατά συνέπεια, ανάκυπτει -εν γένει για τη Συμμετρική Κρυπτογραφία- το πρόβλημα της ασφαλούς ανταλλαγής του κλειδιού καθόσον η εν λόγω κοινή πληροφορία αποτελεί, εν προκειμένω, το απαραίτητο συστατικό για την άμεση κρυπτογράφηση/αποκρυπτογράφηση της μεταδιδόμενης πληροφορίας. Μία άμεση λύση είναι η δια ζώσης ανταλλαγή του κλειδιού. Ωστόσο, επειδή κατά κόρον η δια ζώσης ανταλλαγή δεν είναι εφικτή, αλλά και επειδή για λόγους ασφαλείας το κλειδί αλλάζει περιοδικά, θα πρέπει οι συνδαιτημόνες, να μπορούν να καταλήξουν στην κοινή μυστική πληροφορία (συμμετρικό κλειδί) δίχως πρότερη επαφή ή συνεννόηση. Προς τούτο, παρέχονται οι αλγόριθμοι ανταλλαγής κλειδιού.

Diffie-Hellman

Παρακάτω παρουσιάζεται το πρωτόκολλο των Diffie-Hellman [DH76], όπου δύο ενδιαφερόμενες οντότητες συμφωνούν σε μια κοινή πληροφορία η οποία είναι γνωστή μόνο σε εκείνες.

Για τις ανάγκες του πρωτοκόλλου, οι οντότητες συμφωνούν δημόσια σε μια ομάδα $\langle G, * \rangle$, όπου G είναι ένα σύνολο στοιχείων και $* : G \times G \longrightarrow G$ είναι μία πράξη επί των στοιχείων του συνόλου G. Για $g \in G$ και $\alpha \in \mathbb{N}$, συμβολίζεται ως g^{α} η επαναλαμβανόμενη εφαρμογή της πράξης $*$ στο στοιχείο g με τον εαυτό του, ήτοι $g^2 = g * g$, $g^3 = g * g * g$, και ούτο καθ' εξής. Ως **γεννήτορας** της ομάδος $\langle G, * \rangle$ καλείται εκείνο το στοιχείο (αν υπάρχει) $g \in G$ το οποίο έχει την ιδιότητα να παράξει όλα τα στοιχεία της ομάδας εφαρμοζόμενο στο εαυτό

του, δηλαδή $(\forall h \in G)(\exists \alpha \in \mathbb{N})[h = g^\alpha]$. Το πρωτόκολλο των Diffie-Hellman έχει ως ακολούθως:

Έστω μια ομάδα $\langle G, * \rangle$ και $g \in G$ ένας γεννήτορας αυτής

Αλίκη		**Βασίλης**
Επιλέγει $\alpha \in \mathbb{N}$		Επιλέγει $\beta \in \mathbb{N}$
Υπολογίζει $g^\alpha \in G$		Υπολογίζει $g^\beta \in G$
Κοινοποιεί το g^α	$\longrightarrow$	g^α
g^β	$\longleftarrow$	Κοινοποιεί το g^β
Υπολογίζει $(g^\beta)^\alpha \in G$		Υπολογίζει $(g^\alpha)^\beta \in G$

Εν τέλει οι δύο οντότητες (εν προκειμένω, η Αλίκη και ο Βασίλης) κατέληξαν στην κοινή πληροφορία $(g^\alpha)^\beta = g^{\alpha\beta} = g^{\beta\alpha} = (g^\beta)^\alpha$. Το παραπάνω πρωτόκολλο μπορεί να επαναληφθεί έως ότου οι δύο οντότητες να έχουν συμφωνήσει στην ολότητα της πληροφορίας της οποίας επιθυμούν να χρησιμοποιήσουν ως συμμετρικό κλειδί. Επίσης, καμμία εκ των οντοτήτων δεν αποκαλύπτει την προσωπική (μυστική) επιλογή του εκθέτη, ούτε αναμεταξύ τους, ούτε προς τρίτους. Η δυσκολία εύρεσης του εκθέτη $\alpha \in \mathbb{N}$, δεδομένων των $g, g^\alpha \in G$ (αντίστοιχα του $\beta \in \mathbb{N}$ δεδομένων των $g, g^\beta \in G$), καλείται το **πρόβλημα του διακριτού λογαρίθμου** και θεωρείται ως υπολογιστικά δύσκολο* να επιλυθεί.

*Δεν υπάρχει κάποια απόδειξη του δυσεπίλυτου του προβλήματος, αλλά διέπεται από μια ισχυρή πεποίθηση πως είναι δύσκολα επιλύσιμο.

Κεφάλαιο 4

Κρυπταλγόριθμοι τμήματος

Οι αλγόριθμοι τμήματος ανήκουν στους συμμετρικούς αλγορίθμους, δηλαδή το ίδιο κλειδί χρησιμοποιείται κατά την κρυπτογράφηση και αποκρυπτογράφηση ενός μηνύματος. Το μήνυμα διαμερίζεται σε ισομήκη τμήματα συνεχόμενων χαρακτήρων με το κάθε ένα να επεξεργάζεται από τον αλγόριθμο. Η εν λόγω ιδιότητα καθιστά εφικτή την παράλληλη προσπέλαση των τμημάτων (ή συλλογής τμημάτων) του μηνύματος καθόσον ο κρυπταλγόριθμος επιδράει σε τμηματικό επίπεδο.

Οι αλγόριθμοι τμήματος, επιθυμούν το απλό κείμενο το οποίο τους παράσχεται για κρυπτογράφηση να έχει μήκος πολλαπλάσιο του μήκους του κλειδιού. Για τις περιπτώσεις όπου αυτό δεν ισχύει, χρησιμοποιείται η τεχνική του παραγεμίσματος (padding), όπου το απλό κείμενο επαυξάνεται με την προσθήκη μιας ακολουθιας από bytes στο τέλος του. Ένα παράδειγμα παραγεμίσματος αποτελεί το πρότυπο PKCS5 [RfC2315].

Σύμβαση. *Ως* $s \ll b$ *συμβολίζεται η αριστερή ολίσθηση της ακολουθίας* $s \in \{0,1\}^*$ *κατά* $b \in \mathbb{N}$ *δυφία. Επί παραδείγματι* $(1,1,0,0,0) \ll 1 = (1,1,0,0,0,0)$.

Σύμβαση. *Ως* $s \lll b$ *συμβολίζεται η αριστερή κυκλική ολίσθηση της ακολουθίας* $s \in \{0,1\}^*$ *κατά* $b \in \mathbb{N}$ *δυφία. Επί παραδείγματι* $(1,1,0,0,0) \lll 1 = (1,0,0,0,1)$.

Σύμβαση. *Έστω* $\alpha \in \{0,1\}^{\ell_1}$ *και* $\beta \in \{0,1\}^{\ell_2}$. *Ορίζεται* $\alpha \parallel \beta := (\alpha \ll \ell_2) + \beta$, *όπου ως* $\ll$ *νοείται η αριστερή ολίσθηση της ακολουθίας.*

Σύμβαση. *Οι κρυπταλγόριθμοι πραγματεύονται τις δυαδικές αναπαστάσεις των απλών κειμένων και των κρυπτοκειμένων (μια τέτοια αναπαράσταση μπορεί να προκύψει αντιστοιχόντας κάθε γράμμα του κειμένου στο ASCII κωδικό του).*

Το 2001 το NIST (National Institute for Standards and Technology) ανακύρηξε [FIPS197] τον νικητή του διαγωνισμού AES (Advanced Encryption Standard) ο οποίος αποσκοπουσε στην ανάδειξη ενός νέου κρυπτογραφικού πρωτοκόλλου. Το νέο (προηγμένο) κρυπτογραφικό πρωτκόλλο θα αντικαθιστούσε το παλαιότερο, τον DES (Data Encryption Standard) [FIPS46-3] το οποίο είχε καθιερωθεί από τον Νοέμβριο του 1976. Ο DES αποτελεί αλγόριθμο τμήματος, ο οποίος

- χρησιμοποιεί ένα δοθέν κλειδί 56 δυφιών,
- επεξεργάζεται τμήματα μήκους 64 δυφίων,
- εμπεριέχει 8 κουτιά αντικατάστασης (Sboxes), 5 πίνακες μεταθέσεων ($\mathrm{IP}, \mathrm{IP}^{-1}, E, P, \mathrm{PC\text{-}1}, \mathrm{PC\text{-}2}$)
- υλοποιείται σε 16 γύρους, όπου ο κάθε ένας εξ αυτών αποτελεί ένα δίκτυο Feistel [MOV01].

Το επίπεδο ασφαλείας του DES αποτελούσε αμφισβητούμενο σχεδόν από τη δημοσίευσή του [DH77]. Σε μια προσπάθεια να υπερκεραστούν τα κενά στην ασφάλεια του DES, αυξήθηκε το μήκος του κλειδιού το οποίο χρησιμοποιούσε. Επίσης, εισήχθει η τριπλή εκδοχή του αλγορίθμου [RfC1851]: $\mathcal{E}_{K_3}\Big(\mathcal{D}_{K_2}\big(\mathcal{E}_{K_1}(M)\big)\Big)$. Ωστόσο, το NIST οδηγήθηκε στην ανακύρηξη της διεξαγωγής του διαγωνισμού AES το 1997 [AES].

4.1 AES (Rijndael)

Ο αλγόριθμος Rijndael [RD99], το οποίο προκύπτει από τον συνδυασμό των εμπευστών του: Vincent Rijmen και Joan Daemen, κέρδισε τον διαγωνισμό AES καθιστώντας τον πρότυπο κρυπτογράφησης [FIPS197]. Το εν λόγο γεγονός αποτέλεσε στη χρήση του ονόματος του αλγορίθμου ως AES, αντί του αρχικού του. Το κάθε τμήμα προς κρυπτογράφηση αναπαρίσταται ως ένας $4 \times N_b$ πίνακας λέξεων μήκους 8 δυφίων εκάστη. Το μήκος του κλειδιού ποικίλει διαμορφώνοντας το πλήθος των γύρων του αλγορίθμου. Κάθε block του (απλού/κρυπτο) κειμένου καλείται **state**. Τα τεχνικά χαρακτηριστηκά του Rijndael είναι τα εξής:

- Μήκος τμήματος: 128 δυφία, συνεπώς προκύπτει $N_b = \dfrac{128}{4 \times 8} = 4$.
- Μήκος κλειδιού: $N_k \times 32$ δυφία για $N_b \in \{4, 6, 8\}$ (δηλαδή 128, 192 ή 256 δυφία αντίστοιχα).
- Πλήθος γύρων: $N_r = 10, 12, 14$ οι οποίοι αντιστοιχούν στο κάθε μήκος κλειδιού παραπάνω.

- 1 κουτί αντικατάστασης (Sbox) εισόδου 8 δυφίων και εξόδου 8 δυφίων.

4.1.1 Κρυπτογράφηση

Παρατίθεται η υλοποίηση του αλγορίθμου κρυπτογράφησης του Rijndael.

Algorithm 1 Το κρυπτοσύστημα Rijndael (κρυπτογράφηση)

Είσοδος: Μήνυμα $M \in \{0,1\}^{8\times\ell}$ όπου το ℓ είναι πολλαπλάσιο του 16 και το αρχικό κλειδί $K_0 \in \{0,1\}^{4\times N_b\times 8}$.
΄Εξοδος: Κρυπτοκείμενο $C \in \{0,1\}^{8\times\ell}$.
1: $(K_1,\ldots,K_{N_r}) \leftarrow \texttt{KeyExpantion}(K_0)$ // Δημιουργία κλειδιού του κάθε γύρου
2: $C \leftarrow$ ''
3: **για κάθε** (state: τμήμα του M μήκους 128 δυφίων) **επανάλαβε**
4: state $\leftarrow \texttt{AddRoundKey}(\text{state}, K_0)$
5: **για** $(i = 1, 2, \ldots, N_r)$ **επανάλαβε**
6: state $\leftarrow \texttt{SubBytes}(\text{state})$
7: state $\leftarrow \texttt{ShiftRows}(\text{state})$
8: **εάν** $(i \neq N_r)$ **τότε**
9: state $\leftarrow \texttt{MixColumns}_{\alpha}(\text{state})$
10: **τέλος εάν**
11: state $\leftarrow \texttt{AddRoundKey}(\text{state}, K_i)$
12: **τέλος για**
13: $C \leftarrow C \parallel$ state
14: **τέλος για**
15: **επίστρεψε** C

Συνεπώς, κατά τον τελευταίο γύρο του αλγορίθμου δεν εφαρμόζεται το ανακάτεμα στηλών `MixColumns`.

Βασικές λειτουργίες γύρων

Η απεικόνιση

$$\texttt{AddRoundKey} : \mathbb{M}_{4\times N_b}(\{0,1\}^8) \times \mathbb{M}_{4\times N_b}(\{0,1\}^8) \longrightarrow \mathbb{M}_{4\times N_b}(\{0,1\}^8)$$
$$\left((b_{ij})_{\substack{i=1,\ldots,4\\ j=1,\ldots,N_b}}, (k_{ij})_{\substack{i=1,\ldots,4\\ j=1,\ldots,N_b}}\right) \longmapsto (b_{ij}\oplus k_{ij})_{\substack{i=1,\ldots,4\\ j=1,\ldots,N_b}}$$

εφαρμόζει την πράξη της αποκλειστικής διάζευξης (xor) $x \oplus y := \begin{cases} 1, & \text{αν } x \neq y \\ 0, & \text{αλλιώς} \end{cases}$ δυφίο προς δυφίο για κάθε λέξη b_{ij} του τμήματος με την αντίστοιχη λέξη k_{ij} από το κλειδί, για $i = 1, \ldots, 4$ και $j = 1, 2, \ldots, N_b$.

Ο μετασχηματισμός

$$\begin{aligned} \texttt{SubBytes} : \mathbb{M}_{4\times N_b}(\{0,1\}^8) &\longrightarrow \mathbb{M}_{4\times N_b}(\{0,1\}^8) \\ (b_{ij})_{\substack{i=1,\ldots,4 \\ j=1,\ldots,N_b}} &\longmapsto \big(Sbox(b_{ij})\big)_{\substack{i=1,\ldots,4 \\ j=1,\ldots,N_b}} \end{aligned}$$

συνιστά στην εφαρμογή ενός S-box σε κάθε λέξη του τμήματος. Κατά το πρότυπο [RD99] που προτάθηκε έχει ορισθεί ένας $S = (s_{ij})_{\substack{i=0,\ldots,15 \\ j=0,\ldots,15}} \in \mathbb{M}_{16\times 16}\big(\{0,1\}^8\big)$. Έστω $b_{ij} = (x_{ij}^1, x_{ij}^2, \ldots, x_{ij}^8)_2$ η δυαδική αναπαράσταση της λέξης b_{ij}, για κάποια $i = 1, \ldots, N_k$ και $j = 1, \ldots, N_b$, του τμήματος το οποίο επεξεργάζεται ο αλγόριθμος, τότε $\{0,1\}^8 \ni b_{ij} \longmapsto s_{\alpha\beta} \in \{0,1\}^8$, όπου

- α είναι η δεκαδική αναπαράσταση του δυαδικού αριθμού $(x_{ij}^1, x_{ij}^2, x_{ij}^3, x_{ij}^4)_2$,
- β είναι η δεκαδική αναπαράσταση του δυαδικού αριθμού $(x_{ij}^5, x_{ij}^6, x_{ij}^7, x_{ij}^8)_2$.

Ο μετασχηματισμός

$$\begin{aligned} \texttt{ShiftRows} : \mathbb{M}_{4\times N_b}(\{0,1\}^8) &\longrightarrow \mathbb{M}_{4\times N_b}(\{0,1\}^8) \\ (b_{ij})_{\substack{i=1,\ldots,4 \\ j=1,\ldots,N_b}} &\longmapsto \big(b_{i,1+(4+j-i \mod 4)}\big)_{\substack{i=1,\ldots,4 \\ j=1,\ldots,N_b}} \end{aligned}$$

Κάθε γραμμή του τμήματος θεωρείται ως δυαδικός αριθμός: $(b_{i,1}, b_{i,2}, b_{i,3}, b_{i,4})_2$. Κατόπιν εφαρμόζεται αριστερή κυκλική ολίσθηση κατά $2^{8\cdot(i-1)}$ δυφία (bits), για τις γραμμές $i = 1, 2, 3, 4$. Άρα:

- Για τη γραμμή 1: $(b_{1,1}, b_{1,2}, b_{1,3}, b_{1,4}) \longmapsto (b_{1,1}, b_{1,2}, b_{1,3}, b_{1,4})$ (αμετάβλητη).
- Για τη γραμμή 2: $(b_{2,1}, b_{2,2}, b_{2,3}, b_{2,4}) \longmapsto (b_{2,2}, b_{2,3}, b_{2,4}, b_{2,1})$.
- Για τη γραμμή 3: $(b_{3,1}, b_{3,2}, b_{3,3}, b_{3,4}) \longmapsto (b_{3,3}, b_{3,4}, b_{3,1}, b_{3,2})$.
- Για τη γραμμή 4: $(b_{4,1}, b_{4,2}, b_{4,3}, b_{4,4}) \longmapsto (b_{4,4}, b_{4,1}, b_{4,2}, b_{4,3})$.

Για το ανακάτεμα των στηλών, θεωρείται το πολυώνυμο $3x^3+x^2+x+2$. Ο μετασχηματισμός συνίσταται στην $\texttt{MixColumns}_\alpha : \mathbb{M}_{4\times N_b}(\{0,1\}^8) \longrightarrow \mathbb{M}_{4\times N_b}(\{0,1\}^8)$, όπου

$$(b_{ij})_{\substack{i=1,\ldots,4 \\ j=1,\ldots,N_b}} \longmapsto \Big[(3x^3+x^2+x+2) \bullet p\big(b_{1j}x^3+b_{2j}x^2+b_{3j}x+b_{4j}\big) \mod (x^4+1, 2^8)\Big]_{j=1,\ldots,N_b}$$

όπου ως $\bullet$ συμβολίζεται ο πολλαπλασιασμός πολυωνύμων κι ως $\mod(x^4+1, 2^8)$ το υπόλοιπο της διαίρεσης των πολυωνύμων του αποτελέσματος με το x^4+1 με συντελεστές στο $\mathrm{GF}(2^8)$ (δηλαδή modulo 2^8).

4.1.2 Παραγωγή κλειδιών

Το αρχικό κλειδί $K_0 = (k^0_{ij})_{\substack{i=1,\dots,4\\ j=1,\dots,N_b}} \in \mathbb{M}_{4\times N_b}\big(\{0,1\}^8\big)$ θεωρείται ως μέρος της εισόδου του αλγορίθμου. Για τη δημιουργία του κλειδιού

$$K_\alpha = (k^\alpha_{ij}) \in \mathbb{M}_{4\times N_b}\big(\{0,1\}^8\big), \qquad \alpha = 1, 2, \dots, N_k$$

το οποίο πρόκειται να χρησιμοποιηθεί κατά τον i-οστό γύρο πραγματοποιούνται οι εξής υπολογισμοί:

$$K_\alpha := \begin{bmatrix} \mathrm{RC}(\mathrm{Sbox}(k^{\alpha-1}_{2,N_k}) \oplus k^{\alpha-1}_{1,1}) & \cdots & \mathrm{RC}(k^{\alpha-1}_{1,\beta} \oplus k^{\alpha}_{1,\beta-1}) & \cdots & \mathrm{RC}(k^{\alpha-1}_{1,N_k} \oplus k^{\alpha}_{1,N_k-1}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{RC}(\mathrm{Sbox}(k^{\alpha-1}_{N_b,N_k}) \oplus k^{\alpha-1}_{N_b-1,1}) & \cdots & \mathrm{RC}(k^{\alpha-1}_{N_b-1,\beta} \oplus k^{\alpha}_{N_b-1,\beta-1}) & \cdots & \mathrm{RC}(k^{\alpha-1}_{N_b-1,N_k} \oplus k^{\alpha}_{N_b-1,N_k-1}) \\ \mathrm{RC}(\mathrm{Sbox}(k^{\alpha-1}_{1,N_k}) \oplus k^{\alpha-1}_{N_b,1}) & \cdots & \mathrm{RC}(k^{\alpha-1}_{N_b,\beta} \oplus k^{\alpha}_{N_b,\beta-1}) & \cdots & \mathrm{RC}(k^{\alpha-1}_{N_b,N_k} \oplus k^{\alpha}_{N_b,N_k-1}) \end{bmatrix}$$

Άρα, για $\alpha, \beta > 1$, η β-οστή στήλη του κλειδιού $K_\alpha \in \mathbb{M}_{4\times N_b}\big(\{0,1\}^8\big)$ προκύπτει ως η αποκλειστική διάζευξη της β-οστής στήλης από το κλειδί K_α με την $(\beta-1)$-οστή στήλη του τρέχοντος κλειδιού K_α. Παραπάνω, ο τελεστής SBox υπολογίζεται όπως αναγράφεται στην πράξη `SubBytes`. Τέλος, ο τελεστής RC (Round Constant - rcon όπως αναφέρεται στο [RD99]) επίσης εμπεριέχεται στην αρχική πρόταση του αλγορίθμου στο [RD99].

