ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Η κρυπτογραφία στην εποχή των κβαντικών υπολογιστών.
Τρέχουσα κατάσταση και μελλοντικές προκλήσεις.

ΓΚΕΡΑΛΝΤΟ ΑΛΙΑΙ
Α.Μ. 711161005

Εισηγητής: ΙΩΑΝΝΑ ΚΑΝΤΖΑΒΕΛΟΥ

ΕΠΙΚΟΥΡΗ ΚΑΘΗΓΗΤΡΙΑ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Η κρυπτογραφία στην εποχή των κβαντικών υπολογιστών. Τρέχουσα κατάσταση και μελλοντικές προκλήσεις.

ΓΚΕΡΑΛΝΤΟ ΑΛΙΑΙ
Α.Μ. 711161005

Εισηγητής:

ΙΩΑΝΝΑ ΚΑΝΤΖΑΒΕΛΟΥ, ΕΠΙΚΟΥΡΗ ΚΑΘΗΓΗΤΡΙΑ

Ioanna Kantzavelou
Digitally signed by Ioanna Kantzavelou
Date: 2024.04.11 12:48:19 +03'00'

Εξεταστική Επιτροπή:

ΒΑΣΙΛΕΙΟΣ ΜΑΜΑΛΗΣ, ΚΑΘΗΓΗΤΗΣ

Vasileios Mamalis
Digitally signed by Vasileios Mamalis
Date: 2024.04.12 11:43:10 +03'00'

ΓΕΩΡΓΙΟΣ ΜΠΑΡΔΗΣ, ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

Georgios Bardis
Digitally signed by Georgios Bardis
Date: 2024.04.22 11:45:11 +03'00'

Ημερομηνία εξέτασης 22/3/2024

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

«Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Ο Δηλών

ΑΛΙΑΙ ΓΚΕΡΑΛΝΤΟ

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες στην επιβλέπουσα καθηγήτρια μου κυρία Ιωάννα Καντζάβελου άλλα και στον καθηγητή κύριο Κωνσταντίνο Λιμνιώτη, τόσο για την πολύτιμη βοήθεια τους που ήταν καθοριστική στην ολοκλήρωση αυτής της διπλωματικής εργασίας, όσο και για την εμπιστοσύνη και την υπομονή που έδειξαν καθ' όλη τη διάρκεια εκπόνησης αυτής, παρ' όλες τις δυσκολίες που υπήρξαν.

ΠΕΡΙΛΗΨΗ

Αντικείμενο της διπλωματικής εργασίας θα αποτελέσει η διερεύνηση των αλλαγών που έχουν επέλθει στους συγχρόνους τομείς της κρυπτογραφίας λόγω νέων ανακαλύψεων στον τομέα της κβαντικής υπολογιστικής. Πιο συγκεκριμένα θα αναλυθεί ο αλγόριθμος του Shor και πώς αυτός απειλεί τους ήδη υπάρχοντες αλγορίθμους ασύμμετρης κρυπτογράφησης. Στην συνέχεια θα γίνει μια εκτενής έρευνα στους νέους μέτα-κβαντικούς αλγορίθμους που έχουν προταθεί από τον οργανισμό NIST ως νέα πρότυπα, καθώς και αυτών που ακόμα εξετάζονται για πιθανή προτυποποίηση.

ABSTRACT

The subject of the thesis will be the investigation of the changes that have occurred in the contemporary fields of cryptography due to new discoveries in the field of quantum computing. More specifically, Shor's algorithm will be analyzed and how it threatens existing asymmetric encryption algorithms. Then there will be an extensive survey of the new meta-quantum algorithms that have been proposed by the NIST organization as new standards, as well as those that are still being considered for possible standardization.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Κρυπτογραφία
ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Κρυπτογράφηση Δημοσίου Κλειδιού, Ασύμμετρη Κρυπτογράφηση, Αλγόριθμος του Shor,Μέτα Κβαντική Αλγόριθμοι κρυπτογράφησης

Περιεχόμενα

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

DH	Diffie–Hellman key exchange
ECC	Elliptic-curve cryptography
RSA	River-Shamir-Adelman cryptography
NIST	National Institute of Standards and Technology
NSA	National Security Agency
VPN	Virtual private network
TLS	Transport Layer Security
SSH	Secure Shell
SFTP	Secure File Transfer Protocol
FTPS	File Transfer Protocol Secure
IP	Internet Protocol address
VoIP	Voice over Internet Protocol
QS	Quadratic Sieve
GNFS	General Number Field Sieve
SNFS	Special Number Field Sieve
DSA	Digital Signature Algorithm
MITM	Man in the Middle
ECDH	Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
PGP	Pretty Good Privacy
TLS	Transport Layer Security
SSL	Secure Sockets Layer
FIPS	Federal Information Processing Standard
DNS	Domain Name System
DSKS	Duplicate Signature Key Selection
CA	Certification Authority
WOTS	Winternitz OTS