4.1.3 Αποκρυπτογράφηση

Η αποκρυπτογράφηση επιτυγχάνεται με την εφαρμογή των βημάτων κρυπτογράφησης σε αντίστροφη σειρά χρησιμοποιώντας τις αντίστροφες λειτουργίες από εκείνες της κρυπτογράφησης. Συνεπώς, ορίζονται:

`SubBytes`$^{-1}$(**state**) είναι η εφαρμογή του Sbox^{-1} (δηλαδή της αντίστροφης μετάθεσης της Sbox), όπως αυτό εμπεριέχεται στο πρότυπο [RD99].

`ShiftRows`$^{-1}$(**state**) είναι η δεξιά κυκλική ολίσθηση κατά $i-1$ λέξεις, όπου $i = 1, 2, 3, 4$ είναι ο δείκτης της γραμμής που συμβαίνει η ολίσθηση.

$$(b_{ij})_{\substack{i=1,\dots,4\\ j=1,\dots,N_b}} \longmapsto (b_{i,1+((j+i-1) \mod 4)})_{\substack{i=1,\dots,4\\ j=1,\dots,N_b}}$$

`MixColumns`$_{\alpha^{-1}}$(**state**) είναι η εφαρμογή της διαδικασίας του ανακατέματος στηλών με χρήση του αντιστρόφου πολυωνύμου α, το οποίο έγκειται στο $\alpha^{-1} = 11x^3 + 13x^2 + 9x + 14$.

Τελικώς, ο αλγόριθμος αποκρυπτογράφησης συντίθεται από την εξής αλληλουχία εντολών:

Algorithm 2 Το κρυπτοσύστημα Rijndael (αποκρυπτογράφηση)

Είσοδος: Κρυπτοκείμενο $C \in \{0,1\}^{8\times\ell}$ όπου το ℓ είναι πολλαπλάσιο του 16 και το αρχικό κλειδί $K_0 \in \{0,1\}^{4\times N_b\times 8}$.
Είσοδος: Μήνυμα $M \in \{0,1\}^{8\times\ell}$.
1: $(K_1, \ldots, K_{N_r}) \leftarrow \texttt{KeyExpantion}(K_0)$ // Δημιουργία κλειδιού του κάθε γύρου
2: $M \leftarrow \{\}$
3: **για κάθε** (state: τμήμα του C μήκους 128 δυφίων) **επανάλαβε**
4: state $\leftarrow \texttt{AddRoundKey}(\text{state}, K_{N_r})$
5: **για** $(i = 1, 2, \ldots, N_r)$ **επανάλαβε**
6: state $\leftarrow \texttt{ShiftRows}^{-1}(\text{state})$
7: state $\leftarrow \texttt{SubBytes}^{-1}(\text{state})$
8: state $\leftarrow \texttt{AddRoundKey}(\text{state}, K_{N_r-i})$
9: **εάν** $(i \neq N_r)$ **τότε**
10: state $\leftarrow \texttt{MixColumns}_{\alpha^{-1}}(\text{state})$
11: **τέλος εάν**
12: **τέλος για**
13: $M \leftarrow M \cup \text{state}$
14: **τέλος για**
15: **επίστρεψε** M

4.2 Serpent

Ο Serpent [ABK98] συμπεριλαμβάνεται στους 5 τελικούς αλγορίθμους του διαγωνισμού για το AES. Δημιουργοί του Serpent αποτέλεσαν οι Ross Anderson, Eli Biham, και Lars Knudsen. Το κρυπτοσύστημα διατίθεται σε δύο εκδοχές:

1. Εφαρμογή γραμμικών μετασχηματισμών στα δυφία με χρήση πινάκων (οι οποίοι περιέχονται στο [ABK98]).
2. Εφαρμογή κυκλικών ολισθήσεων στις λέξεις του τμήματος.

Τα τεχνικά χαρακτηριστηκά του κρυπτοσυστήματος είναι τα εξής:

- Μήκος τμήματος: 128 δυφία.
- Μήκος κλειδιού: 128, 192 ή 256 δυφία.

- Πλήθος γύρων: 32.
- 8 κουτιά αντικατάστασης $\mathcal{S}_0, \ldots, \mathcal{S}_7$ εισόδου 4 δυφίων και εξόδου 4 δυφίων.

  [Η αρχική εκδοχή του κρυπτοσυστήματος [ABK98] (Serpent-0) περιείχε 32 Sboxes. Κατά τη συμμετοχή στον διαγωνισμό του AES, υποβλήθηκε η (παρούσα) εκδοχή με 8 Sboxes (Serpent-1).]

4.2.1 Κρυπτογράφηση

Εφαρμογή γραμμικών μετασχηματισμών

Ο αλγόριθμος συνίσταται στον ακόλουθο:

Algorithm 3 Το κρυπτοσύστημα Serpent (κρυπτογράφηση με χρήση πινάκων)

Είσοδος: Μήνυμα $M \in \{0,1\}^{8\times\ell}$ όπου το ℓ είναι πολλαπλάσιο του 16 και το αρχικό κλειδί $K \in \{0,1\}^{4\times j\times 8}$, για $j = 4, 6, 8$.
΄Εξοδος: Κρυπτοκείμενο $C \in \{0,1\}^{8\times\ell}$.

$(K_0, \ldots, K_3 2) \leftarrow \texttt{KeyExpantion}(K)$
$C \leftarrow$ ''
για κάθε (P: τμήμα του M μήκους 128 δυφίων) **επανάλαβε**
  $\hat{B}_0 \leftarrow \texttt{IP}(P)$
  για ($i = 0, 1, \ldots, 30$) **επανάλαβε**
    $\hat{B}_{i+1} \leftarrow L\big(\mathcal{S}_{i \bmod 8}(\hat{B}_i \oplus \text{IP}(K_i))\big)$
  τέλος για
  $\hat{B}_{32} \leftarrow \mathcal{S}_7(\hat{B}_{31} \oplus \texttt{IP}(K_{31})) \oplus \texttt{IP}(K_{32})$
  $\hat{B} \leftarrow \texttt{FP}(\hat{B}_{32})$
  $C \leftarrow C \parallel \hat{B}$
τέλος για
επίστρεψε C

Για την παρούσα εκδοχή επιπλέον χρειάζονται (και παρέχονται στο [ABK98]):

- γραμμικός μετασχηματισμός L
- 2 πίνακες μεταθέσεων, οι IP (Initial Permutation) και FP (Final Permutation).

τα οποία υλοποιούνται με χρήση πινάκων. Οι 31 γύροι του Αλγορίθμου 3 χρησιμοποιούν τον γραμμικό μετασχηματισμό $L(\cdot)$ και κατά τον 32ο γύρο ο L αντικαθίσταται από την πράξη της αποκλειστική διάζευξης με το τελευταίο μέρος του κλειδιού.

Εφαρμογή κυκλικών ολισθήσεων

Πρωτίστως, ορίζεται η βοηθητική συνάρτηση `SBitSlice` [St98].

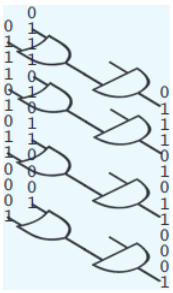
Algorithm 4 Η βοηθητική συνάρτηση SBitSlice

Είσοδος: Τον δείκτη i του γύρου του Serpent και 4 ακολουθίες Y_0, Y_1, Y_2, Y_3 μήκους 32 δυφίων εκάστη.
΄Εξοδος: Μία ακολουθία $X \in \{0,1\}^{4\times 32}$ 4 λέξεων 32 δυφίων εκάστη.

$X \leftarrow \{\text{‘’}, \text{‘’}, \text{‘’}, \text{‘’}\}$
για $(j = 0, 1, \ldots, 31)$ **επανάλαβε**
  $\text{quad} \leftarrow \mathcal{S}_{i \bmod 8}\big((Y_{0,j}, Y_{1,j}, Y_{2,j}, Y_{3,j})\big)$ // Y_{ij} είναι το j-οστό δυφίο της Y_i
  για $(k = 0, 1, 2, 3)$ **επανάλαβε**
    $X_k \leftarrow X_k \parallel \text{quad}_k$ // quad_k είναι το k-οστό δυφίο της quad
  τέλος για
τέλος για
επίστρεψε X

Παραπάνω, η απαρίθμηση εκκινεί από τον δείκτη 0. Ως $(Y_{0,j}, Y_{1,j}, Y_{2,j}, Y_{3,j})$ νοείται η δυαδική ακολουθία η οποία έχει ως δυφία τα $Y_{0,j}, Y_{1,j}, Y_{2,j}$ και $Y_{3,j}$. Η εφαρμογή του κουτιού αντικατάστασης $\mathcal{S}_i$, $i = 0, \ldots, 7$, θα δώσει ως αποτέλεσμα την δυαδική ακολουθία $\text{quad} \in \{0,1\}^4$.

Η παρούσα υλοποίηση του Serpent προσφέρει ένα επιπλέον επίπεδο παράλληλων υπολογισμών (πέραν του να υπολογισθούν τα τμήματά του παράλληλα) στο επίπεδο του κάθε επεξεργαστή ξεχωριστά. Στην `SBitSlice` οι 32 επαναλήψεις (του δείκτη j) μπορούν να αντικατασταθούν από παράλληλους υπολογισμούς του κάθε j. Η παράλληλη εκδοχή απαιτεί τα κουτιά αντικατάστασης να έχουν υλοποιηθεί ως λογικές (boolean) συναρτήσεις κι όχι ως πίνακες (όπως παρέχονται στο [ABK98]). Επομένως, τα κουτιά αντικαταστάσεων πραγματεύονται λογικές πράξεις ανάμεσα σε δυφία κι όχι μετασχηματισμούς λέξεων. Τα αποτελέσματα των πράξεων συγκολλώνται σε κατακόρυφη μορφή, όπως φαίνεται στο Σχήμα 4.1. Το πλήθος των παράλληλων πράξεων ανά επεξεργαστή ισούται με το μέγεθος της Αριθμητικής και Λογικής Μονάδας (Arithmetic and Logic Unit - ALU) του κάθε επεξεργαστή. Κατά συνέπεια, ένας επεξεργαστής 64-bit, μπορεί να υλοποιήσει ένα στιγμιότυπο της `SBitSlice` σε δύο βήματα παράλληλων υπολογισμών, αφού η συνάρτηση πραγματεύεται 4 λέξεις 32 δυφίων



Σχήμα 4.1: Η αναπαράσταση της παράλληλης υλοποίησης της συνάρτησης `SBitSlice` [St98]

έκαστη. Άρα, 2 λέξεις των 32 δυφίων μπορούν να υλοποιηθούν παράλληλα στον επεξεργαστή σε κάθε βήμα.

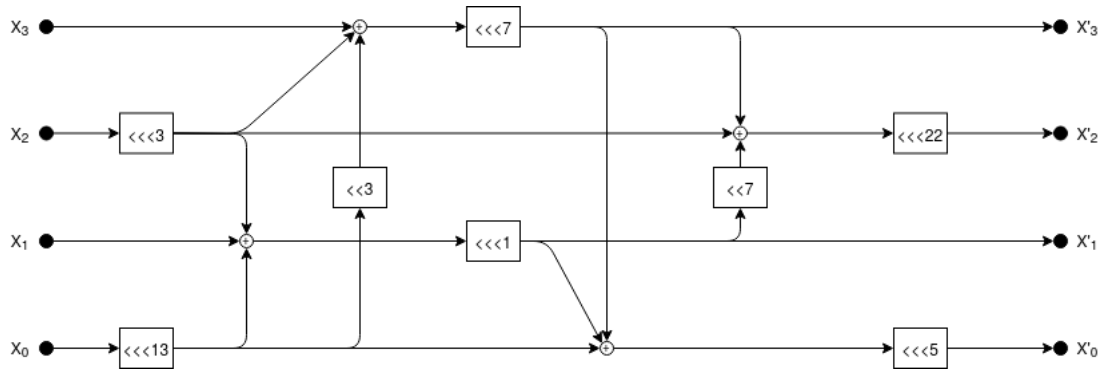
Η μεθοδολογία της παρούσας εκδοχής του αλγορίθμου συνίσταται στην εξής:

Algorithm 5 Το κρυπτοσύστημα Serpent (κρυπτογράφηση με χρήση ολισθήσεων)

Είσοδος: Μήνυμα $M \in \{0,1\}^{8\times\ell}$ όπου το ℓ είναι πολλαπλάσιο του 16 και το αρχικό κλειδί $K \in \{0,1\}^{4\times j\times 8}$, για $j = 4, 6, 8$.
Έξοδος: Κρυπτοκείμενο $C \in \{0,1\}^{8\times\ell}$.

$(K_0, \ldots, K_3 2) \leftarrow \texttt{KeyExpantion}(K)$
$C \leftarrow$ ''
για κάθε (P: τμήμα του M μήκους 128 δυφίων) **επανάλαβε**
  $B_0 \leftarrow P$
  για ($i = 0, 1, \ldots, 31$) **επανάλαβε**
    $Y_0, Y_1, Y_2, Y_3 \leftarrow B_i \oplus K_i$ // 4 λέξεις μήκος 32 δυφίων εκάστη
    $X_0, X_1, X_2, X_3 \leftarrow \texttt{SBitSlice}\big(i, (Y_0, Y_1, Y_2, Y_3)\big)$
    $X'_0, X'_1, X'_2, X'_3 \leftarrow \texttt{Transform}\big(X_0, X_1, X_2, X_3\big)$
    $B_{i+1} \leftarrow X'_0 \parallel X'_1 \parallel X'_2 \parallel X'_3$
  τέλος για
  $B \leftarrow \mathcal{S}_7(B_{31} \oplus K_{31}) \oplus K_{32}$
  $C \leftarrow C \parallel B$
τέλος για
επίστρεψε C

Η απεικόνιση $\text{Transform}(X_0, X_1, X_2, X_3) = (X'_0, X'_1, X'_2, X'_3)$, με $X_i, X'_i \in \{0,1\}^{32}$, $i = 1, 2, 3, 4$, υλοποιείται από το κάτωθι διάγραμμα:



όπου παραπάνω ως $\oplus$ συμβολίζεται η αποκλειστική διάζευξη (δυφίο προς δυφίο) δυαδικών ακολουθιών.

4.2.2 Παραγωγή κλειδιών

Παρακάτω παρατίθεται ο αλγόριθμος παραγωγής κλειδιού `KeyExpantion` για δοθέν αρχικό κλειδί K, μήκους 256 δυφίων.

Αναλύεται το $K \in \{0,1\}^{256}$ σε 8 λέξεις 32 (διαδοχικών) δυφίων εκάστη:

$$(w_{-8}, w_{-7}, \ldots, w_{-1}) \leftarrow K$$

Κατόπιν υπολογίζονται οι λέξεις:

$$w_i := (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) \lll 11 \in \{0,1\}^{32}, \qquad \text{για } i = 0, 1, \ldots, 131$$

όπου $\phi = \texttt{9e3779b9}_{16}$ είναι η δεκαεξαδική αναπαράσταση ενός μέρους της χρυσής τομής $\phi = \dfrac{\sqrt{5}+1}{2}$. Υστερα υπολογίζονται οι συνιστώσες του κάθε κλειδιού:

$$(k_i, k_{33+i}, k_{66+i}, k_{99+i}) := \mathcal{S}_{(3-i \pmod 8)}\big((w_i, w_{33+i}, w_{66+i}, w_{99+i})\big), \qquad \text{για } i = 0, 1, \ldots, 32$$

όπου κατά την πράξη $3 - i \pmod 8$ εννοούνται τα θετικά υπόλοιπα, δηλαδή για $i = 4 \implies 3 - 4 = -1 \equiv 7 \pmod 8$, για $i = 5 \implies 3 - 5 = -2 \equiv 6 \pmod 8$ και ούτο καθ' εξής. Τελικώς, το κλειδί το οποίο χρησιμοποιείται στον κάθε έναν από τους 32 γύρους του αλγορίθμου ορίζεται ως:

$$K_i := (k_{4\cdot i}, k_{4\cdot i+1}, k_{4\cdot i+2}, k_{4\cdot i+3}) \in \{0,1\}^{4\times 32}, \qquad \text{για } i = 0, 1, \ldots, 32$$

4.2.3 Αποκρυπτογράφηση

Η αποκρυπτογράφηση του Serpent επιτυγχάνεται υλοποιώντας τον Αλγόριθμο 3 ή 5 χρησιμοποιώντας τα κλειδιά των γύρων με την αντίστροφη σειρά, τα αντίστροφα κουτιά αντικαταστάσεων $\mathcal{S}_0^{-1}, \ldots, \mathcal{S}_7^{-1}$ και

- στον Αλγόριθμο 3 χρησιμοποιώντας την $L^{-1}(\cdot)$, αντί της $L(\cdot)$.
- στον Αλγόριθμο 5 χρησιμοποιώντας την $\text{Transform}^{-1}(\cdot)$, αντί της $\text{Transform}(\cdot)$.

4.2.4 Το κρυπτοσύστημα Serpent στην κρυπτογράφηση εικόνων

Ο Serpent εν αντιθέσει με τον Rinjdael υλοποιείται σε 32 γύρους, αντί 10. Η εν λόγω προτίμηση αυξάνει το επίπεδο ασφαλείας του αλγορίθμου, ωστόσο τον καθιστάει πιο χρονοβορό. Γι αυτό και ο Rinjdael επιλέχθηκε ως νικητής του διαγωνισμού AES όντας γρηγορότερα υλοποιήσιμος ιδιαίτερα σε μικρότερα κομμάτια πληροφορίας [SKWWFKS00]. Παρ' όλο που δεν επελέγει ως πρότυπο, το κρυπτοσύστημα Serpent χρησιμοποιείται -εκτός των άλλων- και στην κρυπτογράφηση εικόνων [Iz15], [AR16].

Μια παραλλαγή [SHF18] του Serpent, η οποία χρησιμοποιεί τον δακτύλιο πολυωνύμων* $R_8 := {}^{\mathbb{F}_2[x]}\!/_{\langle x^8 \rangle}$, τον καθιστάει γρηγορότερα υλοποιήσιμο. Στο [SHF20] η εν λόγω παραλλαγή χρησιμοποιείται για την ξεχωριστή κρυπτογράφηση των τριών επιπέδων μιας έγχρωμης εικόνας (με το κάθε επίπεδο να αντιστοιχεί στο κάθε ένα από τα βασικά χρώματα: κόκκινο, μπλε, πράσινο). Θεωρώντας το κάθε χρωματικό επίπεδο ως ανεξάρτητη εικόνα, θα ήταν δυνατόν να επεκταθεί η υλοποίηση του Κεφαλαίου 7 ώστε σε παράλληλο χρόνο να κρυπτογραφείται το κάθε χρωματικό επίπεδο και εν συνεχεία να συγκεντρώνονται και να τοποθετούνται στην κατάλληλη θέση τους. Ωστόσο, η προκείμενη θεώρηση θα αύξανε τον χρόνο επεξεργασίας της εκάστοτε εικόνας, η οποία θα θεωρούνταν ως 3 ξεχωριστές εικόνες.

4.3 Τρόποι τμηματικής προσπέλασης μηνυμάτων

Έστω P το απλό κείμενο, το οποίο είναι επιθυμητό να κρυπτογραφηθεί κι έστω

$$P = P_1 \parallel P_2 \parallel \cdots \parallel P_n$$

η διαμέρισή του σε ισομήκη συστοιχίες συνεχόμενων χαρακτήρων, με εξαίρεση ίσως το μήκος της τελευταίας συστοιχίας, το οποίο μπορεί να είναι μικρότερο. Ο αλγόριθμος μπορεί να επιδράσει στα τμήματα του απλού κειμένου και να παράξει το κρυπτοκείμενο

$$C = C_1 \parallel C_2 \parallel \cdots \parallel C_n$$

αλλά και αντίστροφα, σύμφωνα με τους παρακάτω τρόπους.

*Ο δακτύλιος R_8 αποτελείται από τα πολυώνυμα τα οποία αποτελούν το υπόλοιπο της διαίρεσης ενός πολυωνύμου με συντελεστές από το $\mathbb{Z}_2 = \{0, 1\}$, με το πολυώνυμο x^8. Ο R_8 διέπεται από τις πράξεις της πρόσθεσης και του πολλαπλασιασμού πολυωνύμων, διαιρώντας το παραχθέν πολυώνυμο με το x^8 και αναπροσαρμόζοντας τους συντελεστές του στο $\mathbb{Z}_2$.