IoT	Intern of Thinks
QC	Quantum Computer
QCs	Quantum Computers
QFT	Quantum Fourier Transform
FT	Fourier Transform
QPE	Quantum Phase Estimation
HDL	Hardware Descriptio Language
ETSI	European Telecommunications Standards Institute
ISO	International Organization for Standardization
KIMP	Key Management Interoperability Protocol
SVP	Shortest Vector Problem
LBC	Lattice-based Cryptography
LLL	Lenstra–Lenstra–Lovász algorithm
PQC	Post-quantum Cryptography
OTS	One Time Settlement Scheme
PRG	Pseudorandom number generator
FTS	Few Time Signature
LWE	Learning With Errors
SIS	Short integer solution
LPE	Learning Problem with Errors
KEM	*Key Encapsulation Method*
PKM	Public Key Encryption
LPR	Lyubashevsky Peikert Regev
SIS	Short Integer Solution
GPV	Gentry Peikert Vaikuntanathan
FORS	Forest of Random Subsets
XMSS	eXtended Merkle Signature Scheme

Κεφάλαιο 1 : Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Η κρυπτογραφία («κρυπτός» + «γράφω») αποτελεί ένα επιστημονικό πεδίο που ασχολείται ιδίως με την απόκρυψη του περιεχομένου ενός μηνύματος ή μιας πληροφορίας. Στην σύγχρονη κοινωνία το πλήθος, ο όγκος, η ταχύτητα και η αξία των πληροφοριών που ανταλλάζονται στο διαδίκτυο είναι πλέον ασύλληπτα μεγάλη. Όλες αυτές οι πληροφορίες είναι αναγκαίο να παραμείνουν κρυφές, εκτός από τον χρήστη ή τους χρήστες του δικτύου για τους οποίους ο αρχικός αποστολέας της πληροφορίας επιθυμεί να κάνει διαμερισμό. Η παρακάτω ανάγκη καλύπτεται μέσω διάφορων αλγόριθμων κρυπτογραφίας, οι όποιοι συνδυάζοντας δύσκολα μαθηματικά προβλήματα και εξειδικευμένες τεχνικές κρυπτογραφίας καταφέρνουν να εξασφαλίσουν πως μόνο ο αναμενόμενος αποδέκτης του μηνύματος θα καταφέρει να ανακτήσει την αναμενόμενη πληροφορία που έχει αποσταλεί και κανένας άλλος, επίσης οι αλγόριθμοι αυτοί καλύπτουν μια ακόμα ανάγκη αυτή της επιβεβαίωσης του αποστολέα του μηνύματος μέσω κάποιων ειδικών τεχνικών που βασίζονται σε κάποιους από τους αλγορίθμους αυτούς. Αντικείμενο της παρούσας διπλωματικής εργασίας αποτελεί η μελέτη και η ανάλυση των ήδη υφιστάμενων αλγορίθμων και τεχνικών κρυπτογραφίας που χρησιμοποιούνται στο Διαδίκτυο, καθώς και των νέων αλγόριθμων που προορίζονται να τους αντικαταστήσουν, όπως επίσης και τους λόγους που οδήγησαν στην ανάγκη για δημιουργία νέων ασύμμετρων κρυπτογραφικών αλγορίθμων.

1.2 Μεθοδολογία

Αναλύοντας ήδη υπάρχουσα βιβλιογραφία αλλά και μέσω παραδειγμάτων και πειραμάτων θα απαντηθούν μια σειρά από σημαντικά ερωτήματα. Αρχικά θα απαντηθεί το ερώτημα για ποιο λόγο είναι σημαντική η ασύμμετρη κρυπτογράφηση και ποιοι είναι οι αλγόριθμοι εκείνοι που εφαρμόζονται για να την υλοποιήσουμε. Στην συνέχεια θα πραγματοποιηθεί μια εκτενής ανάλυση στον αλγόριθμο του Shor, ένας αλγόριθμος που «τάραξε» τα θεμέλια της σύγχρονης κρυπτογραφίας. Μέσω παραδειγμάτων για την λειτουργία του αλγόριθμου θα προσφερθεί στον αναγνώστη της εργασίας ένα σύνολο από δεδομένα και πληροφορίες για να κρίνει ο ίδιος αν ο αλγόριθμος αυτός είναι ικανός να αποτελέσει μια πραγματική απειλή ή αν απλά είναι ένας θεωρητικός αλγόριθμος χωρίς κάποια πρακτική αξία. Το ερώτημα αυτό δεν μπορεί να απαντηθεί στην παρακάτω εργασία διότι η υπάρχουσα βιβλιογραφία δεν δίνει κάποια ξεκάθαρη απάντηση. Ωστόσο ο NIST θεώρησε ότι η υπάρχει μια απειλή για τα θεμέλια της ασύμμετρης κρυπτογραφίας, για αυτό και το 2022 μετά από ένα διαγωνισμό που διαρκεί ήδη επί έξι και πλέον χρόνια ανακοίνωσε κάποιους νέους αλγορίθμους ασύμμετρης κρυπτογράφησης που δεν κινδυνεύουν από τον αλγόριθμο του Shor ή κάποιον άλλο γνωστό κβαντικό αλγόριθμο. Ο τελικός στόχος της διπλωματικής αποτελεί μέσω μια εκ βάθους ανάλυσης των αλγορίθμων να δώσει κάποιες απαντήσεις σε μια σειρά ερωτημάτων σχετικά με την λειτουργικότητα και την πρακτική τους αξία σε πραγματικές εφαρμογές κυβερνοασφάλειας.

1.3 Δομή εργασίας

Στο δεύτερο κεφάλαιο της εργασίας θα αναλυθούν τα στοιχεία εκείνα τα οποία καθιστούν σημαντική την ασύμμετρη κρυπτογράφηση. Θα παρουσιαστούν πρωτόκολλα που διαχειρίζονται την διάδοση της πληροφορίας στο Διαδίκτυο και πώς εφαρμόζουν κάποιο αλγόριθμο ασύμμετρης κρυπτογραφίας (υποκεφάλαια 2.2.1, 2.3.5 και 2.5.4) τόσο για την απόκρυψη της πληροφορίας όσο και για την επιβεβαίωση της πηγής που την παρέχει. Στην συνέχεια στο ίδιο κεφάλαιο θα μελετηθεί,το γιατί εδραιωμένοι αλγόριθμοι που χρησιμοποιούνται αυτή την στιγμή δεν θεωρούνται αξιόπιστη και πως το φαινόμενο αυτό κλονίζει την εμπιστοσύνη των ειδικών στην κυβερνοασφάλεια για το μέλλον και την βιωσιμότητα τους. Συγκριμένα στο υποκεφάλαιο 2.3.6 θα γίνει ανάλυση του αλγόριθμου GNFS, για το πόσο ο αλγόριθμος αυτός οδήγησε σε μεγάλη αύξηση του μεγέθους των κλειδιών στους αλγόριθμους DH και RSA για να παραμείνουν ασφάλισης, και γιατί η αύξηση αυτή είχε ως αποτέλεσμα, οι ασύμμετροι αλγόριθμοι που βασίζονται σε ελλειπτικές καμπύλες πχ. ο ECDH (υποκεφάλαιο 2.5) να γίνουν πιο δημοφιλής και να εδραιωθούν ως εναλλακτικοί στους αλγορίθμους που βασίζονταν στην παραγοντοποιήσει πρώτον αριθμών. Στο τρίτο κεφάλαιο θα γίνει ενδελεχείς ανάλυση στο αλγόριθμο του Shor, ένας αλγόριθμος που τάραξε τα θεμέλια της ασύμμετρης κρυπτογραφίας/κρυπτογραφία δημόσιου κλειδιού και οδήγησε σε θεμελιώδης αλλαγές. Ο αλγόριθμος του Shor συγκριμένα όταν θα υπάρξει ένας λειτουργικός QC θα καταφέρει να σπάσει μερικούς από τους σημερινούς εδραιωμένους αλγορίθμους σε κάποιες ώρες συμφώνα με τους ειδικούς (υποκεφάλαιο 3.2) και όπως θα δούμε στο υποκεφάλαιο 3.1.2 πληθώρα ειδικών θεωρούν πως ένας λειτουργικός QC δεν απέχει και πολύ για να γίνει πραγματικότητα. Παρόλα αυτά ακόμα και αν δεν απαιτούνται ώρες για σπάσει έναν αλγόριθμο, ο QC παραμένει μια απειλή, όπως θα δούμε στο υποκεφάλαιο 3.1.2. Διότι μπορεί ο αλγόριθμο του Grover αυτή τη στιγμή να μην αποτελεί πρακτική απειλή για αλγόριθμο κρυπτογραφίας βασισμένο σε ελλειπτικές καμπύλες, παρόλα αυτά όπως και με τον GNFS η γνώση της ύπαρξής του και η αύξηση της ταχύτητας του hardware είναι αρκετή για να δημιουργήσει προβληματισμούς. Για αυτό και ο NIST το 2016 ξεκίνησε έναν ανοικτό διαγωνισμό προς την παγκόσμια κρυπτογραφικοί κοινότητα για νέους αλγορίθμους ικανούς να αντέξουν επιθέσεις από κβαντικούς υπολογιστές αλλά φυσικά και επιθέσεις από κλασικούς. Το 2022 (URL: NIST Announces First Four Quantum-Resistant Cryptographic Algorithms | NIST) NIST ανακοίνωσε τέσσερις νέους αλγόριθμους που πέρασαν την τρίτη φάση του διαγωνισμού και προωθούνται για προτυποποίηση και χρήση το αργότερο έως το 2024. Φυσικά η απάντηση στο ερώτημα ποιος από τους παραπάνω αλγόριθμους είναι καλύτερος και πιο πιθανών να εδραιωθεί, είναι δύσκολο να δοθεί. Ωστόσο στο τέταρτο κεφάλαιο θα αναλυθούν οι αλγόριθμοι αυτή σε βάθος για να μελετηθούν τα πλεονεκτήματα και τα μειονεκτήματα του καθενός (υποκεφάλαιο 4.4 Kyber, 4.5 Dilithium, 4.6 Falcon, 4.7 Sphincs^{+}).