Ως $\mathcal{E}_K(\cdot), \mathcal{D}_K(\cdot)$ θεωρούνται η συναρτήση κρυπτογράφησης/αποκρυπτογράφησης ενός αλγορίθμου τμήματος η οποία χρησιμοποιεί το κλειδί K. Το αρχικό διάνυσμα IV, όπου εμφανίζεται, δεν είναι απαραίτητο να διατηρείται μυστικό, ωστόσο δεν θα πρέπει να είναι καταφανής ο τρόπος δημιουργίας του.

Electronic Codebook (ECB): Εφαρμόζεται η συνάρτηση κρυπτογράφησης σε κάθε τμήμα του απλού κειμένου ξεχωριστά:

$$C_i = \mathcal{E}_K(P_i) \qquad\qquad P_i = \mathcal{D}_K(C_i)$$

Cipher Block Chaining (CBC): Κάθε τμήμα του κρυπτοκειμένου που δημιουργείται χρησιμοποιείται στην παραγωγή του επόμενου τμήματος του κρυπτοκειμένου ως εξής: Το παραχθέν τμήμα του κρυπτοκειμένου συνδυάζεται με το τρέχον τμήμα τους απλού κειμένου με την πράξη της λογικής διάζευξης και το αποτέλεσμα κρυπτογραφείται για να παράξει το τρέχον τμήμα του κρυπτοκειμένου.

$$C_i = \begin{cases} \mathcal{E}_K(P_1 \oplus \mathrm{IV}), & \text{αν } i = 1 \\ \mathcal{E}_K(P_i \oplus C_{i-1}), & \text{αλλιώς} \end{cases} \qquad P_i = \begin{cases} \mathcal{D}_K(C_1) \oplus \mathrm{IV}, & \text{αν } i = 1 \\ \mathcal{D}_K(C_i) \oplus C_{i-1}, & \text{αλλιώς} \end{cases}$$

Cipher Feedback (CFB): Κατά την κρυπτογράφηση, κάθε τμήμα κρυπτοκειμένου το οποίο παράγεται αναμειγνύεται (μέσω της αποκλειστικής διάζευξης) με το επερχόμενο τμήμα απλού κειμένου.

$$C_i = \begin{cases} \mathrm{IV}, & \text{εάν } i = 0 \\ \mathcal{E}_K(C_{i-1}) \oplus P_i, & \text{αλλιώς} \end{cases} \qquad P_i = \mathcal{E}_K(C_{i-1}) \oplus C_i$$

Η παραπάνω υλοποίηση καλείται *Full block CFB*. Υπάρχει δυνατότητα να χρησιμοποιούνται τα $s \in \mathbb{N}$ (για δοθέν s) περισσότερο σημαντικά δυφία του (κρυπτο)κειμένου. Το πλήθος s ονοματίζει και τον τρόπο προσπέλασης, όπως: 1-bit CFB mode, 8-bit CFB mode, 64-bit CFB mode, ή 128-bit CFB mode.

Output Feedback (OFB): Ορίζεται η (αναδρομική) ακολουθία:

$$O_i = \begin{cases} \mathcal{E}_K(\mathrm{IV}), & \text{αν } i = 1 \\ \mathcal{E}_K(O_{i-1}), & \text{αλλιώς} \end{cases}$$

για κάποιο αρχικό διάνυσμα IV. Η κρυπτογράφηση και αποκρυπτογράφηση των τμημάτων γίνεται ως εξής:

$$C_i = P_i \oplus O_i \qquad\qquad P_i = C_i \oplus O_i$$

Εξαίρεση μπορεί να αποτελέσει το τελευταίο τμήμα (είτε του μηνύματος, ήτοι το P_n, είτε του κρυπτοκειμένου, δηλαδή το C_n), το οποίο αν έχει μήκος, έστω l, μικρότερο από τα υπόλοιπα, τότε χρησιμοποιούνται τα l περισσότερο σημαντικά δυφία του O_n. Κατά τον OFB τρόπο, το IV θα πρέπει να αλλάζει κάθε φορά που χρησιμοποιείται ο αλγόριθμος.

Counter (CTR): Ορίζεται η ακολουθία:

$$O_i = \mathcal{E}_K(T_i)$$

για τα διανύσματα $T_1, T_2, \ldots, T_n$, όπου

$$T_i = \text{Nonce} \parallel \text{ctr}_i \qquad\qquad i = 1, 2, \ldots, n$$

Κάθε T_i, $i = 1, \ldots, n$ έχει το ίδιο πλήθος δυφίων όσα και ένα τμήμα του (κρυπτο)κειμένου, έστω $b \in \mathbb{N}$. Επιλέγεται ένα $s \leq b$. Το T_i, $i = 1, \ldots, n$ διαμερίζεται

- στο διάνυσμα Nonce $\in \{0,1\}^{b-s}$ (το οποίο είναι ισοδύναμο με την έννοια του IV) και
- στο διάνυσμα $\text{ctr}_i \in \{0,1\}^s$, όπου $\text{ctr}_i = \begin{cases} 0 \cdots 0, & \text{εάν } i = 1 \\ \text{ctr}_{i-1} +_2 1, & \text{αλλιώς} \end{cases}$, όπου ως $+_2$ νοείται η πράξη της πρόσθεσης στο δυαδικό σύστημα. Άρα, αναλόγως της επιλογής του $s \in \mathbb{N}$, μπορεί να επέλθει υπερχείλιση και ο μετρητής (ctr) να επανεκκινήσει από το 0.

Η κρυπτογράφηση και αποκρυπτογράφηση επιτυγχάνεται ως εξής:

$$C_i = P_i \oplus O_i \qquad\qquad P_i = C_i \oplus O_i$$

Συνοψίζοντας τις μεθόδους από την άποψη των παράλληλων υπολογισμών:

	Μέθοδος:	ECB	CBC	CFB	OFB	CTR
Υποστηρίζει παράλληλη	κρυπτογράφηση:	✓	×	×	×	✓
	αποκρυπτογράφηση:	✓	✓	✓	×	✓

Τα παραπάνω αποτελέσματα αφορούν την (απο)κρυπτογράφηση ενός κειμένου (εικόνας). Στην υλοποίηση του Κεφαλαίου 7, η εικόνα διαμερίζεται και πραγματεύεται ως ξεχωριστές εικόνες οι οποίες επεξεργάζονται στον ίδιο χρόνο. Συνεπώς, προτίνεται πως μπορεί να χρησιμοποιηθεί κάθε μία από τις ανωτέρω μεθόδους στο κάθε τμήμα της εικόνας ξεχωριστά. Απαραίτητο είναι αμφότερες οι επικοινωνούσες πλευρές να είναι γνώστες των IV που χρησιμοποιήθηκαν καθώς και της συνάρτησης διαμερισμού της εικόνας.

4.4 Σύγχηση και διάχυση

Οι έννοιες της σύγχησης και της διάχυσης εισήχθησαν από τον Claude Shannon [Sh$_{45}$] με σκοπό τον χαρακτηρισμό ενός κρυπταλγορίθμου ως ασφαλούς (αξιόπιστο). Αρχικώς, γίνονται οι εξής παρατηρήσεις:

Πρωτίστως, μην έχοντας κάποιον εξειδικευμένο κανόνα επιλογής κλειδιού, κάθε δυαδική ακολουθία μπορεί δυνητικά να αποτελέσει κλειδί του αλγορίθμου. Τα κρυπτοσυστήματα, των §4.1 και §4.2, έχουν κοινές επιλογές για το μήκος του κλειδιού. Επιλέγοντας κλειδιά μήκους 256 δυφίων, ο χώρος των πιθανόν κλειδιών συνίσταται σε

$$|\mathcal{K}| = 2^{256} \text{ πιθανά κλειδιά}$$

Κάθε επίθεση ωμής βίας[†] (brute force attack) θα απαιτούσε χρόνο δυσανάλογο του περιεχομένου της πληροφορίας η οποία θα ανακτούνταν.

Δευτερευόντως, η πράξη της αποκλειστικής διάζευξης (exclusive or - xor) προτιμάται στα κρυπτοσυστήματα καθόσον έχει "ζυγισμένη" έξοδο, γεγονός το οποίο εκμαιεύεται από τον πίνακα αληθείας της πράξεως, όπου οι δύο δυνατές έξοδοι (0 ή 1, εφόσον αφορά δυφία) είναι ισοπίθανες.

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

4.4.1 Σύγχηση

Η σύγχηση (confusion) αποτελεί την ιδιότητα *απόκρυψης της σχέσης του κλειδιού με το μήνυμα*. Η έκφανσή της έγκειται στο γεγονός της συσχέτισης ένος δυφίου του απλού κειμένου με πολλά δυφία του κλειδιού. Η μερική αλλαγή στα δυφία ενός κλειδιού θα πρέπει να επηρεάζει αρκετά το κρυπτοκείμενο. Έτσι, δεν θα πρέπει το κρυπτοκείμενο να αποκαλύπτει στοιχεία για το κλειδί.

Η σύγχηση επιτυγχάνεται με χρήση μη-γραμμικών μετασχηματισμών, οι οποίοι θα πρέπει να είναι αρκετά περίπλοκες συσχετίσης ώστε να είναι δύσκολο να αντιστραφούν (δηλαδή να ανακτηθεί το απλό κείμενο, μελετώντας το κρυπτοκείμενο). Η ιδιότητα αφορά και την κατηγορία των αλγορίθμων τμήματος, αλλά και την κατηγορία των αλγορίθμων ροής.

[†]Δοκιμή ένα προς ένα των πιθανών κλειδιών, μέχρις ότου να προκύψει κάποιο κατανοητό μήνυμα. Σημειώνεται πως η μερική ανάκτηση του κλειδιού, δεν επάγει την μερική αποκρυπτογράφηση των δεδομένων.

4.4.2 Διάχυση

Η ιδιότητα της διάχυσης (diffusion) έγκειται στο γεγονός πως [St14] *η αλλαγή ενός δυφίου στο απλό κείμενο, επιφέρει πως (περί) τα μισά δυφία του κρυπτοκειμένου μεταβάλλονται.* Η ιδιότητα της διάχυσης αποσκοπεί στην αποσυσχέτιση του κρυπτοκειμένου από το απλό κείμενο. Η ιδιότητα αφορά τους αλγορίθμους τμήματος και τις συναρτήσεις κατακερματισμού.

Σκοπός της διάχυσης είναι η δημιουργία μιας εξόδου (εν προκειμένω του κρυπτοκειμένου) η οποία θα μοιάζει (όσο το δυνατόν περισσότερο) με μια ακολουθία τυχαίων χαρακτήρων. Η εν λόγω τυχαιότητα είναι φαινομενική καθ' όσον στην παραγματικότητα δεν πρόκειται για μια τυχαία ακολουθία, αλλά για μια ακολουθία συσχετιζόμενη με μια άλλη (συγκεκριμένα, με το απλό κείμενο και τούμπαλιν). Επιπλέον, οι ηλεκτρονικοί υπολογιστές δεν δύνανται να δημιουργήσουν τυχαιότητα, αλλά παράγουν ψευδοτυχαίους αριθμούς, οι οποίοι φαινομενικά μοιάζουν τυχαίοι, ωστόσο αποτελούν μέρος μιας ακολουθίας[‡]. Έτσι, μέσα από επαναλήψεις, οι κρυπταλγόριθμοι τμήματος προσπάθούν να αποκρύψουν τις συσχετίσεις των χαρακτήρων ώστε το εξαγόμενό τους να φαντάζει τυχαίο. Η διάχυση θα πρέπει να προκαλεί το φαινόμενο της χιονοστοιβάδας (avalanche effect - όρος ο οποίος αποδίδεται στον Horst Feistel [Fe73] αν και χρονολογικά προηγείται ο ορισμός της διάχυσης από τον Claude Shannon), ήτοι η αλλαγή ενός (οποιουδήποτε) χαρακτήρα στο απλό κείμενο να επηρεάζει όσο το δυνατόν περισσότερους χαρακτήρες του κρυπτοκειμένου.

4.4.3 Δίκτυα αντικαταστάσεων-μεταθέσεων

Αμφότεροι οι Rinjdael και Serpent αποτελούν αλγορίθμους αντικατάστασης-μετάθεσης.

- Το μέρος της αντικατάστασης το οποίο συνεισφέρει στην ύπαρξη της ιδιότητας της σύγχησης για το κρυπτοσύστημα και υλοποιείται από τα Sboxes (Substitution boxes).

- Το μέρος των μεταθέσων, το οποίο προσδίδει στα κρυπτοσυστήματα την ιδιότητα της διάχυσης συντίθεται από τους μετασχηματισμούς `ShiftRows` και `MixColumns` για τον Rinjdael και από την απεικόνιση L για τον Serpent.

[‡]Μια προσπάθεια παραγωγής περισσότερο φαινομενικά τυχαίων ακολουθιών επιχειρείται με τη χρήση χαοτικών ακολουθιών στην Κρυπτογραφία.

Συγκεκριμένα για τον Rinjdael, έχει μελετηθει [St17] πως πληροί τις ιδιότητες της σύγχησης και διάχυσης, αφού

- αλλάζοντας ένα δυφίο του απλού κειμένου, τότε μεταβάλλονται κατά μέσο όρο τα μισά δυφία του κρυπτοκειμένου,
- το πλήθος των γύρων αυξάνει την πολυπλοκότητα της μετάθεσης των δυφίων και
- σε κάθε γύρο του το κλειδί αναμειγνύεται (με χρήση της πράξης της αποκλειστικής διάζευξης) με το αποτέλσμα του προηγούμενου γύρου καταστρέφοντας την οποιαδήποτε προφανή συσχέτιση του αποτελέσματος με το κλειδί.

Κεφάλαιο 5

Μια εισαγωγή στην Χαοτική Κρυπτογραφία

Οι αλγόριθμοι του Κεφαλαίου 4 δημιουργήθηκαν έχοντας ως γνώμονα την κρυπτογράφηση κειμένων. Η Χαοτική Κρυπτογραφία αποτελεί μια πτυχή της Κρυπτογραφίας η οποία έχει δώσει αλγορίθμους κρυπτογράφησης εικόνων οι οποίοι επιτυγχάνουν χρόνους γρηγορότερους ή παραπλήσιους σε σχέση με τη χρήση των αλγορίθμων του Κεφαλαίου 4. Οι χαοτικές απεικονίσεις, μπορούν να αποφέρουν (με τις κατάλληλες παραμέτρους) τιμές οι οποίες να είναι φανομενικά τυχαίες (δηλαδή να μην μπορεί να εξαχθεί κάποιο πρότυπο μέσα από την παρατήρησή τους).

5.1 Στοιχεία της Θεωρίας Χάους

Το 1963 ο Edward Lorenz στο [Lo63] μελέτησε τη μεταβολή της ατμόσφαιρας, χρησιμοποιώντας αιτιοκρατικά* μη-γραμμικά συστήματα εξισώσεων. Τα εν λόγω συστήματα σε πολύ μικρές μεταβολές των αρχικών συνθηκών, συν το χρόνω, επέρχονταν τεράστιες αποκλίσεις στα εξαγόμενα†.

Σε ένα αφαιρετικό ορισμό, ένα δυναμικό σύστημα είναι μια συνάρτηση του χρόνου με κάποιο σημείο του χώρου.

*Εάν το σύστημα θεωρηθεί ως μια διαδικασία παραγωγής εξαγόμενων στον χρόνο, τότε αν κατά διαφορετικές υλοποιήσεις της διαδικασίας, δοθεί η ίδια είσοδος (αρχική συνθήκη), τότε το σύστημα πρόκειται να έχει ακριβώς την ίδια συμπεριφορά (δηλαδή να δώσει τα ίδιες τιμές, με την ίδια σειρά).

†Συμπεριφορά στην οποία προσδόθηκε η ονομασία: "το φαινόμενο της πεταλούδας".

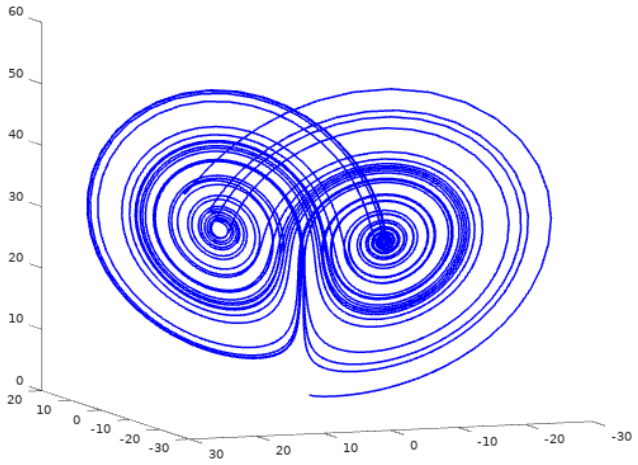
Ορισμός 5.1. Ένα **δυναμικό σύστημα** συνίτθεται από τη δομή $\langle T, M, \Phi \rangle$, όπου το T είναι ένα μονοειδές[‡], το M ένα σύνολο (καλείται **χώρος καταστάσεων**) και η $\Phi : U \subseteq (T \times M) \longrightarrow M$ μία συνάρτηση (καλείται **συνάρτηση εξέλιξης**), όπου

$$\begin{aligned} \pi_2(U) &= M \\ \Phi(0, x) &= x \\ \Phi(t_1,\, \Phi(t_2, x)) &= \Phi(t_1 + t_2, x) \qquad\qquad \forall t_1, t_1 + t_2 \in T,\ t_2 \in I(x) \end{aligned}$$

για $\pi_2(x, y) = y$ και $I(x) = \{t \in T : (t, x) \in U\}$.

Συνεπώς, το μονοειδές T αναπαριστά τον χρόνο (τον i-οστό όρο της χαοτικής ακολουθίας), το σύνολο (σημείων) M είναι ο χώρος και η συνάρτηση Φ υλοποιεί την χαοτική ακολουθία.

Ένας *ελκυστής*[§] (attractor) είναι ένα σημείο στο οποίο το δυναμικό σύστημα τείνει να συγκλίνει. Η χαοτική συμπεριφορά κάποιων δυναμικών συστημάτων με πολλούς ελκυστές έγκειται στο "απρόβλεπτο"[¶] της μεταφοράς του συστήματος από ελκυστή σε ελκυστή.



Σχήμα 5.1: Το σύστημα Lorenz για $\rho = 28$, $\sigma = 10$ και $\beta = 8/3$

Στο [Lo63], ο Edward Lorenz εμπεριέχει το δυναμικό σύστημα:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} -\sigma & \sigma & 0 \\ \rho - z & -1 & -x \\ y & x & -\beta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Στο Σχήμα 5.1 εμφανίζεται η εικόνα του συστήματος για $\rho = 28$, $\sigma = 10$ και $\beta = 8/3$, όπου διακρίνονται δύο ελκυστές. Ο κώδικας για την αναπαραγωγή του Σχήματος εμπεριέχεται στο Παράρτημα Β.1.

[‡]Πρόκειται για ένα σύνολο εφοδιασμένο με μία προσεταιριστική δυαδική πράξη, έστω πως συμβολίζεται $+$ (ήτοι $(a + b) + c = a + (b + c)$, για τα στοιχεία του συνόλου), το οποίο περιέχει ένα ταυτοτικό στοιχείο -έστω 0- όπου $0 + x = x = x + 0$, για κάθε στοιχείο x του συνόλου.

[§]Ενας ορισμός εμπεριέχεται στο [Mi85].

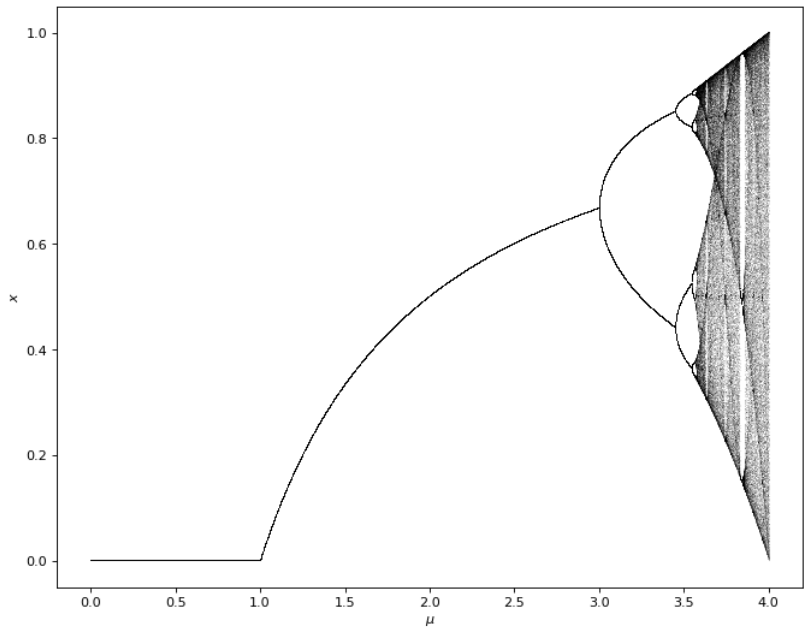
[¶]αδυναμίας εξεύρεσης μιας "προφανούς" συμπεριφοράς

Λογιστική απεικόνιση

Ένα παράδειγμα μιας χαοτικής ακολουθίας συνιστάει η λογιστική απεικόνιση

$$x_{n+1} = \mu \cdot x_n \cdot (1 - x_n) \qquad \text{όπου } (\forall n \in \mathbb{N}_0)\big[x_n \in (0,1)\big] \tag{5.1}$$

για κάποια $\mu \in (0,4)$ και $x_0 \in (0,1)$. Στο Σχήμα 5.1 διακρίνεται το διάγραμμα διακλάσδωσης (bifurcation diagram) της λογιστικής απεικόνισης (5.1) για συντελεστή διακλαδώσεως $\mu \in [0,4]$. Στο διάγραμμα διαφαίνεται ο διαχωρισμός της μίας τροχιάς (ενός ελκυστή) σε δύο (ελκυστές), των δύο (ελκυστών) σε τέσσερις (ελκυστές), και ούτο καθ' εξής.



Τα αραιά σημεία του διαγράμματος αποτελούν ασθενή κλειδιά για τα χαοτικά κρυπτοσυστήματα. Εμφανή αραιά σημεία στο διάγραμμα εντοπίζονται για $\mu < 3.6$ και για $3.825 < \mu < 3.875$. Ωστόσο, αραιά σημεία διακρίνονται και για $\mu \approx 3.65$, $\mu \approx 3.75$ και άλλα.

5.2 Χαοτικά κρυπτοσυστήματα

Η απαρχή της Χαοτικής Κρυπτογραφίας έγινε από τον Robert A. J. Matthews στο [Ma89] το 1989. Ίσως το πρώτο χαοτικό κρυπτοσύστημα σχεδιασμένο για κρυπτογράφηση εικόνων, τοποθετείται στο 1992 και να είναι εκείνο των

Bourbaki και Αλεξόπουλου [BA92]. Η Χαοτική Κρυπτογραφία μπορεί να διακριθεί ως ακολούθως:

- συμμετρική (επί παραδείγματι: [KMA05], [KSFV04]) και
- ασύμμετρη (δημοσίου κλειδιού) (όπως: [ASA11], [BAMA08], [MC05]).

Κατά τα πρότυπα των δικτύων αντικαταστάσεων-μεταθέσεων, αρκετά χαοτικά κρυπτοσυστήματα:

1. εφαρμόζουν μεταθέσεις εικονοστοιχείων (scrambling) και
2. επιφέρουν αλλαγή στη τιμή του κάθε εικονοστοιχείου (diffusion)

σύμφωνα με κάποια(ες) χαοτική(ές) απεικόνιση(εις). Ωστόσο, οι μεταθέσεις αφορούν την εικόνα στην ολότητά της κι όχι μόνον έναν τμήμα της.

Σε μια αντιπαραβολή των κρυπτοσυστημάτων τμήματος και των χαοτικών προκύπτουν οι εξής αντιστοιχίες [AL06]:

Χαοτικοί κρυπταλγόριθμοι	Κρυπταλγόριθμοι τμήματος
Εργοδικότητα	Σύγχηση
Ευαισθησία στις αρχικές συνθήκες	Επιρροή στο κρυπτοκείμενο μεταβάλλοντας το απλό κείμενο ή το κλειδί (σύγχηση και διάχυση)
Ιδιότητα μίξης	Επιρροή στο κρυπτοκείμενο μεταβάλλοντας ένα τμήμα του απλού κειμένου
Αιτιοκρατική δυναμική	Παραγωγή ψευδο-τυχαίων αριθμών
Πολυπλοκότητα των πράξεων μίξης του κλειδιού με το κείμενο καθώς και οι πράξεις παραλλαγής του	Πολυπλοκότητα της δομής της ακολουθίας

Η εργοδικότητα είναι ο όρος ο οποίος χρησιμοποιείται για να περιγράψει τη συμπεριφορά ενός δυναμικού συστήματος, το οποίο παρουσιάζει εν γένει την ίδια συμπεριφορά ως προς τον χρόνο και τον χώρο. Οι αρχικές συνθήκες του χαοτικού συστήματος αποτελούν το κλειδί του κρυπτοσυστήματος. Με τη γνώση των αρχικών συνθηκών, οι επικοινωνούσες οντότητες μπορούν να συγχρονίσουν τα χαοτικά τους συστήματα και να επιτύχουν την (απο)κρυπτογράφηση. Η αιτιοκρατική δυναμική είναι η δυνατότητα παραγωγής τιμών μέσα από τα

χαοτικά συστήματα. Η απλή παρατήρηση των τιμών (χάριν των ιδιοτήτων των χαοτικών συστημάτων) είναι πολύ δύσκολο στο να οδηγήσει στην εξαγωγή ενός μοτίβου παραγωγής αυτών (το οποίο μοτίβο είναι πιο περίπλοκο από τα αντίστοιχα που παράγονται με απλούστερες αριθμητικές ακολουθίες). Τέλος, η ιδιότητα της μίξης στα χαοτικά κρυπτοσυστήματα μπορεί να μεταθέσει ένα στοιχείο του μηνύματος (ένα δυφίο ενός εικονοστοιχείου) σε οποιοδήποτε μέρος ολόκληρου του κρυπτοκειμένου, εν αντιθέσει με τους αλγορίθμους τμήματος οι οποίοι μεταθέτουν τα στοιχεία του τμήματος, μόνον εντός του τμήματος. Σε μια πρώτη ανάγνωση, η εν λόγω ιδιότητα αποκλείει την παράλληλη υλοποίηση των χαοτικών κρυπτοσυστημάτων (αφού δεν υφίσταται η έννοια της μόνον εντός τμήματος μετάθεσης), Ωστόσο, έχουν σχεδιασθεί χαοτικά κρυπτοσυστήματα τα οποία υποστηρίζουν παράλληλους υπολογισμούς όπως το [MYI12] (για το οποίο υπάρχει επιτυχημένη επίθεση από στο [WZL16]), το [LZLCD18], το [RSSN17] στα οποία χρησιμοποιείται -μεταξύ άλλων και- η λογιστική απεικόνιση (5.1). Τέλος, το [MMB10] αποτελεί ένα κρυποσύστημα βασισμένο στο σύστημα ελκυστών του Lorenz (βλ. Σχήμα 5.1) και μπορεί επίσης να υλοποιηθεί μέσω παράλληλων διεργασιών.

5.3 AES με στοιχεία Χάους

Δύο από τις σημαντικότερες παραμέτρους του κρυπτοσυστήματος AES είναι το κλειδί και το κουτί αντικαταστάσεων (Sbox). Σε αμφότερες τις περιπτώσεις οι οποίες επακολουθούν, οι αρχικές συνθήκες για τις χαοτικές ακολουθίες θα πρέπει να θεωρηθούν ως μέρος του συμμετρικού κλειδιού.

Παραγωγή κλειδιού μέσω χαοτικών ακολουθιών

Σε μια *προσπάθεια* παραγωγής πιο δυνατών κλειδιών το [PB13] οι C. Pradhan και A. K. Bisoi προτείνουν αντί το κλειδί να διαλέγεται απευθείας από τις οντότητες οι οποίες επικοινωνούν, να προκύπτει από μία διαδικασία παραγωγής μέσω μιας χαοτικής ακολουθίας. Προς τούτο, ορίζεται η συνάρτηση

$$f : [0,1] \longrightarrow \{0,1\} \qquad\qquad f(z) := \begin{cases} 1, & \text{εάν } z \geq 0.5 \\ 0, & \text{αλλιώς} \end{cases}$$

Θεωρείται η μονοδιάστατη λογιστική απεικόνιση (5.1). Από την ακολουθία $\big(f(x_n)\big)_{n\in\mathbb{N}_0}$, μπορούν να εξαχθούν τα δυφία του αρχικού κλειδιού K_0 του AES.

Θεωρούνται $k, \mu \in \mathbb{N}$. Ορίζεται η πεπλεγεμένη χαοτική ακολουθία

$$\begin{cases} x_{n+1} = 1 - \mu \cdot y_i^2 \\ y_{n+1} = \cos\left(k \cdot \cos^{-1}(x_i)\right) \end{cases} \qquad \text{όπου } (\forall n \in \mathbb{N}_0)\big[x_n, y_n \in [-1, 1]\big]$$

Στο [PB13] προτείνεται $\mu = 2$ και $k = 6$. Η παραπάνω ακολουθία δίδει τις εξής δύο υπακολουθίες:

$$X = x_0\, x_1\, x_2\, \ldots \qquad\qquad Y = y_0\, y_1\, y_2 \ldots$$

Από τις παραπάνω ακολουθίες προκύπτει η ακολουθία

$$Z = (z_{ij})_{\substack{i=1 \\ j=1}}^{+\infty}, \qquad \text{όπου } z_{ij} = x_i \cdot y_j$$

Από την παραπάνω ακολουθία μπορούν να εξαχθούν τα δυφία τα οποία θα συντελέσουν το αρχικό κλειδί K_0 του AES.

Στο [PB13] σημειώνεται πως η χρήση χαοτικών ακολουθιών αυξάνει τον χρόνο υλοποίησης του αλγορίθμου AES. Ωστόσο, ο συνδυασμός αμφότερων των ανωτέρω θεωρήσεων επιτυγχάνει παραπλήσιο χρόνο με την κλασσική υλοποίηση του AES.

Κουτιά αντικαταστάσεων μέσω χαοτικών ακολουθιών

Στο [BMMH10] *προτείνεται* η παραγωγή του κουτιού αντικαταταστάσεων του AES μέσω της χρήσης της μονοδιάστατης λογιστικής απεικόνισης (5.1). Ένα κουτί αντικατάστασης, θα πρέπει

1. να εισάγει μη-γραμμικότητα στα δεδομένα και

   [Να μην προκύπτει κάποιο πρότυπο σύμφωνα με το οποίο μετατίθενται τα δυφία του κειμένου.]

2. να μην περιέχει σταθερά σημεία.

   [Ένα σταθερό σημείο αφήνει κάποιο(α) δυφίο(α) στην ίδια θέση.]

Στο [BMMH10] προτείνεται $\mu = 4 - \varepsilon$, για κάποιο $\varepsilon > 0$ οσοδήποτε μικρό καθώς. Επιπλέον, παρατίθενται τα αποτελέσματα των δοκιμών των για $x_0 \in (0, 1)$ και "απλό κείμενο" παρηγμένο από επιλογή χαρακτήρων σύμφωνα με την ομοιόμορφη και την κανονική κατανομή.

Μέρος ΙΙ

Υλοποίηση

Κεφάλαιο 6

Παράλληλος προγραμματισμός

Το παρόν Κεφάλαιο παρουσιάζει τα εργαλεία που χρησιμοποιήθηκαν για την επίτευξη του σκοπού της Εργασίας. Όπως αναφέρθηκε και στο Κεφάλαιο 1, οι εικόνες (για την ακρίβεια, η αναπαράστασή τους στον υπολογιστή) αποτελούν ένα αντικείμενο το οποίο απαιτεί περισσότερο χρόνο επεξεργασίας, απ' όσο ένα αριθμητικό δεδομένο ή ένα κομμάτι κειμένου. Για το λόγο αυτό, επιχειρείται η χρήση προγραμμάτων διαχείρησης μεγάλων δεδομένων με σκοπό την κρυπτογράφηση και αποκρυπτογράφηση εν εξελίξει ροών εικόνων για την απευθείας μετάδοσή τους μέσω ενός ανασφαλούς διαύλου επικοινωνίας.

6.1 Ροές (Streams)

Οι ηλεκτρονικοί υπολογιστές καλούνται να υλοποιήσουν κάθε στιγμή πολλές διεργασίες ταυτόχρονα. Έστω για παράδειγμα πως ένας χρήστης στον ηλεκτρονικό του υπολογιστή έχει ενεργά:

1. έναν κειμενογράφο ο οποίος αποτελείται:
    - από τη διεπαφή με το χρήστη (το UI) η οποία ανανεώνεται δείχνοντας στον χρήστη τι έχει πληκτρολογήσει και
    - μία διεργασία η οποία σώζει την πρόοδο του χρήστη στον κειμενογράφο κάθε 2 λεπτά.
2. έναν περιηγητή (browser) στο υπόβαθρο το οποίο αναπαράγει μουσική όσο ο χρήστης συγγράφει στον κειμενογράφο και αποτελείται:
    - από τη διεπαφή με τον χρήστη (το UI) και
    - μία διεργασία η οποία αναπαράγει τη μουσική.

Γίνεται αντιληπτό πως διαφορετικά προγράμματα (και κατ' επέκτασην οι διεργασίες οι οποίες τα συνοδεύουν) ζητούν πόρους από το σύστημα την ίδια στιγμή. Οι ηλεκτρονικοί υπολογιστές δημιουργούν ροές (streams) οι οποίες εξυπηρετούνται από τον υπολογιστή παράλληλα, αξιοποιόντας διαφορετικούς επεξεργαστές του. Έτσι, η μουσική δεν διακόπτεται κάθε φορά που ο χρήστης πατάει ένα πλήκτρο για να συγγράψει στον κειμενογράφο, ούτε ο κειμενογράφος μένει αδρανής έως ότου ολοκληρωθεί η αναπαραγωγή της μουσικής στο υπόβαθρο. Επίσης, τα προγράμματα (και οι διεργασίες αυτών) τα οποία εκτυλίσσονται στο υπόβαθρο απασχολούν πόρους του ηλεκτρονικού υπολογιστή, παρ' όλο που ο χρήστης χρησιμοποιεί μια συγκεκριμένη εφαρμογή.

***Σύμβαση**. Έστω, για λόγους απλότητας, πως κάθε μία από τις παραπάνω διεργασίες αποτελεί μία ροή. Έτσι, γίνεται εναλλάξ χρήση των όρων "διεργασία" και "ροή" για το υπόλοιπο της παρούσης Ενότητος.*

Υπάρχουν διαφορετικές αντιμετωπίσεις για το πως εξυπηρετούνται οι ροές από τον υπολογιστή. Επί παραδείγματι,

Εκ περιτροπής: κάθε μία διεργασία που έρχεται προς εκτέλεση διαβιβάζεται στον πρώτο διαθέσιμο επεξεργαστή.

Η εν λόγω -φαινομενικά βέλτιστη- λύση, παρόλο που δεν μεροληπτεί, δυνητικά θα μπορούσε να οδηγήσει σε κωλυσιεργία των υπολοίπων διεργασιών. Επί παραδείγματι, εάν μόνον ένας επεξεργαστής ήταν διαθέσιμος και ανέκυπτε η διεργασία της αποθήκευσης αλλαγών σε ένα μεγάλο κείμενο, τότε μόλις έφθανε η σειρά της να εξυπηρετηθεί, μέχρι την περάτωσή της, δεν θα άφηνε χρόνο για κάποια άλλη διεργασία, άρα η μουσική δεν θα μπορούσε να ξεκινήσει να αναπαράγεται όσο η διαδικασία της αποθήκευσης ήταν σε εξέλιξη ή μεχρις ούτε βρεθούν διαθέσιμοι πόροι.

Οι λιγότερο χρονοβόρες πρώτα: Οι εργασίες οι οποίες κοστίζουν λιγότερο σε χρόνο, εξυπηρετούνται έναντι των πιο κοστοβόρων διεργασιών.

Ωστόσο, μια τέτοια αντιμετώπιση θα δημιουργούσε "πείνα" (starvation) υπό την έννοια πως η εν λόγω μεροληψία θα ανέβαλε την εκτέλεση πιο χρονοβόρων διεργασιών όσο θα ανέκυπταν καινούργιες, λιγότερο χρονοβόρες, διεργασίες.

Για να αντιμετωπίσουν φαινόμενα, όπως τα παραπάνω, κάθε επεξεργαστής σημειώνει μια πρόοδο σε μια διεργασία, περατώνοντάς την μερικώς και κατόπιν στρέφει την προσοχή τους σε μια άλλη διεργασία. Επαναλαμβάνεται η ίδια τακτική. Έτσι, αποφεύγεται η μεροληψία και εξυπρετούνται όλες οι

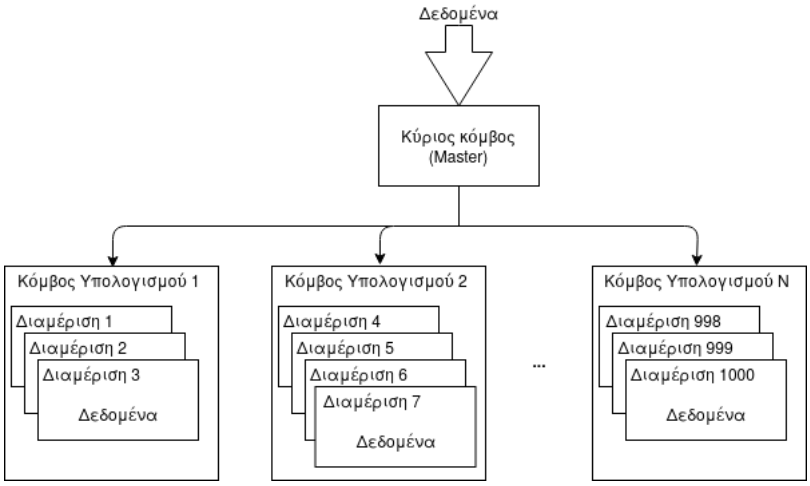
διεργασίες ανεξαρτήτως του χρονικού τους κόστους. Ωστόσο, ακόμη και η εν λόγω θεώρηση ενέχει επίσης τον κίνδυνο ανάκαμψης κωλυσιεργίας ανάμεσα στις διεργασίες. Πρωτίστως, απαιτεί από τον ηλεκτρονικό υπολογιστή ανά τακτά χρονικά διαστήματα να στρέφει την προσοχή του σε άλλες διεργασίες. Η εν λόγω "στρέψη της προσοχής" απαιτεί μία οργάνωση η οποία κοστίζει σε πόρους και σε χρόνο. Επίσης, έστω πως μια διεργασία απαιτεί την περάτωση μιας άλλης διεργασίας έτσι ώστε να εξυπηρετήσει τον σκοπό της. Ο ηλεκτρονικός υπολογιστής δεν δύναται να γνωρίζει ένα τέτοιο γεγονός, άρα, θα ενεργοποιεί την εν λόγω διεργασία η οποία θα δηλώνει πως δεν μπορεί να προχωρήσει και κατόπιν θα στρέφει την προσοχή του σε άλλη διεργασία. Η στρέψη της προσοχής όμως σε μια διεργασία, όπως αναφέρθηκε αποτελεί μία κοστοβόρα διαδικασία. Προς τούτο, υπάρχουν προγραμματιστικές τεχνικές οι οποίες "κοιμίζουν" μια ροή (κι άρα δεν απασχολεί το σύστημα) έως ότου είναι διαθέσιμα όλα τα προαπαιτούμενά της. Κατόπιν, η ροή ενεργοποιείται. Συνεπώς, θα πρέπει να γίνεται μία προσεκτική οργάνωση των διεργασιών που υλοποιούνται ταυτόχρονα, με σκοπό τη βέλτιστη αξιοποίηση των παρεχόμενων πόρων του συστήματος. Τέλος, φαντάζει πως μπορούν να υλοποιηθούν όλες οι διεργασίες παράλληλα, αναθέτοντας την κάθε μία σε μια ροή. Αυτό επίσης ενέχει την δυσλειτουργική περίπτωση όπου ο ηλεκτρονικός υπολογιστής θα αργήσει παρά πολύ να υλοποιήσει ακόμη και την πιο σύντομη διεργασία καθότι θα υπάρχει πληθώρα διεργασιών στις οποίες θα στρέφει την προσοχή του. Ένας εμπειρικός κανόνας υποδεικνύει πως ο αριθμός ροών που χρησιμοποιούνται πρέπει να εμπίπτει ανάμεσα στο πλήθος των φυσικών επεξεργαστών του ηλεκτρονικού υπολογιστή και στο πλήθος των εικονικών επεξεργαστών που διαθέτει. Την ευθύνη για την οργάνωση των υπολογισμών (διεργασιών) σε διαφορετικές ροές για την υλοποίηση της εφαρμογής που περιγράφεται στις παρακάτω Ενότητες, την έχει αναλάβει (ο εσωτερικός μηχανισμός) του Spark.