Κεφάλαιο 2 : Τρέχουσα κατάσταση στην ασύμμετρη κρυπτογραφία

2.1 Αλγόριθμοι ασύμμετρης κρυπτογράφησης /κρυπτογράφησης δημόσιου κλειδιού

Η μέθοδος της ασύμμετρης κρυπτογράφησης (δεν χρησιμοποιείται το ίδιο κλειδί για κρυπτογράφηση και αποκρυπτογράφηση ενός μηνύματος) περιλαμβάνει δύο διακριτά κλειδιά κρυπτογράφησης που σχετίζονται μέσου ενός μαθηματικού προβλήματος μεταξύ τους. Η μέθοδος της ασύμμετρης κρυπτογράφησης είναι γνωστή και ως «Κρυπτογραφία Δημόσιου Κλειδιού», διότι τα δύο κλειδιά που χρησιμοποιούνται για διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης είναι γνωστά ως "δημόσιο" και ως "ιδιωτικό" κλειδί. Ένα μήνυμα που έχει κρυπτογραφηθεί με την μέθοδο της ασύμμετρης κρυπτογράφησης μπορεί να αποκρυπτογραφηθεί και να διαβαστεί μόνο από τον αναμενόμενο παραλήπτη του μηνύματος χρησιμοποιώντας το συσχετισμένο ιδιωτικό κλειδί, το οποίο είναι μυστικό και το γνωρίζει μόνο αυτός. Για την δημιουργία του δημόσιου κλειδιού χρησιμοποιούνται διάφοροι κρυπτογραφική αλγόριθμοι οι οποίοι μέσω διάφορων μαθηματικών προβλημάτων ζευγαρώνουν το συσχετισμένο ιδιωτικό κλειδί με το αντίστοιχο δημόσιο , με τέτοιο τρόπο που καθιστά αδύνατη την εύρεση τους μέσω κάποιας πχ επίθεσης ωμής βίας.

Η μέθοδος αυτή μπορεί να χαρακτηριστεί και ως μια συνάρτηση μονόδρομης λειτουργίας (one way function) διότι είναι εύκολο να υπολογίσει κανείς την έξοδο της(δημοσία κλειδιά) αλλά δύσκολο και χρονοβόρο να υπολογίσεις τα αρχικά δεδομένα της εισόδου (ιδιωτικά κλειδιά) ή trapdoor function όπως είναι γνωστές στον κλάδο της θεωρητικής επιστήμης των υπολογιστών. Γενικά αυτό είναι το μοτίβο σε όλες τις μεθόδους/αλγορίθμους ασύμμετρης κρυπτογράφησης που θα εξετάσουμε στην συνέχεια, ωστόσο καθένα έχει μια διαφορετική προσεγγίσει πάνω στο μαθηματικό πρόβλημα που βασίζεται για να δημιουργία του ζευγάρι δημόσιου/ιδιωτικού κλειδιού.