6.2 Παράλληλοι υπολογισμοί στο Spark

Το πρόγραμμα διαχείρησης μεγάλων δεδομένων το οποίο πρόκειται να χρησιμοποιηθεί για την παράλληλη διενέργεια των (χρονοβόρων) υπολογισμών της (απο)κρυπτογράφησης εικόνων είναι το Spark [Sp], της εταιρίας Apache [Ap]. Το Spark δημιουργεί ένα πλάνο προς εκτέλεση (execution plan) το οποίο, όταν μόνον του ζητηθεί, πρόκειται να υλοποιηθεί. Όντας πρόγραμμα επεξεργασίας μεγάλων δεδομένων, το Spark δεν πρόκειται να δράσει άπαξ στην ολότητα των δεδομένων. Πρωτίστως πρόκειται να δημιουργήσει ροές (streams) για την ανάγνωση των δεδομένων, άλλως θα ήταν πιθανό να ανακύψει υπερχείληση μνήμης ενός υπολογιστή κατά τη προσπάθεια να ανακτήσει εξ ολοκλήρου ένα

αρκετά μεγάλο αρχείο. Το Spark χρησιμοποιεί τους διαθέσιμους υπολογιστικούς πόρους που του παράσχονται. Μπορεί να είναι οι επεξεργαστές ενός φυσικού μηχανήματος (ηλεκτρονικού υπολογιστή) ή οι πόροι πολλών υπολογιστών που επικοινωνούν αναμεταξύ τους (εν τοιαύτη περιπτώσει λέγεται πως σχηματίζεται μία *συστάδα* (cluster)).

Τα δεδομένα διαμοιράζονται από έναν ειδικό κόμβο -καλούμενο κύριο κόμβο- προς τους κόμβους υπολογισμών. Παρακάτω ακολουθεί μία επίπλαστη σχηματοποίηση της περιγραφήσας δομής.



Κάθε υπολογιστικός κόμβος έχει τα δεδομένα ταξινομημένα (σύμφωνα με τους εσωτερικούς μηχανισμούς του Spark) σε διαμερίσεις καθώς και κάποιον αριθμό ροών που έχει διαθέσιμο για την (εν παραλλήλω) επεξεργασία των δεδομένων που εμπεριέχει. Κάθε φορά όπου ανακύπτει μία εντολή η οποία αφορά τα δεδομένα, εκείνη διαβιβάζεται στους υπολογιστικούς κόμβους κι εκείνοι την εκτελούν σε κάθε διαμέριση που εμπεριέχουν με όσο το δυνατόν παράλληλο τρόπο.

Εκ κατασκευής, το Spark επιλέγει να δημιουργήσει 3-4 διαμερίσεις ανά φυσικό επεξεργαστεί που διαθέτει το μηχάνημα (ή η συστάδα μηχανημάτων) στο οποίο εκτελείται.

Πλάνο εκτέλεσης

Το πλάνο εκτέλεσης αποτελεί ένα ακυκλικό κατευθυνόμενο γράφη (Directed Acyclic Graph - DAG).

Ακυκλικό: καθόσον το Spark δεν υποστηρίζει βρόγχους, διότι κάθε μετασχηματισμός λαμβάνει χώρα άπαξ στα δεδομένα (μετά τα παραλλάσσει),

Κατευθυνόμενο: καθόσον απεικονίζει *ροή* δεδομένων.

Το γράφημα υποδεικνύει τη ροή των δεδομένων καθώς εκείνη επεξεργάζονται από το Spark. Διακρίνεται σε στάδια (stages). Κάθε στάδιο απαιτεί τη συγκέντρωση των δεδομένων από τους κόμβους. Κάθε στάδιο δύναται όπως εμπεριέχει πολλούς μετασχηματισμούς οι οποίοι εκτελούνται από τον κάθε κόμβο ξεχωριστά.

6.3 Ροές Δεδομένων στο Spark

Η εφαρμογή του Κεφαλαίου 7 χρησιμοποιεί δεδομένα πραγματικού χρόνου (καταγραφή εικόνας μέσω κάμερας). Το Spark παρέχει τη δομή των DStreams για τη διαχείρησή τους.

6.3.1 Ελαστικά σε Σφάλματα Κατανεμημένα Σύνολα Δεδομένων (RDDs)

Τα Ελαστικά σε Σφάλματα Κατανεμημένα Σύνολα Δεδομένων (Resilient Distributed Datasets - RDDs [RDD]) αποτελούν συλλογές δεδομένων που παρέχονται Spark. Η ονομασία τους περιέχει όλες τους τις ιδιότητες:

Dataset (σύνολο δεδομένων): Η εν λόγω δομή του Spark περιέχει δεδομένα τα οποία έχουν παρασχεθεί στο Spark για επεξεργασία. Μπορεί να είναι μεγάλα δεδομένα, μπορεί όμως και να είναι ένα πιο εύκολα διαχειρίσιμο σύνολο δεδομένων και απλά να γίνεται χρήση της δυνατότητας της εν παραλλήλω επεξεργασίας τους από το Spark.

Distributed (κατανεμημένα): Τα δεδομένα κατανέμονται στις διαφορετικές διαμερίσεις τις οποίες έχει ορίσει το Spark.

Resilient (fault resistant - ελαστικά σε σφάλματα): Εάν για οποιονδήποτε λόγο τα δεδομένα μιας διαμέρισης καταστραφούν, τότε μπορούν να ανακτηθούν ή να επαναϋπολογιστούν.

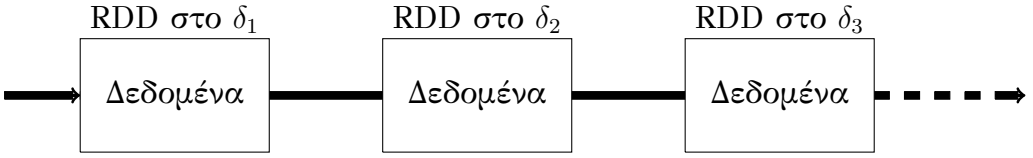
Η εν λόγω δομή παράσχει τη δυνατότητα παράλληλης διενέργειας υπολογισμών στα δεδομένα που περιέχει. Στην εφαρμογή που παρουσιάζεται στην §7.2, θα διενεργηθεί (απο)κρυπτογράφηση διαφορετικών τμημάτων της εικόνας σε παράλληλο χρόνο. Προς τούτο, έχουν επιλεγεί οι κατωτέρω κατηγορίες κρυπταλγορίθμων.

- Αλγόριθμοι τμήματος (οι οποίοι διαμερίζουν το απλό κείμενο σε τμήματα και το επεξεργάζονται ανεξάρτητα το ένα από το άλλο) και

- Παράλληλες υλοποιήσεις αλγορίθμων της Χαοτικής Κρυπτογραφίας, καθόσον οι εν λόγω αλγόριθμοι χρειάζονται λιγότερο χρόνο για να κρυπτογραφήσουν/αποκρυπτογραφήσουν εικόνες.

6.3.2 Διακριτοποιημένες Ροές (DStreams)

Μία διακριτοποιημένη ροή (Discretized Stream - DStream [DStr]), είναι μία συλλογή από RDDs η οποία είναι διακριτοποιημένη σε (μη αλληλεπικαλυπτόμενα) χρονικά διαστήματα. Θεωρώντας τα χρονικά διαστήματα $\delta_{i+1} := [t_i, t_{i+1})$, για $i \in \mathbb{N}_0$, όπου $t_0 = 0$ (η απαρχή της καταγραφής των δεδομένων) και $\Delta t = t_{i+1} - t_i > 0$ για $i \in \mathbb{N}_0$, ένα DStream μπορεί να αναπαρασταθεί ως ακολούθως.



Το `Spark` (έκδοση 3.0) υποστηρίζει την ανάγνωση και την αποστολή DStreams μέσα από sockets. Άρα, παρέχεται η δυνατότητα τροφοδότησης του `Spark` με τη ροή των εικόνων από έναν εξυπηρετητή (server) ο οποίος καταγράφει τις εικόνες και κατ᾽ όπιν της τροποποιήσεως των εικόνων, αυτές δύναται να αποσταλλούν σε έναν εξυπηρετούμενο (client) ο οποίος μπορεί να τις προβάλει (ή και να τις αποκρυπτογραφήσει με τη γνώση του συμμετρικού κλειδιού).

Στην εφαρμογή που παρουσιάζεται στην §7.2, τα δεδομένα πρόκειται να είναι εικόνες που έχουν ληφθεί από το `Spark` σε μια σειρά συνεχόμενων χρονικών διαστημάτων. Χρησιμοποιώντας τη μεθοδολογία του Map-Reduce:

Map: Οι εικόνες διαμερίζονται σε μέρη τα οποία θα (απο)κρυπτογραφούνται ανεξάρτητα (σε παράλληλες διεργασίες).

Reduce: Τα (απο)κρυπτογραφημένα μέρη θα συντίθενται παράσχοντας την κρυπτογραφημένη εικόνα.

6.3.3 Μετασχηματισμοί στο Spark

Οι παρεχόμενες από το Spark συναρτήσεις μετατρέπουν τα δεδομένα σύμφωνα με τις υποδείξεις του πλάνου εκτέλεσης. Διακρίνονται δύο τύποι μετασχηματισμών στις δομές του Spark:

Περιορισμένοι Μετασχηματισμοί ((Narrow) Transformations): Πρόκειται για μετασχηματισμούς στα δεδομένα οι οποίοι μπορούν να εφαρμοστούν ξεχωριστά στον κάθε κόμβο υπολογισμού. Το Spark επιμερίζει τους μτασχηματισμούς στον κάθε κόμβο και εκείνοι με τη σειρά τους, τους εφαρμόζουν σε κάθε διαμέριση που διαθέτουν.

Ευρείς Μετασχηματισμοί (Wide Transformations / Actions): Πρόκειται για μετασχηματισμούς συσσωμάτωσης (grouping), οι οποίοι χρειάζονται την ολότητα των δεδομένων ώστε να εφαρμοσθούν. Το Spark ανταλλάζει δεδομένα αναμεταξύ των κόμβων (άρα δημιουργείται κίνηση ανάμεσα στους επεξεργαστές του υπολογιστή (network traffic) οι οποίοι ανταλλάζουν δεδομένα) και κατόπιν εφαρμόζει τους μετασχηματισμούς. Η εν λόγω ανταλλαγή δεδομένων ανάμεσα στους υπολογιστικούς κόμβους του Spark καλείται *ανακάτεμα* (**shuffling**).

Από τα ανωτέρω, καθίσταται κατανοητό πως οι ευρείς μετασχηματισμοί κοστίζουν σε πόρους και σε χρόνο, καθώς και μπορούν να προξενήσουν υπερχείληση μνήμης (αφού το Spark στην κυριότητά του χρησιμοποιείται για τη διαχείρηση μεγάλων δεδομένων). Κατ' επέκτασην, είναι επιθυμητό -όποτε αυτό καθίσταται εφικτό- να αντικαθιστούνται από μία αλληλουχία περιορισμένων μετασχηματισμών, οι οποίοι εφαρμόζονται στο κάθε κόμβο υπολογισμού ξεχωριστά και κατόπιν τα -μετασχηματισμένα πλέον- δεδομένα να ανταλλάζονται αναμεταξύ των κόμβων. Επί παραδείγματι, αντί

(α΄) της εφαρμογής του ευρέος μετασχηματισμού `groupByKey` (συσσωμάτωση των δεδομένων, από όλους τους υπολογιστικούς κόμβους, σύμφωνα με το χαρακτηριστικό τους γνώρισμα) και κατόπιν

(β΄) εφαρμογή ενός περιορισμένου μετασχηματισμού

μια αποτελεσματικότερη υλοποίηση προκύπτει μέσω μιας παράκαμψης:

(i) εφαρμόζεται ο περιορισμένος μετασχηματισμός `reduceByKey` (συσσωμάτωση των δεδομένων, σε κάθε υπολογιστικό κόμβο ξεχωριστά),

(ii) εφαρμόζεται ένας περιορισμένος μετασχηματισμός και τελικά

(iii) shuffling (ανταλλαγή των ήδη ομαδοποιημένων/μετασχηματισμένων δεδομένων).

Στην Εφαρμογή (βλ. Ενότητα §7.2) χρησιμοποιήθηκαν αμφότερα τα είδη των μετασχηματισμών όπως:

Περιορισμένοι Μετασχηματισμοί: απεικόνιση σε πλειάδες, εφαρμογή συναρτήσεων σε πλειάδες, αναγωγή κλειδιού, επιλογή εικόνων οι οποίες έχουν πλήρως κρυπτογραφηθεί.

Ευρείς Μετασχηματισμοί: Συγκέντρωση και ταξινόμηση των εικόνων σε αύξουσα σειρά σύφωνα με την χρονική στιγμή που απεικονίζουν, μετάδοση εικόνας μέσω ενός socket.

Κεφάλαιο 7

Εφαρμογή

Κατά το παρόν Κεφάλαιο παρουσιάζεται και αναλύεται ολόκληρο το οικοσύστημα το οποίο χρησιμοποιήθηκε για να προσομοιωθεί η εν παραλλήλω κρυπτογράφηση και μετάδοση μιας εν εξελίξει ροής εικόνων.

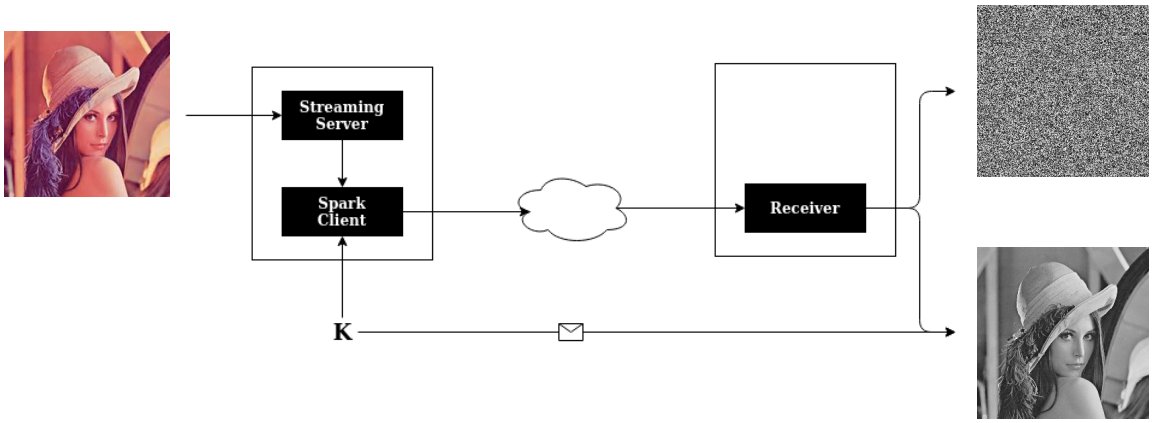
Η εφαρμογή διακρίνεται στα 3 ακόλουθα μέρη:

Streaming Server: Πρόκειται για το μέρος το οποίο καταγράφει από την κάμερα του ηλεκτρονικού υπολογιστή και αποστέλει τη ροή στο Spark.

Spark Client: Εμπεριέχει τη ροή των δεδομένων και την επεξεργασία τους από το Spark. Επίσης είναι το σημείο όπου τα ((απο)κρυπτογραφημένα) δεδομένα μεταδίδονται.

Receiver: Αποτελεί τον παραλήπτη της ροής, δηλαδή το κομμάτι το οποίο δέχεται μια ροή εικόνων και την προβάλει.

Το πανόραμα της εφαρμογής (για το συμμετρικό κλειδί **K**) έχει ως εξής:



Τεχνικές λεπτομέρειες

Η γλώσσα προγραμματισμού όπου έγινε η υλοποίηση είναι η `Python` (έκδοση 3.6). Ο λόγος της επιλογής είναι πως παρ' όλο που:

- το Spark (έκδοση 3.0) υποστηρίζεται στις γλώσσες `Scala`, `Java`, `Python`, `R` και
- οι γλώσσες `Java`, `Python`, `C#`, ...υποστηρίζουν προγραμματισμό με χρήση νημάτων

η `Python`, πλέον του γεγονότος ότι συγκεράζει τα παραπάνω, υποστηρίζει τη άμεση πρόσβαση καταγραφής εικόνας μέσω κάμερας ηλεκτρονικού υπολογιστή, καθώς και τη μετατροπή της σε μορφή εύκολα διαχειρίσιμη από την ίδια.

Για πιο σύντομη επεξεργασία εικόνων, θεωρούνται ασπρόμαυρες εικόνες, οι οποίες αποτελούν δισδιάστατα αντικείμενα (εν αντιθέσει με τις έγχρωμες οι οποίες αναπαριστώνται ως τρισδιάστατα αντικείμενα). Το μέγεθος των εικόνων είναι 256×256.

Όλες οι επικοινωνίες ανάμεσα στις συνιστώσες της Εφαρμογής γίνονται μέσα από sockets, δηλαδή διευθύνσεις (αποτελούμενες από IP (Interenet Protocol) και μία θύρα (port) στην IP) στις οποίες αποστέλονται δεδομένα. Ειδικότερα, οι StreamingServer και Receiver αποτελούν έκαστως έναν TCP Server (Transmission Control Protocol) όπου στη πρώτερη περίπτωση μεταδίδει μια ροή δεδομένων στο `Spark`, ενώ στη δεύτερη λαμβάνει τη παραγόμενη από το `Spark` ροή δεδομένων.

Σύμβαση. *Στις διαστατικές αναλύσεις οι εικόνες πρόκειται να πραγματεύονται έχοντας (αν και δεν αποτελεί δέσμευση) αρχική διάσταση* 256×256.

7.1 Δημιουργία ροής εικόνων (Streaming Server)

Παρακάτω περιγράφεται ο μηχανισμός ο οποίος είναι υπεύθυνος για τη δημιουργία της ροής εικόνων. Ως μετασχηματισμός ορίζεται:

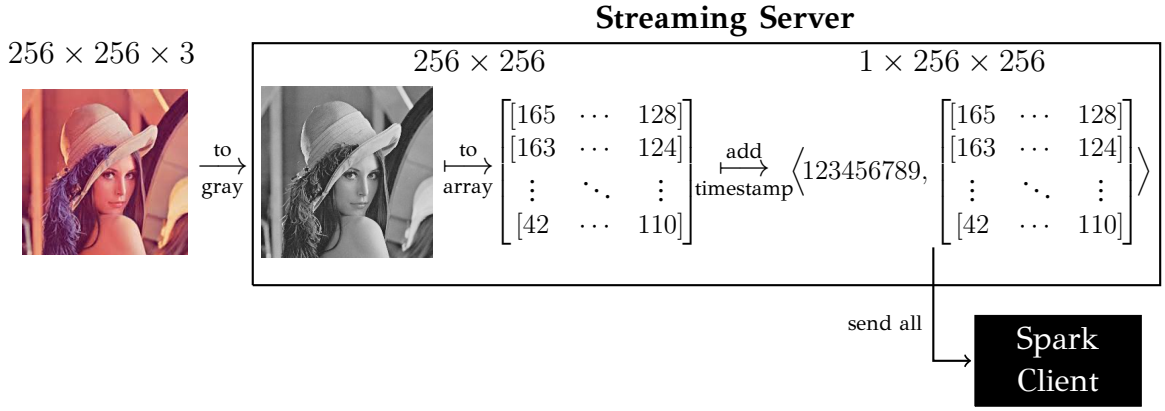
$$\text{StreamingServer} : \Phi\text{υσικό Αντικείμενο} \longrightarrow \Big\langle \mathbb{Z},\, \mathbb{M}_{256\times 256}\big(\{0,1,\ldots,255\}\big) \Big\rangle$$

όπου ως $\mathbb{M}_{256\times 256}\big(\{0,1,\ldots,255\}\big)$ συμβολίζεται ο χώρος των 256×256 πινάκων με στοιχεία στο σύνολο $\{0,1,\ldots,255\}$ (τονικότητα του γκρίζου). Ο συμβολισμός αναπαριστάει ένα ζεύγος. Στην παρούσα περίπτωση, η δεύτερη συντεταγμένη

του ζεύγους εμπεριέχει την εικόνα ως πίνακα και η πρώτη συντεταγμένη τη χρονική στιγμή η οποία απεικονίζεται.