Οι Hellman, Ralph Merkle και Whitfield Diffie ήταν οι πρώτοι ερευνητές που παρουσίασαν επίσημα το 1976 στον Πανεπιστήμιο του Στάνφορντ την ιδέα της κρυπτογραφίας δημόσιου κλειδιού. Για την δημιουργία του ζευγαριού δημόσιου και ιδιωτικού κλειδιού χρησιμοποίησαν το πρόβλημα υποσυνόλου(subset-sum problem) γνωστό ως πρόβλημα με το σακίδιο (knapsack problem). Το πρόβλημα του σακιδίου αποτελεί ένα πλήρες NP πρόβλημα, και μπορεί να περιγραφεί ως ένα κοντέινερ κάποιου σταθερού μεγέθους και κάποιου συνόλου από διακριτά μπλοκ, στο πρόβλημα αυτό κάποιος μπορεί να βρει το σύνολο των μπλοκ που χωράνε ακριβώς στο κοντέινερ χρησιμοποιώντας μια εξαντλητική αναζήτηση ή χρησιμοποιώντας ειδικές γνώσεις οι όποιες σχετίζονται με το πρόβλημα. Ωστόσο η μέθοδος Merkle–Hellman knapsack cryptosyste που δημοσιεύτηκε το 1978 θεωρείτε μια από τις εύκολες μεθόδους κρυπτογράφησης δημόσιου κλειδιού και το 1984 ο Adi Shamir πρότεινε μια επίθεση πολυωνυμικού χρόνου η οποία κατέστησε την χρήση της μέθοδο για κρυπτογράφηση μη βιώσιμη. Επίσης γνωστοί αλγόριθμοι δημοσίου κλειδιού που θα αναλυθούν στην συνέχεια είναι ο Diffie–Hellman key exchange(DH) που παρουσιάστηκε το 1976 από τον Ralph Merkle ο οποίος αποτελεί ένας από τους πρώτους πρακτικούς αλγορίθμους στο πεδίο της κρυπτογραφίας που εφάρμοσαν την ιδέα της ανταλλαγής δημόσιου κλειδιού. Ο ποίο γνωστός και πλέον διαδεδομένος αλγόριθμος ασύμμετρης

κρυπτογράφηση είναι ο RSA που πήρε το όνομα του από τους δημιουργούς του Ron Rivest, Adi Shamir and Leonard Adleman που δημοσίευσαν τον αλγόριθμο το 1976. Στην συνέχεια το 1991 ο NIST παρουσίασε τον DSA (Digital Signature Algorithm) έναν αλγόριθμο κρυπτογράφησης δημοσίου κλειδιού που χρησιμοποιείτε για ηλεκτρονικές υπογραφές. Τέλος ο ποιο σχετικά νέος αλγόριθμος είναι ο Elliptic-curve cryptography (ECC) ο οποίος έχει αρχίσει να γίνετε ευρέως γνωστός από το 2004 αν και ιδέα να αναπτυχθεί κάποιος αλγόριθμος δημοσίου κλειδιού χρησιμοποιώντας τις μαθηματικές ιδιότητες των ελλειπτικών καμπυλών υπάρχει ήδη από το 1985.

Στα πλεονεκτήματα της ασύμμετρης κρυπτογράφησης περιλαμβάνετε η ασφάλεια που παρέχει. Με την κρυπτογράφηση αύτη διασφαλίζει ότι τα δεδομένα παραμένουν προστατευμένα από επιθέσεις man-in-the-middle (MiTM) αν έχουν εφαρμοστεί σωστά οι μέθοδοι μαζί με κάποια επιπρόσθετα πρωτόκολλα ασφάλειας. Για διακομιστές web/email που συνδέονται με εκατοντάδες χιλιάδες πελάτες ανά πάσα στιγμή, η ασύμμετρη κρυπτογράφηση αποτελεί ένα πολύ σημαντικό εργαλείο, καθώς χρειάζεται να διαχειρίζονται και να προστατεύουν ένα μόνο κλειδί. Ωστόσο το βασικό πλεονέκτημα που χαρίζει η κρυπτογράφηση δημόσιου κλειδιού φαίνεται σε μη ασφαλή δίκτυα διότι επιτρέπει την δημιουργία κρυπτογραφημένης σύνδεσης μεταξύ χρηστών του δικτυού χωρίς να χρειάζεται να ανταλλάξουν πρώτα κλειδιά εκτός σύνδεσης. Επίσης εξίσου σημαντικό χαρακτηριστικό της ασύμμετρη κρυπτογράφηση είναι ο έλεγχος ταυτότητας που προσφέρει. Τα δεδομένα που έχουν κρυπτογραφηθεί με ένα δημόσιο κλειδί μπορούν να αποκρυπτογραφηθούν αντίστοιχα χρησιμοποιώντας μόνο το ιδιωτικό κλειδί που σχετίζεται με αυτό. Με αυτόν τον τρόπο, διασφαλίζεται ότι τα δεδομένα αποκρυπτογραφούνται και διαβάζονται μόνο από τον αποδέκτη που είναι εξουσιοδοτημένος να τα λαμβάνει. Δηλαδή επαληθεύεται ότι ο αποστολέας του μηνύματος επικοινωνεί με τον αναμενόμενο δέκτη.