Η υλοποίηση αποτελείται από έναν TCP Server ο οποίος ενεργοποιείται όταν υπάρχει κάποιο πρόγραμμα διαθέσιμο να λάβει τα δεδομένα του στο socket που εκμπέμπει. Στην προκειμένη περίπτωση, στο socket θα βρίκεται το Spark όπου ο Streaming Server θα αποστέλλει οτιδήποτε καταγράφει μέσω της κάμερας του ηλεκτρονικού υπολογιστή. Ως ροή εννοείται μια αλληλουχία εικόνων οι οποίες στο παρόν στάδιο επεξεργάζονται ως εξής:

1. Μετατροπή της εικόνας σε διασδιάστατο πίνακα.
2. Επισύναψη της χρονικής στιγμής η οποία απεικονίζεται.
3. Αποστολή στο socket επικοινωνίας με το Spark του ζεύγους της χρονικής στιγμής με την εικόνα.



Η υλοποίηση του Streaming Server εμπεριέχεται στο Παράρτημα Α.2.

7.2 Ο αλγόριθμος (απο)κρυπτογράφησης (Spark Client)

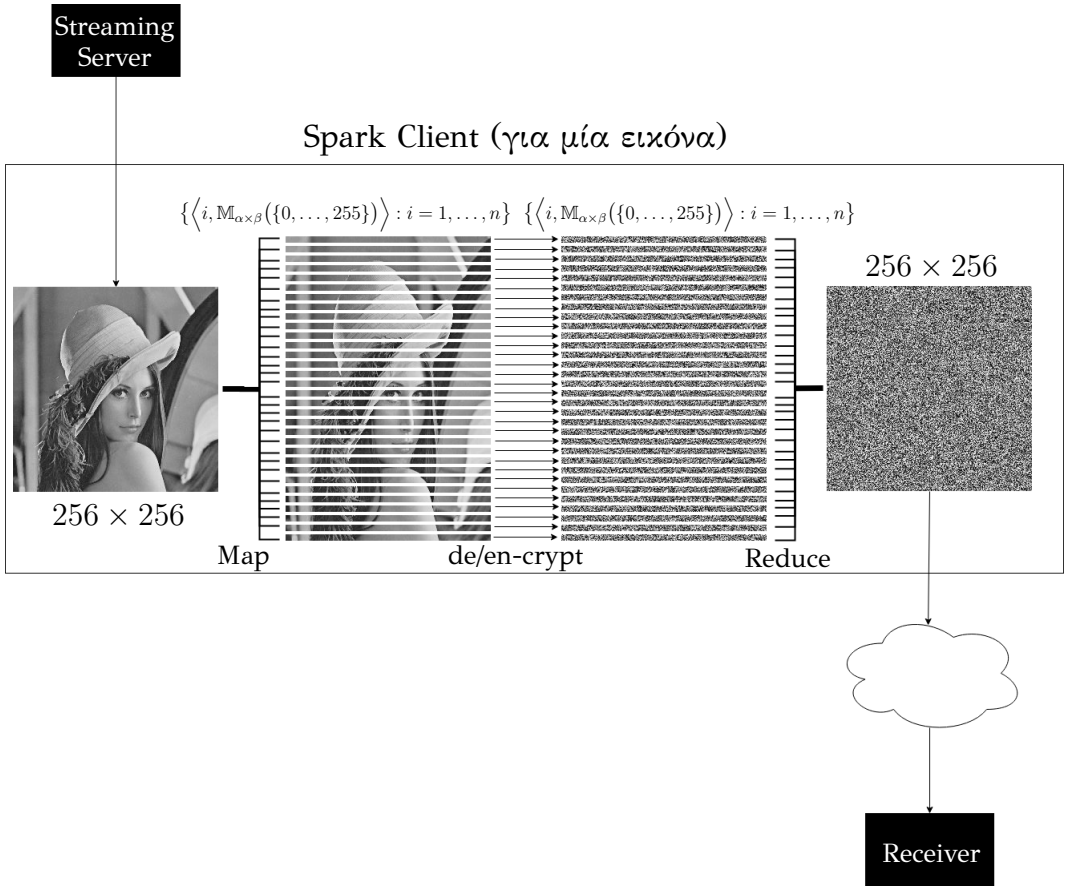
Σύμβαση. *Παρά του γεγονότος πως, η Εφαρμογή αφορά ροή δεδομένων, στην ενθάδε Ενότητα πρόκειται να παρουσιασθεί η πορεία μιας εικόνος.*

Το μέρος του Spark Client περιέχει την αλγοριθμική λογική. Είναι επιφορτισμένο με όλες τις πράξεις και τους μετασχηματισμούς που πρέπει να διενεργηθούν ώστε να (απο)κρυπτογραφηθεί η ροή των εικόνων. Ως μετασχηματισμός

περιγράφεται ως εξής:

$$\text{SparkClient} : \Big\langle \mathbb{N},\, \mathbb{M}_{256\times256}\big(\{0,1,\ldots,255\}\big)\Big\rangle \longrightarrow \Big\langle \mathbb{N},\, \mathbb{M}_{256\times256}\big(\{0,1,\ldots,255\}\big)\Big\rangle$$

Παρακάτω αναλύονται σε βήματα οι μετασχηματισμοί που χρησιμοποιούνται.



Διαμέριση της εικόνας (Map)

Μετά τη λήψη της εικόνας εκείνη διαμερίζεται σε τμήματα τα οποία πρόκειται στο επόμενο στάδιο να προσπελαστούν σε παράλληλο χρόνο. Υπο μορφή απεικόνισης, το παρόν βήμα περιγράφεται ως εξής:

$$\Big\langle \mathbb{N},\, \mathbb{M}_{256\times256}\big(\{0,1,\ldots,255\}\big)\Big\rangle \longrightarrow \Big\{\Big\langle \mathbb{N},\, i,\, \mathbb{M}_{\alpha\times\beta}\big(\{0,1,\ldots,255\}\big)\Big\rangle : i = 1,2,\ldots,n\Big\}$$

όπου $n \in \mathbb{N}$ είναι το πλήθος των μερών στα οποία θα αποσυντεθεί η εικόνα και $\alpha, \beta \in \mathbb{N}$ είναι κατάλληλες* διαστάσεις για το μέγεθος της διαμέρισης.

*Θα πρέπει να είναι συμβατές με το μέγεθος του τμήματος το οποίο είναι αποδεκτό από το κρυπτοσύστημα.

Ο παραπάνω τμηματικός διαμερισμός διαχωρίζει την εικόνα σε πλειάδες τριών στοιχείων όπου:

- Στην 1η θέση υπάρχει η χρονική στιγμή.
  [Πρόκειται για το χαρακτηριστικό γνώρισμα της εικόνας το οποίο θα συντελέσει στην ορθή ανάκτησή της μετά την επεξεργασία της).
- Στη 2η θέση υπάρχει το αναγνωριστικό θέσης του τμήματος της εικόνας.
  [Πρόκειται για τη θέση την οποία καταλαμβάνει το μέρος της εικόνας στο κάδρο της. Βοηθάει στην σύνθεση των μερών της εικόνας με τη σωστή (αρχική) σειρά.]
- Στη 3η θέση εμπεριέχεται το μέρος της εικόνας.
  [Πρόκειται για τα εικονοστοιχεία (δηλαδή στοιχεία της δισδιάστατης αναπαράστασης της εικόνας στον ηλεκτρονικό υπολογιστή) τα οποία συνθέτουν ένα μέρος της εικόνας.]

Επεξεργασία των τμημάτων

Αφότου η εικόνα διαμερισθεί σε τμήματα, εκείνα παράσχονται στα προκαθορισμένα RDDs όπου εφαρμόζεται σε παράλληλο χρόνο ο μετασχηματισμός τους (δηλαδή εφαρμογή του κρυπτοσυστήματος).

$$\{\left\langle \mathbb{N},\, i,\, \mathbb{M}_{\alpha\times\beta}\big(\{0,\ldots,255\}\big)\right\rangle : i=1,\ldots,n\} \xrightarrow{\mathcal{E}/\mathcal{D}} \{\left\langle \mathbb{N},\, i,\, \mathbb{M}_{\alpha\times\beta}\big(\{0,\ldots,255\}\big)\right\rangle : i=1,\ldots,n\}$$

Σύνθεση της εικόνας (Reduce)

Τα (απο)κρυπτογραφημένα μέρη συλλέγονται για την παραγωγή του αποτελέσματος (της τροποποιημένης εικόνας). Υπεθυμίζεται πως κάθε μέρος της εικόνας αποτελεί μία τριάδα. Η διαδικασία της επανασύνθεσης έχει ως ακολούθως:

1. Αναγωγή στο κλειδί (reduce by key).

   [Μέρη με ίδια πρώτη συντεταγμένη στην τριάδα ομαδοποιούνται μαζί. Με άλλα λόγια, συσσωματώνονται τα μέρη των εικόνων τα οποία απεικονίζουν την ίδια χρονική στιγμή.]

2. Τα ανακτηθέντα μέρη τοποθετούνται στη θέση την οποία κατείχαν εξ αρχής (πριν την επεξεργασία τους) στην εικόνα.

   [Ελέγχεται η 2η θέση της τριάδας η οποία υποδεικνύει την ορθή θέση στην οποία πρέπει να τοποθετηθεί το επεξεργασμένο μέρος.]

Τέλος, η εικόνα που έχει συντεθεί μεταδίδεται στον ανασφαλή δίαυλο, με σκοπό την λήψη της από τον Receiver.

Επιπλέον μετασχηματισμοί στην περίπτωση ροής

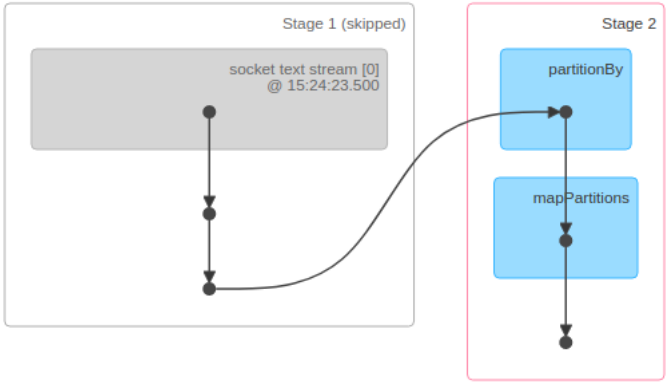
Κατά τη ροή εικόνων, οι παρακάτω διαδικασίας εφαρμόζονται επιπλέον:

1. Ταξινόμηση των εικόνων σε αύξουσα σειρά σύμφωνα με το χαρακτηριστικό τους γνώρισμα.
   [Οι εικόνες οι οποίες είναι έτοιμες για αποστολή προς τον Receiver ταξινομούνται κατά αύξουσα σειρά σε κάθε ένα από τα διαθέσιμα RDDs.]
2. Μετάδοση των εικόνων από όλα τα RDDs.
   [Απαριθμούνται όλα τα διαθέσιμα RDDs και κάθε εικόνα η οποία εμπεριέχεται σε αυτά μεταδίδεται προς τον Receiver.]

Ωστόσο, ο διαμοιρασμός των εικόνων στα RDDs δεν είναι ελέγξιμος. Ως εκ τούτου παρ' όλο που τα RDDs έχουν ταξινομηθεί σε προηγούμενο βήμα, ενδέχεται μια εικόνα να μεταδοθεί σε ύστερη σειρά από μια μεταγενέστερη αυτής (διότι επί παραδείγματι, δεν επιλέχθηκε πρώτα το RDD με την προγενέστερη εικόνα). Ωστόσο, η ταξινόμηση των εικόνων -έστω και σε επίπεδο RDDs- αποτελεί μια προσπάθεια περιορισμού του εν λόγω φαινομένου.

Το διάγραμμα της Εφαρμογής

Το διάγραμμα ροής των δεδομένων εντός της Εφαρμογής είναι το ακόλουθο:



Σχήμα 7.1: Το ακυκλικό κατευθυνόμενο διάγραμμα της Εφαρμογής

Όποτε ανακύπτει ανάγκη για ανακάτεμα (shuffling - βλ. §6.3.3) των δεδομένων, τότε ένα καινούργιο στάδιο δημιουργείται. Τα δεδομένα επεξεργάζονται

σε δύο στάδια: πρώτα (Stage 1) διαβάζονται από τη ροή (DStream) και κατόπιν (Stage 2) διαμερίζονται σε σύνολα δεδομένων (RDDs). Κατά το δεύτερο στάδιο διενεργούνται όλοι οι μετασχηματισμοί στις εικόνες οι οποίες εν τέλει αποστέλλονται στον Receiver. Η διαδικασία του Σχήματος 7.1 επαναλαμβάνεται στον χρόνο από το Spark, το οποίο συλλέγει τα δεδομένα, τα επεξεργάζεται και τα αποστέλλει.

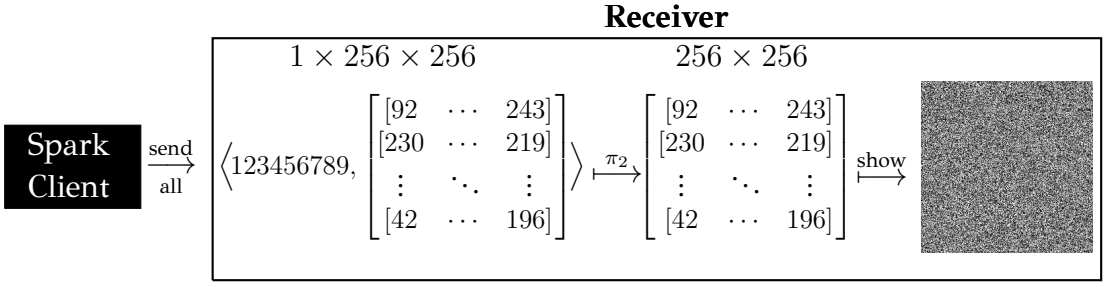
Κρυπτογράφηση/Αποκρυπτογράφηση

Η ανωτέρω διαδικασία χρησιμοποιείται τόσο κατά την κρυπτογράφηση, όσο και κατά την αποκρυπτογράφηση. Η μόνη διαφορά έγκειται στη συνάρτηση του κρυπτοσυστήματος η οποία εφαρμόζεται ανάμεσα στις διαδικασίες του Map και του Reduce.

7.3 Προβολή της ροής (Receiver)

Το τελευταίο βήμα επιφορτίζεται απλώς την εμφάνιση των εικόνων στον αποδέκτη της ροής του Spark. Δεν εφαρμόζονται κάποιοι μετασχηματισμοί. Ο Receiver είναι υπεύθυνος για:

1. Την ολοκληρωτική λήψη του ζεύγους το οποίοι αποστέλλει το Spark.
2. Την προβολή του στον παραλήπτη.



Ο Receiver διατηρεί την πιο ύστερη χρονική στιγμή των εικόνων των οποίων έχει προβάλει. Ο λόγος είναι ότι η κρυπτογράφηση κάποιας εικόνας η οποία προηγείται χρονικά μπορεί να ολοκληρωθεί σε ύστερη στιγμή και να έχει μεταδοθεί ήδη η εικόνα της επόμενη στιγμής. Υπενθυμίζεται πως η κρυπτογράφηση γίνεται εν παραλλήλω, άρα δεν υπάρχει τρόπος το Spark να γνωρίζει ότι πρέπει

να ιεραρχήσει τις εναπομένουσες εργασίες. Επίσης, τα DStreams δεν διαθέτουν (στην παρούσα έκδοση) τρόπο ακύρωσης μιας διεργασίας ((απο)κρυπτογράφηση ενός τμήματος της εικόνας) αν παρέλθει κάποιο χρονικό όριο. Η απόρριψη ενός αποτελέσματος εάν ολοκληρωθεί ύστερα από ένα ορισμένο χρονικό διάστημα καλείται `watermark` και δεν υποστηρίζεται από τη δομή των DStreams. Άρα, για λόγους ανθρώπινης κατανόησης, δεν απεικονίζονται εικόνες οι οποίες ενδέχεται να παρελήφθησαν, ενώ έχουν προβληθεί ύστερες εικόνες. Ωστόσο, ο Receiver δύναται να τροποποιηθεί ώστε η ληφθήσα ροή να αποθηκεύεται σε ένα μέσο στο οποίο σε μεταγενέστερη αναπαραγωγή της ροής, να εμπεριέχει τις ληφθήσες εικόνες ταξινομημένες στις σωστές χρονικές στιγμές.

Η υλοποίηση του Receiver εμπεριέχεται στο Παράρτημα Α.4.

7.4 Υπερπαράμετροι

Κατά την υλοποίηση χρησιμοποιήθηκαν παράμετροι στα 3 μέρη που συνθέτουν την Εφαρμογή. Πρόκειται για τις υπερπαραμέτρους (hyperparameters) οι οποίες καθορίστηκαν με τη μέθοδο δοκιμής-λάθους. Οι εν λόγω παράμετροι δύναται να διαφέρουν/τροποποιηθούν αναλόγως των, παρεχομένων προς το `Spark`, υπολογιστικών δυνατοτήτων. Γι αυτό και συνοψίζονται στην ενθάδε Ενότητα, αρχής γενομένης από:

- τις διαστάσεις των εικόνων.

Μπορεί να επιλεγεί οποιοδήποτε μέγεθος, αρκεί να επαναϋπολογισθούν οι παράμετροι, όπως το πλήθος των διαμερίσεων και ο τρόπος ο οποίος διαμερίζονται οι εικόνες, ώστε να προκύψουν τα σωστά μήκη ανά τμήμα εικόνας.

Streaming Server

Η μόνη υπερπαράμετρος αποτελεί του Streaming Server συνιστά το

(α΄) πλήθος των εικόνων (frames) που αποστέλλονται ανά δευτερόλεπτο (frames per second - fps).

Δεν θα πρέπει να γίνεται κατάχρηση στο εν λόγω πλήθος, καθόσον η εφαρμογή απεικονίζει δεδομένα πραγματικού χρόνου. Συνεπώς, θα πρέπει να προσαρμόζεται αναλόγως της υπολογιστικής δυνατότητας που διαθέτει το `Spark` στον πόρου όπου υλοποιείται δίχως να δημιουργεί συσσωρευόμενες καθυστερήσεις στη μετάδοση.

Spark Client

Ο Spark Client διέπεται από τις εξής υπερπαραμέτρους:

1. Το πλήθος των μερών στα οποία διαμερίζεται η εκάστοτε εικόνα.

   [Για την αποφυγή περιττών παραγεμισμάτων (padding), η κάθε εικόνα θα πρέπει να χωρίζεται σε τμήματα μήκους κάποιου πολλαπλασίου του επιθυμητού μήκους που επεξεργάζεται ο αλγόριθμος. Επί παραδείγματι ο AES ζητεί κείμενα μήκους σε bytes πολλαπλάσιο του 16. Σημειώνεται πως το εν λόγω πλήθος δεν θα πρέπει να είναι 1 (διότι τότε θα χανόταν η έννοια της παράλληλης εφαρμογής αλγορίθμων ανά εικόνα) ούτε πάρα πολύ υψηλό (καθώς τότε το Spark θα ασχολούνταν πάρα πολύ χρόνο με μια εικόνα).]

2. Ο τρόπος που θα διαμεριστεί η εικόνα καθώς και το μήκος του τμήματος.

   [Έγκειται στον τρόπο προσπέλασης των εικονοστοιχείων (ήτοι τη σειρά με την οποία αυτά επιλέγονται). Επίσης, ανά πόσα εικονοστοιχεία συντίθεται ένα τμήμα το οποίο πρόκεται να παρασχεθεί στο κρυπτοσύστημα.]

3. Το μήκος του χρονικού διαστήματος (σε δευτερόλεπτα) όπου το Spark συλλέγει τα δεδομένα τα οποία κατόπιν θα επεξεργαστεί.

   [Πρόκειται για το χρονικό διάστημα κατά το οποίο το Spark "ακούει" (ήτοι συλλέγει δεδομένα από την εισερχόμενη ροή). Τα συλλεχθέντα δεδομένα αποτελούν μία ολότητα και επεξεργάζονται ομού, δηλαδή το Spark τα θεωρεί ως ένα σύνολο δεδομένων και δημιουργεί RDDs για την επεξεργασία τους (βλ. §6.3.2). Κατόπιν, το Spark συλλέγει δεδομένα από τη ροή κατά το επόμενο χρονικό διάστημα και ούτο καθ' εξής. Ο χρόνος θα πρέπει να επιλεγεί αναλόγως των υπολογιστικών δυνατοτήτων του μηχανήματος στο οποίο υλοποιείται το Spark για να μην παύσουν τα μεταδιδόμενα δεδομένα να απεικονίζουν την παρούσα χρονική στιγμή.]

4. Το πλήθος των ολοκληρωμένων μερών που απαιτούνται ώστε να αποσταλλεί η εικόνα.