Ένα από τα κυρία μειονεκτήματα της κρυπτογραφίας δημόσιου κλειδιού είναι η ταχύτητα και οι πόροι που απαιτούν οι διάφοροι αλγόριθμοι για τα υλοποιηθούν . Υπάρχουν πολλές μέθοδοι κρυπτογράφησης με μυστικό κλειδί που είναι σημαντικά ταχύτερες από οποιαδήποτε διαθέσιμη σήμερα μέθοδο κρυπτογράφησης με δημόσιο κλειδί. Ωστόσο, η κρυπτογραφία δημόσιου κλειδιού μπορεί να χρησιμοποιηθεί παράλληλα με κρυπτογραφία μυστικού κλειδιού (συμμετρική κρυπτογράφηση) για να αξιοποιηθεί το καλύτερο και από τις δύο μεθόδους. Ένα άλλο σημαντικό μειονέκτημα της κρυπτογράφησης δημόσιου κλειδιού μπορεί να θεωρηθεί η τρωτότητα στην πλαστοπροσωπία. Αν ένας κακόβουλος χρήστης του δικτύου καταφέρει μια επιτυχημένη επίθεση σε κάποια αρχή πιστοποίησης μπορεί να μιμηθεί όποιον χρήστη επιλέξει χρησιμοποιώντας το αντίστοιχο πιστοποιητικό δημόσιου κλειδιού από την παραβιασμένη αρχή .

2.2 Ανταλλαγή κλειδιών με τον Diffie–Hellman

Η μέθοδος ανταλλαγής κλειδιών Diffie-Hellman (αλλιώς γνωστή και ως εκθετική ανταλλαγή κλειδιών) επιτρέπει σε όσους δεν έχουν συναντηθεί ποτέ πριν σε ένα μη ασφαλές κανάλι που μπορεί να παρακολουθείται από κακόβουλους χρήστες, να δημιουργήσουν με ασφάλεια ένα κοινόχρηστο κλειδί. Για την δημιουργία του ζευγαριού ιδιωτικού δημόσιου κλειδιού η μέθοδος βασίζετε στο μαθηματικό πρόβλημα πολλαπλασιαστικής ομάδα ακεραίων modulo n (multiplicative group of integers modulo n).

Ο αλγόριθμος για την μέθοδο περιλαμβάνει τα παρακάτω βήματα για τους υποθετικούς χρήστες Alice και Bob που θέλουν να επικοινωνήσουν μέσω ενός μη ασφαλούς δικτύου:

1. Αρχικά η Alice και ο Bob θα διαλέξουν από κοινού δύο μεγάλους πρώτους αριθμούς, **p** και **g** και έναν αλγόριθμο ανταλλαγής δημοσίου κλειδιού.
2. Η Alice διαλέγει έναν μυστικό αριθμό **a** και υπολογίζει $\mathbf{A} = \boldsymbol{g^a \bmod p}$, αφού υπολογίσει το **A** το στέλνει στον Bob.
3. Στην συνέχεια ο Bob διαλέγει έναν μυστικό αριθμό **b** και υπολογίζει $\mathbf{B} = \boldsymbol{g^b \bmod p}$, αφού υπολογίσει το **B** το στέλνει στην Alice.
4. Η Alice υπολογίζει $\boldsymbol{s = B^a \bmod p}$ και ο Bob υπολογίζει $\boldsymbol{s = A^b \bmod p}$.
5. Η Alice και ο Bob πλέον μπορούν να δημιουργήσουν ένα ασφαλές κανάλι επικοινωνίας με τα κοινά μυστικά κλειδιά.