   [Χάριν της πιο άμεσης μετάδοσης των εικόνων (αφού πρόκειται για εφαρμογή πραγματικού χρόνου, μπορεί να ζητηθεί να αποσταλλούν εικόνες οι οποίες το πλήθος των μερών τους που έχει κρυπτογραφηθεί να υπερβαίνει κάποιο κατώφλι. Κατ' αυτόν τον τρόπο, δύναται οι εικόνες να μεταδίδονται, δίχως να έχουν κρυπτογραφηθεί όλα τους τα μέρη ώστε να μην περιμένουν κάποιον δύσκολο υπολογισμό τους να περατωθεί. Αφ' ενός χάνεται πληροφορία, αφ' ετέρου επιτυγχάνεται το σύγχρονο της εικόνας που μεταδίδεται. Με μια εμπεριστατομένη διαμέριση κι ένα αποδεκτό

κατώφλι, μπορεί οι εικόνες να μπορούν να μεταδίδονται δίχως να απολύεται μεγάλο μέρος της πληροφορίας τους.]

5. Ο αλγόριθμος (απο)κρυπτογράφησης.

   [Μπορεί να χρησιμοποιηθεί οποιοσδήποτε από τους αλγορίθμους οι οποίοι εμπεριέχονται στο Μέρος I (κι όχι μόνον).]

Receiver

Η μόνη υπερπαράμετρος για τον Receiver, αποτελεί η

(i) χρονοκαθυστέρηση (σε χιλιοστά του δευτερολέπτου) ανάμεσα στις προβαλλόμενες εικόνες (ήτοι το πλήθος των εικόνων ανά δευτερόπλεπτο).

Η χρονοκαθυστέρηση μπορεί να παραληφθεί, ωστόσο μια τέτοια επιλογή θα πρόβαλε τις εικόνες αμέσως μόλις αποστέλλονταν από το `Spark`, δίχως μια σταθερή διαφορά μεταξύ τους, γεγονός που θα δυσχαίρενε την ανθρώπινη κατανόηση (καθώς δεν θα αποτελούσε μια ομαλά προβαλλόμενη ροή). Συνεπώς, η ενθάδε παράμετρος εξυπηρετεί έναν πιο αφηρημένο σκοπό (εκείνον της ανθρώπινης κατανόησης) κι όχι έναν αμοιγώς προγραμματιστικό σκοπό. Ωστόσο, ούσα εφαρμογή δεδομένων πραγματικού χρόνου, η παράμετρος δεν θα πρέπει να δημιουργεί χρονοκαθυστερήσεις που θα συσωρεύονται συν τω χρόνω καθιστώντας έτσι μεγάλες διαφορές ανάμεσα στα εν εξελίξει και στα προβαλλόμενα δεδομένα.

Τέλος, γίνεται λόγος για δύο επιλογές υπερπαραμέτρων του Spark Client οι οποίες δύναται να επηρεάσουν τον Receiver:

- Το κατώφλι των κρυπτογραφημένων τμημάτων που απαιτούνται για την μετάδοση μιας εικόνας προς τον Receiver.

Εάν επιτραπεί η μετάδοση μέρους των εικόνων (συγκεκριμένα των μερών τα οποία έχουν κρυπτογραφηθεί), τότε ούσα ανολοκλήρωτη η εικόνα, θα πρέπει (είτε στον Spark Client είτε στον Receiver) να ληφθεί πρόνοια για το παραγέμισμα των ελλειπόντων μερών (επί παραδείγματι με μηδενικά στοιχεία), ώστε να προκύψει μια ολοκληρωμένη εικόνα, παρ' όλο που θα αποδίδει μέρος της πληροφορίας της.

- Ο τρόπος διαμέρισης της εικόνας.

Ο διαχωρισμός είτε μόνον κατά γραμμές είτε μόνον κατά στήλες μιας εικόνας στην `Python`, υποστηρίζει μια απλή -και σχεδόν άμεση- επανασύνθεση της εικόνας από τα μέρη της. Μια πιο εμπεριστατωμένη μέθοδος διαμέρισης, θα πρέπει να επιφορτήσει είτε τον Spark Client, είτε τον Receiver με την ορθή επνατοποθέτηση των ((απο)κρυπτογραφημένων πλέον) μερών της εικόνας.

7.5 Πιλοτική εφαρμογή

Τα ανωτέρω υλοποίηθηκαν με τις εξής επιλογές:

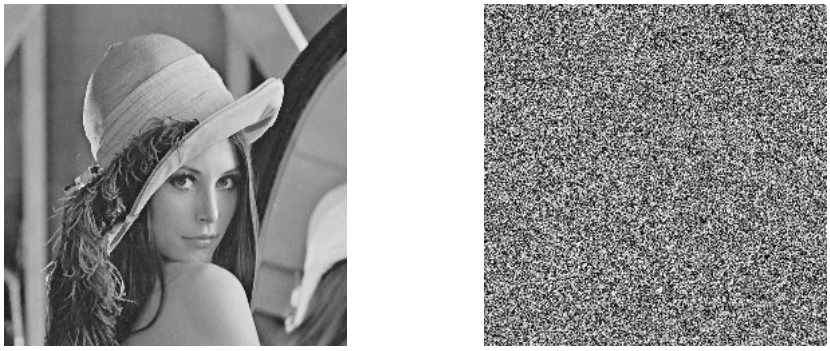
1. Κρυπτοσύστημα: AES με ECB mode.
2. Μέγεθος εικόνων: 256×256.
3. Υπερπαράμετροι:
   - Streaming Server:
     - Πλήθος μεταδιδόμενων εικόνων ανά δευτερόλεπτο: ~ 10 (χρονοκαθυστέρηση 0.2 δευτερολέπτων ανάμεσα σε διαδοχικές εικόνες).
   - Spark Client:
     - Χρονικό διάστημα συλλογής δεδομένων: ανά 0.5 δευτερόλεπτα.
     - Μήκος τμήματος και τρόπος προσπέλασης: οι εικόνες προσπελάστηκαν γραμμή προς γραμμή. Το εκάστοτε τμήμα είναι μήκος 2048 εικονοστοιχεία.
     - Κατώφλι μετάδοσης εικόνας: 32 μέρη, δηλαδή μεταδίδονται μόνον οι πλήρως (απο)κρυπτογραφημένες εικόνες, αφού
     $$\frac{\text{μήκος} \times \text{πλάτος εικόνας}}{\text{μήκος τμήματος}} = \frac{256 \times 256}{2048} = 32$$
   - Receiver:
     - Πλήθος προβαλλόμενων εικόνων ανά δευτερόλεπτο: ~ 20 (χρονοκαθυστέρηση 0.1 δευτερολέπτων ανάμεσα σε διαδοχικές εικόνες).

Κατά τις δοκιμές υλοποίηθηκαν τα εξής σενάρια:

Σενάριο Κρυπτογράφησης: Ο Streaming Server καταγράφει τις εικόνες και τις στέλνει στον Spark Client ο οποίος τις κρυπτογραφεί.

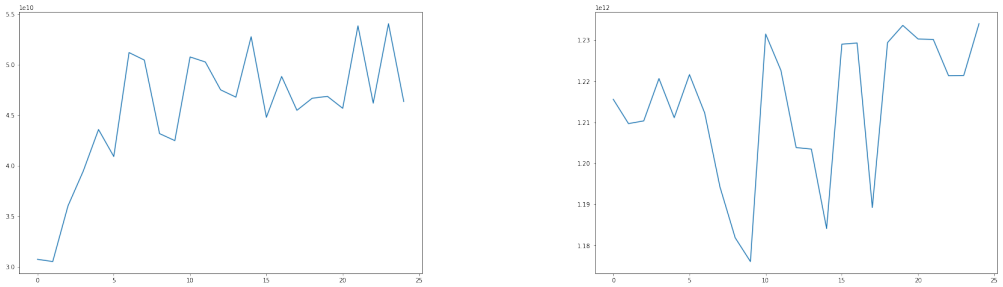
Σενάριο Αποκρυπτογράφησης: Ο Streaming Server καταγράφει τις εικόνες, τις κρυπτογραφεί και κατόπιν τις στέλνει στον Spark Client ο οποίος τις αποκρυπτογραφεί.

Η εικόνα η οποία χρησιμοποιήθηκε κατά τις δοκιμές, καθώς και η κρυπτογράφηση που προέκυψε από τον AES εμφανίζονται στο Σχήμα 7.2.



Σχήμα 7.2: Κρυπτογράφηση της εικόνας `lena.jpg` σύμφωνα με το κρυπτοσύστημα AES χρησιμοποιώντας τον ECB τρόπο προσπέλασης με κλεδί την 16 bytes ακολουθία (στη δεκαεξαδική της μορφή): `969f449f7fa5fee36c7ec7df40540bd7`

Η Εφαρμογή υλοποιήθηκε σε φορητό ηλεκτρονικό υπολογιστή, με 8 πυρήνες Intel® Core™ i7-8550U. Σύμφωνα με τις ρυθμίσεις οι οποίες εμπεριέχονται στην παρούσα Ενότητα, η χρονοκαθυστέρηση όταν η Εφαρμογή κρυπτογραφούσε για 30 λεπττά βρέθηκε να έχει παράγοντα $\frac{10^{12}}{5 \times 10^{10}} = 20$ (εκ των κλιμάκων των διαγραμμάτων κατωτέρω).



(α') Αρχή της Εφαρμογής (β') Κατόπιν 30 λεπτών υλοποίησεως

Σχήμα 7.3: Χρονοκαθυστερήσεις στην προβολή των εικόνων

Συνεπώς, οι εικόνες οι οποίες προβάλλονταν ύστερα από 30 λεπτά είχαν χρονοκαθυστέρηση 20πλάσια απ' όσο είχαν οι αρχικές εικόνες.

Μέρος III

Επίλογος

Κεφάλαιο 8

Επίλογος

Δεδομένου του νόμου του Moore [Mo65], πως το πλήθος των κρυσταλοτριόδων (transistors) σε ένα ολοκληρωμένο κύκλωμα διπλασιάζεται σχεδόν κάθε δύο έτη, αλλά και του γεγονότος πως συν τω χρόνω καινούργιες τεχνικές και αλγόριθμοι ανακύπτουν καθιστώντας άλυτα προβλήματα του παρελθόντος πιο ευεπίλυτα, κάθε κρυπτοσύστημα τίθεται υπό διαρκή αμφισβήτηση. Παράδειγμα αποτελεί η αντικατάσταση του DES -ο οποίος αποτέλεσε πρότυπο για σχεδόν 30 έτη- από τον AES.

Σύνοψη

Κατά την Εργασία προσεγγίστηκε το πρόβλημα της ασφαλούς μετάδοσης μιας ροής εικόνων (βίντεο) μέσω ενός ανασφαλούς διαύλου (επί παραδείγματι ασύρματα). Πρωτίστως, δημιουργήθηκε ένας μηχανισμός δημιουργίας (Streaming Server) και ένας μηχανισμός προβολής της ροής (Receiver). Κατά το ενδιάμεσο παρενεβλήθει μία λογική διακριτοποίησης και παράλληλης επεξεργασίας της ροής, εικόνα προς εικόνα (Spark Client). Κάθε εικόνα (απο)κρυπτογραφήθηκε ξεχωριστά και αντιμετωπίστηκε ως μεγάλο δεδομένο, συνεπώς, χρησιμοποιήθηκαν εργαλεία κατάλληλα για τη διαχείρηση μεγάλων δεδομένων. Παρουσιάσθηκε μία πτυχή της Κρυπτογραφίας η οποία περιέχει αλγορίθμους οι οποίοι αφενός προσφέρονται για την κρυπτογράφηση εικόνων, αφετέρω η φύση τους επιτρέπει την παράλληλη διεξαγωγή υπολογισμών. Ως εφαρμογή όλων των ανωτέρω παρετέθει μία υλοποίηση της διαδικασίας μέσω ενός πρωτοκόλλου διαχείρησης της ροής των εικόνων. Η διαδικασία εφαρμόζεται τόσο κατά την κρυπτογράφηση, όσο και κατά την αποκρυπτογράφηση της ροής, με μόνη διαφορά στη συνάρτηση του κρυπτοσυστήματος η οποία εφαρμόζεται αναλόγως του σκοπού.

Όπως ήδη έχει αναφερθεί στην Εισαγωγή της Εργασίας, αλλά και όπως θα παρατεθεί κατωτέρω, καθίσταται σαφές πως οι κλάδοι της Κρυπτογραφίας και των Μεγάλων Δεδομένων είναι διαρκώς εξελισσόμενοι. Έτσι, ακόμη κι αν οι κρυπταλγόριθμοι ή/και τα προγράμματα πολυνηματικής διαχείρησης αλλάξουν, η Εργασία προσπάθησε να αποδόσει μια μεθοδολογία αντιμετώπισης της κρυπτογράφησης εικόνων. Ωστόσο, η εξυπηρέτηση του σκοπού χρειάστηκε κάποιες παραδοχές, όπως να επιλεγεί μια γλώσσα προγραμματισμού και ένα πρόγραμμα πολυνηματικής διαχείρησης για να επιτευχθεί η υλοποίηση.

Μια αέναη εξέλιξη

Στην παρούσα Εργασία επιχειρήθηκε να δοθεί μία μεθοδολογία, ανεξάρτητη του κρυπταλγορίθμου ο οποίος προτιμάται. Ωστόσο η λύση προσφέρθηκε έχοντας ως βάση τη χρήση του Spark, ως γλώσσα υλοποίησης την Python και σκοπώντας μόνον αλγορίθμους τμήματος και χαοτικούς. Σημειώνονται τα εξής:

1. Τα προγράμματα διαχείρησης μεγάλων δεδομένων εξελίσσονται ή/και αντικαθίστανται.

Πρόγονο του Spark, αποτελεί ένα άλλο πρόγραμμα διαχείρησης δεδομένων, το Hadoop [Ha] (1η έκδοση: Απρίλιος 2006 [HR]). Το Hadoop αποτελεί βάση για τον τρόπο διαχείρησης των αρχείων (Hadoop Distributed File System - HDFS) τα οποία περιέχουν τα δεδομένα του Spark. Όπως το Spark πλέον χρησιμοποιεί το Hadoop, κατά τρόπο παρόμοιο μπορεί να ανακύψει κάποιο νέο πρόγραμμα διαχείρησης μεγάλων δεδομένων, ή μια μεταγενέστερη έκδοση του Spark να προσφέρει αποτελεσματικότερα εργαλεία.

2. Οι γλώσσες προγραμματισμού εξελίσσονται.

Η Python (έκδοση 3.6.9) προσέφερε γρήγορη πρόσβαση σε εικόνες, μετατροπή τους σε πίνακες και σύνθεσης/αποσύνθεσης των εν λόγω πινάκων. Νέες επεκτάσεις στις ήδη υπάρχουσες γλώσσες προγραμματισμού ανακύπτουν, καθώς επίσης για οι τάσεις χρήσης των γλωσσών προγραμματισμού διαμορφώνονται αναλόγως των δυνατοτήτων που προσφέρει η κάθε γλώσσα.

3. Τα κρυπτοσυστήματα αναπτύσσονται/εξελίσσονται/αντικαθίστανται.

Κατά το ρουν της Εργασίας, παρουσιάσθηκαν μερικά μόνο κρυπτοσυστήματα (τμηματικοί και χαοτικοί αλγόριθμοι) τα οποία άπτονται στη φύση των παράλληλων υπολογισμών, τα οποία δεν εξαντλούν τις επιλογές των κρυπτοσυστημάτων που προσφέρονται για παράλληλη κρυπτογράφηση.

Μια παραλλαγή της προτεινόμενης εφαρμογής

Η υλοποίηση η οποία παρουσιάσθηκε στο Κεφάλαιο 7 μπορεί να παραλλαχθεί περαιτέρω. Το κύριο μέρος της έγκειται στην υλοποίηση του Spark Client, όπου ως απεικόνιση αναπαρίσταται:

$$\text{SparkClient} : \Big\langle \mathbb{N},\, \mathbb{M}_{256\times 256}\big(\{0,1,\ldots,255\}\big) \Big\rangle \longrightarrow \Big\langle \mathbb{N},\, \mathbb{M}_{256\times 256}\big(\{0,1,\ldots,255\}\big) \Big\rangle$$

δηλαδή η είσοδος και η έξοδος του αλγορίθμου είναι της ίδιας μορφής. Συνεπώς, κάθε βήμα ώστε να προκύψει ο μετασχηματισμός μπορεί να τροποποιηθεί, δίχως η μορφή της εξόδου να επηρεάζεται προδίδοντας έτσι την αλλαγή.

Σε μια προσπάθεια να αυξηθεί η ασφάλεια της Εφαρμογής, μία παραλλαγή αποτελεί η χρήση του AES με Block Chainning (CBC) επιλογή. Έτσι το κάθε μέρος της εικόνας θα κρυπτογραφείται και θα εξαρτάται από τα προηγούμενα blocks, τα οποία όμως έχουν δεν έχουν επιλεγεί γραμμικά (γραμμή ανά γραμμή), αλλά σύμφωνα με μια συνάρτηση διαμέρισης της εικόνας. Έτσι, διατηρώντας μυστικό τον τρόπο με τον οποίο διαμερίζεται η εικόνα, προκύπτει ένα πιο εξειδικευμένο κρυπτοσύστημα. Ένας (ακόμη) συνδυασμός χαοτικής κρυπτογραφίας με τον AES μπορεί να αποτελέσει το γεγονός πως η συνάρτηση επιλογής των εικονοστοιχείων διαπράττει επιλογές σύμφωνα με μια γεννήτρια ψευδοτυχαίων αριθμών από ένα χαοτικό σύστημα εξισώσεων. Εν τοιαύτη περιπτώσει, οι συντελεστές του συστήματος αποτελούν μέρος του συμμετρικού κλειδιού.

Άλλες πτυχές της παράλληλης Κρυπτογραφίας

Μία άλλη έκφανση της Κρυπτογραφίας αποτελούν τα συστήματα ταυτοποίησης, όπου μία οντότητα (αποδεικνύων - prover) προσπαθεί να πείσει μια άλλη οντόντητα (επιβεβαιωτής - verifier) για το γνήσιό της. Τα σχήματα ταυτοποίησης εντάσσονται στις Αποδείξεις Μηδενικής Γνώσης. Ο αποδεικνύων καλείται να πείσει τον επιβεβαιωτή -μέσα από μια σειρά προκλήσεων (challenge) που του αναθέτει ο επιβεβαιωτής- πως είναι κάτοχος μιας πληροφορίας, δίχως όμως να αποκαλύψει την εν λόγω πληροφορία. Η ταυτοποίηση μπορεί να γίνει σε παράλληλο χρόνο ως εξής:

1. Ο επιβεβαιωτής στέλνει όλες τις προκλήσεις του.

2. Ο αποδεικνύων τις επιλύει (εάν είναι δυνατόν, σε παράλληλο χρόνο) και στέλνει τις απαντήσεις του πίσω στον επιβεβαιωτή.

3. Ο επιβεβαιωτής ελέγχει την ορθότητα των απαντήσεων σε παράλληλο χρόνο ως εξής:

    - Κάθε απάντηση προσπελαύνεται ξεχωριστά σε παράλληλες διεργασίες.
    - Αν μία διεργασία εντοπίσει ένα λάθος, τότε ενημερώνει τις άλλες διεργασίες να σταματήσουν τους ελέγχους που διεξάγουν και ο επιβεβαιωτής αρνείται την παραχώρηση πρόσβασης στον αποδεικνύοντα.

    Αν έχουν περατωθεί όλες οι παράλληλες διεργασίες, τότε όλοι οι έλεγχοι έχουν ολοκληρωθεί επιτυχώς και ο αποδεικνύων πείθεται για το γνήσιο του επιβεβαιωτή.

Μια υλοποίηση μπορεί να συναχθεί με χρήση της γλώσσας προγραμματισμού Java, η οποία υποστηρίζει (έκδοση 11) πολυνηματική διαχείριση (multi-thread tasking) καθώς και τη δυνατότητα ένα νήμα να διακόπτει τη ροή ενός ή περισσοτέρων άλλων νήματος/ων.

Παράρτημα

Παράρτημα Α

Πηγαίος κώδικας

Στις επόμενες Ενότητες εμπεριέχεται ο κώδικας σε Python του Κεφαλαίου 7.

Α.1 Υπερπαράμετροι

Οι υπερπαράμετροι του συστήματος, όπως εκείνο δοκιμάστηκε κατά την §7.5:

```
# General settings
DEBUG_MODE = False
ENCRYPT_MODE = True # True stands for ENCRYPT, False stands for DECRYPT
image_size = (256, 256)
key = b'\x96\x9fD\x9f\x7f\xa5\xfe\xe3l~\xc7\xdf@T\x0b\xd7'
#key = Random.new().read(16)
spark_timeout = 60 # in seconds
verbose = 50 # per images

# Spark Client
segment_length = 2048
transmition_threshold = 32
stream_duration = 0.5

# Streaming Server
sleep_between_frames = 0.2

# Receiver
sleep_between_receives = 10
```

A.2 Streaming Server

Ο πηγαίος κώδικας για τα περιεχόμενα της §7.1:

```
import socket
from time import sleep
import time
import cv2
import socketserver
import numpy as np
from PIL import Image
from Crypto.Cipher import AES

from constants import *



class AlphaTCPHandler(socketserver.BaseRequestHandler):
    def handle(self):
        if not DEBUG_MODE:
            cam = cv2.VideoCapture(0)

        lines_sent = 0

        try:
            while True:
                if DEBUG_MODE:
                    img = Image.open('lena.jpg') # For testing
                    out = np.array(img) # For testing
                else:
                    ret, img = cam.read()
                    if not ret:
                        continue # Frame was not captured
                    out = cv2.cvtColor(cv2.resize(img, image_size),\
                        cv2.COLOR_BGR2GRAY)
                now = str(time.time()).replace('.','')
                now += ''.join((20 - len(now)) * ['0'])

                if not ENCRYPT_MODE:
                    enc = []
                    for row in out:
```

```
                        enc.append([int(z) for z in bytearray(AES\
                            .new(key, AES.MODE_ECB)
                            .encrypt(bytes([int(x) for x in row])))])
                    out = enc
                else:
                    out = out.tolist()

                b = bytes(str((now, out)) + '\n', 'utf-8')
                self.request.sendall(b)

                lines_sent += 1
                time.sleep(sleep_between_frames)
        except BrokenPipeError:
            print('broken pipe detected')
            print(lines_sent, "requests were sent.\n")
            lines_sent = 0

            if not DEBUG_MODE:
                cam.release()


if __name__ == '__main__':
    host = '127.0.0.1'
    port = 9999

    server = socketserver.TCPServer((host, port), AlphaTCPHandler)
    print(f'server starting {host}:{port}')
    server.serve_forever()
```

A.3 Spark Client

Ο πηγαίος κώδικας για τα περιεχόμενα της §7.2:

```
import findspark
findspark.init()

from pyspark import SparkContext, SparkConf
from pyspark.streaming import StreamingContext
```

```
from Crypto.Cipher import AES
from Crypto import Random
import socket
import numpy as np
import pandas as pd

from constants import *

def encrypt(plaintext):
    return [str(int(x)) for x in bytearray(AES.new(key, AES.MODE_ECB)\
        .encrypt(bytes(plaintext)))]

def decrypt(ciphertext):
    return [str(int(x)) for x in bytearray(AES.new(key, AES.MODE_ECB)\
        .decrypt(bytes(ciphertext)))]


def mapToTriplets(line):
    ts, img = eval(line)
    img = np.array(img).astype('int').reshape(-1, segment_length)
    return [(ts, i, list(j)) for i, j in enumerate(img)]


def doCipherFinal(keyValueTriplet):
    if ENCRYPT_MODE:
        return (keyValueTriplet[0],\
            {keyValueTriplet[1]: encrypt(keyValueTriplet[2])})
    else:
        return (keyValueTriplet[0],\
            {keyValueTriplet[1]: decrypt(keyValueTriplet[2])})


def gatherToDict(x, y):
    for k, v in y.items():
        x[k] = v
    return x


def dictToImage(p):
    return (p[0], np.array(pd.DataFrame(p[1]).T)\
        .reshape(image_size[0],-1).tolist())
```

```
def transmition(img):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.connect(('127.0.0.1', 12345))
        sock.sendall(bytes((str(img) + '\n').encode('utf-8')))
        sock.close()
    except:
        pass


if __name__ == '__main__':
    sc = SparkContext("local[*]", "CryptoStreaming")
    # Capture every stream_duration seconds
    ssc = StreamingContext(sc, stream_duration)

    # Create a DStream that will connect to hostname:port, like localhost:9999
    ds = ssc.socketTextStream("localhost", 9999)

    ds = ds.flatMap(mapToTriplets)

    ds = ds.map(doCipherFinal)

    ds = ds.reduceByKey(gatherToDict)

    ds = ds.filter(lambda x: len(x[1]) >= transmition_threshold)

    ds = ds.map(dictToImage)

    # Sort stream by key
    ds = ds.transform(lambda rdd: rdd.sortBy(lambda x: x[0], ascending=True))

    # Transmit the (en/de-crypted) image
    ds.foreachRDD(lambda rdd: rdd.foreach(transmition))

    ssc.start() # Start the computation

    if DEBUG_MODE:
        # Terminate after x seconds of collecting data,
        # for demonstrating data in html format
        ssc.awaitTerminationOrTimeout(10)
```

```
    else:
        # Wait for the computation to terminate
        ssc.awaitTermination()

    ssc.stop()
```

A.4 Receiver

Ο πηγαίος κώδικας για τα περιεχόμενα της §7.3:

```
import socketserver
import socket
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import cv2
import time

from constants import *


class AlphaTCPHandler(socketserver.BaseRequestHandler):
    def handle(self):
        global latest

        received = ''
        while '\n' not in received:
            received += str(self.request.recv(8192), 'utf-8')
        ts, img = eval(received)

        if verbose > 0 and len(latencies) >= verbose:
            plt.plot(latencies)
            plt.show()
            latencies = []

        if not latest or latest < ts:
            latest = ts
        else:
            return # This is a past moment
```

```
        cv2.imshow('CryptoStreaming', np.array(img).astype('uint8'))
        cv2.waitKey(sleep_between_receives)


if __name__ == '__main__':
    cv2.destroyAllWindows()
    cv2.namedWindow("CryptoStreaming")
    host = '127.0.0.1'
    port = 12345
    latest = ''

    latencies = []
    plt.rcParams["figure.figsize"] = (12, 8)

    server = socketserver.TCPServer((host, port), AlphaTCPHandler, False)
    server.allow_reuse_address = True
    server.server_bind()
    server.server_activate()
    print(f'server starting {host}:{port}')
    server.serve_forever()
```

Παράρτημα Β

Γραφικές παραστάσεις

Β.1 Το σύστημα Lorenz

Ο κώδικας σε Octave [Oct] ο οποίος παράγει το σύστημα Lorenz του Σχήματος 5.1 είναι ο κάτωθι:

```
function dx = lorenz(x, param)
    dx = [
        param(1) * ( x(2) - x(1) ),
        x(1) * ( param(2) - x(3) ) - x(2),
        x(1) * x(2) - param(3) * x(3)
    ];
end

x0 = [0, -1, 1.05];
param = [10, 28, 8/3];

dt = 0.01;
steps = 5000;

x = zeros(steps, 3);
x(1,:) = x0;

for i = 2 : steps
    x(i, :) = x(i - 1, :) + dt * lorenz(x(i - 1, :), param)';
end
```

```
clf;
plot3(x(:,1), x(:,2), x(:,3), 'b', 'LineWidth', 1.5);
rotate3d on
```

B.2 Η μονοδιάστατη λογιστική απεικόνιση

Ο κώδικας σε Python ο οποίος παράγει το σύστημα Lorenz του Σχήματος 5.1 είναι ο κάτωθι [Ro18]:

```
import numpy as np
import matplotlib.pyplot as plt

def logistic(mu, x):
    return mu * x * (1 - x)

n = 10000
r = np.linspace(0, 4.0, n)
iterations = 2000
last = 100
x = 1e-5 * np.ones(n)
plt.figure(num=None, figsize=(10, 8), dpi=80, facecolor='w', edgecolor='k')

for i in range(iterations):
    x = logistic(r, x)
    if i >= (iterations - last):
        plt.plot(r, x, ',k', alpha=.1)
```

Βιβλιογραφία

[ABK98] R. Anderson, E. Biham, L. Knudsen, *Serpent: A New Block Cipher Proposal*, διαθέσιμο στον https://www.cl.cam.ac.uk/ rja14/Papers/serpent0.pdf.

[ABK98] R. Anderson, E. Biham, L. Knudsen, *Serpent: A New Block Cipher Proposal*, International Workshop on Fast Software Encryption, σελ. 222–238 (1998).

[AEMR10] K. Atighehchi, A. Enache, T. Muntean, G. Risterucci, *An Efficient Parallel Algorithm for Skein Hash Functions*, Cryptology ePrint Archive: Report 2010/432 (2010).

[AES] https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development.

[Al18] R. M. N. Aldahdooh, *Parallel Implementation and Analysis of Encryption Algorithms*, Al-Azhar University-Gaza Deanship of Postgraduate Studies Faculty of Engineering & Information Technology Master of Computing & Information Systems, Master thesis (2018), διαθέσιμο στον: https://www.researchgate.net/publication/324747960_Parallel_Implementation_and_Analysis_of_Encryption_Algorithms.

[AL06] G. Alvarez, S. Li, *Some Basic Cryptographic Requirements for Chaos-Based Cryptosystems*, International Journal of Bifurcation and Chaos, 16(8), σελ. 2129–2125 (2006).

[Ap] https://apache.org/.

[AR16] Y. H. Ali, H. A. Ressan, *Image Encryption Using Block Cipher Based Serpent Algorithm*, Engineering and Technology Journal, vol. 34, Issue 2 Part (B) Scientific, σελ. 278–286 (2016).

[ASA11] A. Akhavan, A. Samsudin, A. Akhshani, *A symmetric image encryption scheme based on combination of nonlinear chaotic maps*, Journal of the Franklin Institute. 348 (8), 1797–1813 (2011).

[ASKC03] R. Acharya, P. Subbanna, S. Kumarc, L. Choo, *Transmission and storage of medical images with patient information*, Comput. Biol. Med. 33, 303–310 (2003).

[Ba17] N. Bardis, *Secure Implementation of Modular Arithmetic Operations for IoT and Cloud Applications*, Green IT Engineering: Components, Networks and Systems Implementation σελ.43–64 (2017).

[BA92] N. Bourbakis, C. Alexopoulos, *Picture data encryption using scan patterns*, Pattern Recognition. 25 (6), σελ. 567–581 (1992).

[BAMA08] S. Behnia, A. Akhshani, H. Mahmodi, A. Akhavan, *A novel algorithm for image encryption based on mixture of chaotic maps*. Chaos, Solitons & Fractals, 35 (2): 408–419 (2008).

[BD09] A. Bienner, B. Dequidt, *Parallel implementations of MD6, a modern cryptographic hash function* (2009) διαθέσιμο στον: `http://moais.imag.fr/membres/jean-louis.roch/perso_html/transfert/2009-06-19-IntensiveProjects-M1-SCCI-Reports/BiennerDequidt_en.pdf`.

[BK19] S. L. Brunton, J. N. Kutz, *Data-Driven Science and Engineering, Machine Learning Dynamical Systems, and Control*, 1st ed., ISBN: 978-1108422093 (2019).

[BM97] M. Bellare, D. Micciancio, *A New Paradigm for Collision-free Hashing: Incrementality at Reduced Cost*, In Eurocrypt '97, vol. 1233 of LNCS, Springer-Verlag, σελ. 163–192(1997).

[BMMH10] E.S. A.-M. ElBadawy, W. A. El-Masry, A. Mokhtar, A. E.-D. S. Hafez, *A New Chaos Advanced Encryption Standard (AES) Algorithm for Data Security*, The International Conferences on Signals and Electronic Systems (2010).

[Da89] I. Damgård, *A Design Principle for Hash Functions*, In Crypto '89, vol. 435 of LNCS, σελ. 416–427, Springer-Verlag, (1989).

[DH76] W. Diffie, M. E. Hellman, *New directions in Cryptograhpy*, IEEE Transactions on Information Theory **22**, σελ. 644–654 (1976).

[DH77] W. Diffie, M. E. Hellman, *Exhaustive Cryptanalysis of the NBS Data Encryption Standard*, Computer. 10 (6), σελ. 74–84 (1977), doi:10.1109/C-M.1977.217750.

[DS] E. Paul, *What is Digital Signature – How it works, Benefits, Objectives, Concept*, EMP Trust HR διαθέσιμο στον: https://www.emptrust.com/blog/benefits-of-using-digital-signatures.

[DStr] https://spark.apache.org/docs/latest/streaming-programming-guide.html.

[Dw01] M. Dworkin, *Recomendation for Block Cipher Modes of Operation - Methods and Techniques*, NIST (2001), διαθέσιμο στον: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf.

[Fe73] H. Feistel, *Cryptography and Computer Privacy*, Scientific American. 228 (5), σελ. 15–23 (1973).

[FIPS46-3] Federal Information Processing Standards Publication 46-3, *Data Encryption Standard*, διαθέσιμο στον: https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf

[FIPS197] Federal Information Processing Standards Publication 197, *Advanced Encryption Standard*, (2001), διαθέσιμο στον: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[FIPS202] Federal Information Processing Standards Pulbication 202, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, (2005), Processing Standards Publication 202, διαθέσιμο στον: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.

[Ha] https://hadoop.apache.org/.

[HR] https://archive.apache.org/dist/hadoop/common/, Apache Software Foundation.

[Iz15] P. O. Izevbizua, *Data security in the cloud using serpent encryptionand distributed steganography*, Int. J. Appl. Sci. Math., vol. 1, no. 8, σελ. 347–359 (2015).

[KDB10] S. Al-Kuwari, J. H. Devenport, R. J. Bradford, *Cryptographic Hash Functions: Recent Design Trends and Security Notions*, Proceedings of 6th China International Conference on Information Security and Cryptology (Inscrypt '10). Science Press of China, σελ. 133–150 (2010).

[Ki16] Α. Κιαγιάς, *Cryptography, Primitives an Protocols*, σημειώσεις από το ομότιτλο μάθημα στο Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, 2016.

[KMA05] L. Kocarev, J. Makraduli, P. Amato, *Public-Key Encryption Based on Chebyshev Polynomials*, Circuits, Systems and Signal Processing, 24 (5), σελ. 497–517, (2005).

[KSFV04] L. Kocarev, M. Sterjev, A. Fekete, G. Vattay *Public-key encryption with chaos*, Chaos: An Interdisciplinary Journal of Nonlinear Science, 14 (4), σελ. 1078–1082 (2004).

[Lo63] E. Lorenz, *Deterministic Nonperiodic Flow*, Journal of the Atmoshperic Sciences, vol. 20, σελ. 130–141 (1963).

[LZLCD18] Y. Luo, R. Zhou, J. Liu, Y. Cao, X. Ding, *A parallel image encryption algorithm based on the piecewise linear chaotic map and hyper-chaotic map*, Nonlinear Dynamics, 93 (5), Springer (2018).

[Ma89] R.A.J. Matthews, *On the derivation of a "chaotic" encryption algorithm*, Cryptologia 13, no. 1, σελ. 29–42 (1989).

[MC05] Y. Mao, G. Chen, *Handbook of Geometric Computing*, Springer Berlin Heidelberg, σελ. 231–265 (2005).

[MD6] R. L. Rivest, D. Bailey, S. Cheng, C. Crutchfield, Y. Dodis, E. Fleming, A. Khan, J. Krishnamurthy, Y. Lin, L. Reyzin, E. Shen, J. Sukha, E. Tromer, Y. L. Yin, Juniper Networks, Cilk Arts, National Science Foundation, *The MD6 Hash Function*, Crypto 2008, (2008).

[Mi85] J. Milnor, *On the concept of attractor*, Communications in Mathematical Physics. 99 (2), σελ. 177–195 (1985).

[MMB10] A. G. Marco, A. S. Martinez, O. M. Bruno, *Fast, parallel and secure cryptography algorithm using Lorenz's attractor*, International Journal of Modern Physics C, vol. 21 (3), σελ. 365–382 (2010).

[MOV01] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography (Fifth ed.)*, (2001) ISBN 978-0849385230.

[Mo65] G. E. Moore, *Cramming more components onto integrated circuits*, intel.com, Electronics Magazine (1965), διαθέσιμο στον: `https://newsroom.intel.com/wp-content/uploads/sites/11/2018/05/moores-law-electronics.pdf`.

[MS07] S. Manuel, N. Sendrier, *XOR-Hash: A Hash Function Based on XOR*, In WEWRC '07, (2007).

[MYI12] O. Mirzaei, M. Yaghoobi, H. Irani, *A new image encryption method: parallel sub-image encryption with hyper chaos*, Nonlinear Dynamics vol. 67, σελ. 557–566 (2012).

[Na09] Π. Ναστου, *Κρυπτογραφία*, σημειώσεις από το ομότιτλο μάθημα στο Πανεπιστήμιο Αιγαίου, (2009).

[NHA09] A. M. Nazlee, F. A. Hussin; N. B. Z. Ali, *Serpent encryption algorithm implementation on Complete Unified Device Architecture (CUDA)*, 2009 IEEE Student Conference on Research and Development (SCOReD) (2009), διαθέσιμο στον: https://ieeexplore.ieee.org/document/5443190.

[Oct] GNU, *GNU Octave*, διαθέσιμο στον: https://www.gnu.org/software/octave/index.

[PB13] C. Pradhan, A. K. Bisoi, *Chaotic Variations of AES Algorithm*, International Journal of Chaos, Control, Modelline and Simulation (IJCCMS), Vol.2, No.2 (2013) arXiv:1307.3057.

[PS01] P. Sarkar, P. Scellenberg, *A Parallel Algorithm for Extending Cryptographic Hash Functions*, In Indocrypt '01, vol. 2247 of LNCS, pages 40–49, Springer-Verlag, (2001).

[PS03] P. Pal, P. Sarkar, *PARSHA-256 – A New Parallelizable Hash Function and a Multithreaded Implementation*, In FSE '03, vol. 2887 of LNCS, Springer-Verlag, σελ. 347–361 (2003).

[RD99] J. Daemen, V. Rijmen, *AES Proposal: Rijndael* (1999) διαθέσιμο στον: https://web.archive.org/web/20070203204845/https://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf.

[RDD] https://spark.apache.org/docs/latest/rdd-programming-guide.html.

[RfC1851] Request for Comments 1851, Internet Security Glossary (1995), διαθέσιμο στον: https://tools.ietf.org/html/rfc1851.

[RfC2828] Request for Comments 2828, Internet Security Glossary (2000), διαθέσιμο στον: https://www.ietf.org/rfc/rfc2828.txt.

[RfC2315] Request for Comments 2315, Internet Security Glossary (1998), διαθέσιμο στον: https://www.ietf.org/rfc/rfc2315.txt.

[Ro18] C. Rossant, *IPython Interactive Computing and Visualization Cookbook*, Second Edition, ISBN: 978-1785888632 (2018).

[RSSN17] M. J. Rostami, A. Shahba, S. Saryazdi H. Nezamabadi-pour, *A novel parallel image encryption with chaotic windows based on logistic map*, Computers & Electrical Engineering vol. 62, σελ. 384–400 (2017).

[Sh45] C. E. Shannon, *A Mathematical Theory of Cryptography*, Bell System Technical Memo MM 45-110-02, (1945).

[SHF18] T. Shah, T. Haq, G. Farooq, *Serpent algorithm: An improvement by* 4×4 *S-box from finite chain ring*, inProc. Int. Conf. Appl. Eng. Math. (ICAEM), σελ. 32–37 (2018).

[SHF20] T. Shah, T. U. Haq, G. Farooq, *Improved SERPENT Algorithm: Design to RGB Image Encryption Implementation*, IEEE Access, vol. 8, σελ.52609–52621 (2020), διαθέσιμο στον: `https://ieeexplore.ieee.org/document/9022986`.

[Skein] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, J. Walker, *The Skein Hash Function Family* (2010) διαθέσιμο στον: `http://www.skein-hash.info/sites/default/files/skein1.3.pdf`.

[SKWWFKS00] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. H. N. Fergusonk, T. Kohno, M. Stay *The Twofish Team's Final Comments on AES Selection* (2000).

[Sp] `https://spark.apache.org/`.

[St98] F. Stajano, *Serpent Reference Implementation*, University of Cambridge, Department of Computer Science and Technology, The Computer Laboratory, διαθέσιμο στον: `https://www.cl.cam.ac.uk/ fms27/serpent/`.

[St14] W. Stallings, *Cryptography and Network Security* (6th ed.), Upper Saddle River, N.J.: Prentic Hall. σελ. 67–68, ISBN 978-0133354690 (2014).

[St17] W. Stallings, *Cryptography and Network Security: Principles and Practice*, Global Edition. Pearson. σελ. 177, ISBN 978-1292158587 (2017).

[Udemy] `https://www.udemy.com/`.

[WSGYJ13] S. Wang, W. Sun, Y. Guo, H. Yang, S. Jiang, *Design and Analysis of Fast Image Encryption Algorithm based on Multiple Chaotic Systems in Real-time Security Car*, International Journal of Security and Its Applications, vol. 7, No. 6, σελ. 229–240 (2013).

[WZL16] X. Y. Wanga, Y. Q. Zhangb, L. T. Liu, *An enhanced sub-image encryption method*, Optics and Lasers in Engineering 86, σελ. 248–254 (2016).