Η ασφάλεια της ανταλλαγής κλειδιών Diffie-Hellman βασίζεται στο γεγονός ότι είναι υπολογιστικά ανέφικτο για έναν εισβολέα να προσδιορίσει σε εύλογο χρόνο τα κοινόχρηστα μυστικά κλειδιά από τις δημόσιες τιμές των **p, g, A** και **B**.

2.2.1 Πεδία εφαρμογής του DH

Πρωτόκολλα ασφαλούς μεταφοράς αρχείων – Πολυάριθμα πρωτόκολλα ασφαλούς μεταφοράς αρχείων βασίζονται στον Diffie-Hellman, για παράδειγμα τα πρωτόκολλα SFTP και FTPS, χρησιμοποιούν τον DH για την δημιουργία ασφαλούς καναλιού μεταξύ δύο μερών για τη μεταφορά αρχείων. Ο DH επιτρέπει την κρυπτογράφηση και την αποκρυπτογράφηση των μεταφερόμενων αρχείων μέσω ενός κοινού μυστικού κλειδιού που έχει συμφωνηθεί αμφότερα από τον αποστολέα και τον δέκτη των αρχείων.

Ασφαλή πρωτόκολλα επικοινωνίας – Ο DH είναι ευρέος διαδιδόμενος και σε αρκετά πρωτόκολλα ασφαλούς επικοινωνίας, όπως για παράδειγμα ο SSL/TLS και ο SSH. Όπου χρησιμοποιείται για την δημιουργία ασφαλούς καναλιού μεταξύ δύο μερών. Μέσω ενός ένα κοινό μυστικό κλειδί που έχει συμφωνηθεί και από τα δύο μέρη που ανταλλάσουν τα μηνύματα ο DH χρησιμοποιείτε τόσο για την κρυπτογράφηση όσο και για την αποκρυπτογράφηση των μηνυμάτων που μεταφέρονται μέσω του καναλιού.

Εικονικά ιδιωτικά δίκτυα (VPN) – VPN υπηρεσίες συχνά χρησιμοποιούν τον DH για την εδραιώσει ασφαλούς σύνδεσης μεταξύ ενός πελάτη και ενός διακομιστή. Χρησιμοποιώντας ένα κοινό μυστικό κλειδί που έχει συμφωνηθεί και από τις δύο μεριές μέσω του DH οι υπηρεσίες VPN κρυπτογραφούν και αποκρυπτογραφούν τις κίνησης που ανταλλάσσονται στην εγκαθιδρυμένη VPN σύνδεση.

Άλλες εφαρμογές – Πληθώρα εφαρμογές οι οποίες χρησιμοποιούνται στο Διαδίκτυο και για τις όποιες απαιτείται ασφαλής επικοινωνία χρησιμοποιούν τον DH. Εφαρμογές όπως για παράδειγμα email, online banking και VoIP(μεταφορά της φωνής μέσω διαδικτύου). Γενικά η ανταλλαγή κλειδιών μέσω του DH είναι μια ευέλικτη και ευρέως διαδομένη αλλά και υποστηριζόμενη τεχνική για τη δημιουργία ασφαλών καναλιών επικοινωνίας σε ένα μη ασφαλές δίκτυο .

2.2.2 Τρωτά σημεία DH

Επιθέσεις εκθετών – Οι μυστικοί εκθέτες (a και b) που χρησιμοποιούνται στον DH για την ανταλλαγή των κλειδιών πρέπει να επιλέγονται τυχαία, σε αντίθετη περίπτωση ένας εισβολέας μπορεί να ανακτήσει το κοινόχρηστο μυστικό κλειδί, εκμεταλλευόμενος την έλλειψη τυχαιότητας. Το ελάχιστο μέγεθος κλειδιού πρέπει να είναι τουλάχιστον 1024 bits ενώ ο NIST προτείνει κλειδιά μεγέθους 2048 bits.

Επιθέσεις μικρών υποομάδων – Εάν τύχει ο πρώτος αριθμός p που χρησιμοποιείται στην ανταλλαγή κλειδιών να έχει κάποια μικρή υποομάδα, κάποιος υποτιθέμενος εισβολέας θα μπορούσε να χρησιμοποιήσει αυτό το κενό ασφάλειας προς όφελός του για να ανακτήσει το κοινόχρηστο μυστικό κλειδί. Για να αποφευχθεί το παρακάνω κενό, είναι συνετό να επιλέγεται κάποιος μεγάλο πρώτος αριθμός ο οποίος δεν έχει γνωστές μικρές υποομάδες.

Επιθέσεις Man-in-the-Middle –Είναι πιθανό κάποιος υποτιθέμενος εισβολέας να καταφέρει να υποκλέψει και να τροποποιήσει τα μηνύματα που δρομολογούνται μεταξύ της Alice και του Bob κατά την ανταλλαγή κλειδιών. Αν ο υποτιθέμενος εισβολέας ολοκλήρωση με επιτυχία αυτή την επίθεση τότε θα μπορεί να είναι σε θέση να υποδυθεί ή την Alice ή τον Bob και να δημιουργήσει ένα ασφαλές κανάλι με το άλλο μέρος παραπλανώντας το για την πραγματική ταυτότητα του χρήστη που βρίσκεται στην άλλη άκρη της γραμμής. Η επίθεση αυτή μπορεί να αποφευχθεί χρησιμοποιώντας παράλληλα με το μυστικό κλειδί και έλεγχο ταυτότητας που βασίζεται σε πιστοποιητικό ή και μέσω επαλήθευσης της αυθεντικότητας, χρησιμοποιώντας κωδικούς ελέγχου ταυτότητας μηνυμάτων, των μηνυμάτων που αποστέλλονται μεταξύ των χρηστών μιας υποτιθέμενης ασφαλούς σύνδεσης

2.3 RSA

Ο αλγόριθμος RSA αποτελεί ο πυρήνα μιας σουίτας κρυπτογραφικών αλγορίθμων που εφαρμόζονται σε συγκεκριμένους σκοπούς ή συγκριμένες υπηρεσίες οπού απαιτεί ασφάλεια . Ο RSA επιτρέπει μέσω της κρυπτογράφηση δημόσιου κλειδιού την ανταλλαγή ευαίσθητων δεδομένων σε μη ασφαλή δίκτυα , ωστόσο είναι επίσης ευρέως διαδομένος και για την διασφάλιση της ακεραιότητας και της αυθεντικότητας των ηλεκτρονικών επικοινωνιών. Η δυνατότητα του να μπορεί να χρησιμοποιείται ταυτόχρονα τόσο για κρυπτογράφηση όσο και για αυθεντικοποίηση, έχει καταστήσει τον RSA ένα από τους ποιο ευρέως χρησιμοποιούμενους αλγόριθμους ασύμμετρης κρυπτογράφησης. Για παράδειγμα συσκευές τεχνολογίας που συνδέονται στο Διαδίκτυο και κατασκευάζονται από τις εταιρίες Toshiba, ASUS, Fujitsu, LG, Lenovo, HP, Samsung, Acer διαθέτουν ενσωματωμένα RSA-enabled chip. Η κεντρική μαθηματική ιδέα πάνω στην οποία ο RSA βασίζεται αφορά την δημιουργία ενός αριθμού πολλαπλασιάζοντας δύο αρκετά μεγάλους πρώτους αριθμούς μεταξύ τους, ωστόσο έχει ο RSA βασίζει την ασφάλεια του, στην διαδικασία ευρέσεις των παραγόντων αυτού του αριθμού πίσω στους αρχικούς πρώτους αριθμούς η οποία αποτελεί αρκετά δύσκολη υπολογιστική διαδικασία. Το ζεύγος δημοσίου και ιδιωτικού κλειδιού δημιουργούνται μέσο δύο αριθμών, ο ένας εκ των οποίων αποτελεί το γινόμενο των δύο μεγάλων πρώτων αριθμών. Τα δύο κλειδιά χρησιμοποιούν τους ίδιους δύο πρώτους αριθμούς για να υπολογίσουν την τιμή τους. Τα κλειδιά RSA συνήθως τείνουν να έχουν μήκος 1024 ή 2048 bit ή και ακόμα μεγαλύτερα για επιπρόσθετη ασφάλεια καθιστώντας εξαιρετικά δύσκολη την παραγοντοποίησή τους, αν και τα κλειδιά 1024 bit πιστεύεται ότι θα σπάσουν σύντομα. Οι μεγάλοι πρώτοι αριθμοί που χρειάζονται παράγονται χρησιμοποιώντας τον αλγόριθμο Rabin-Miller για παραγωγή τέτοιων αριθμών.

2.3.1 Κρυπτογράφηση με τον RSA

Ο RSA χρειάζονται μια πολλαπλασιαστική ομάδα **G =< Zϕ n,*, X >** για την δημιουργία των κλειδιών. Αυτή η ομάδα παρέχει μόνο πολλαπλασιασμό και διαιρέσεις, που απαιτούνται για τη δημιουργία δημόσιων και ιδιωτικών κλειδιών, αυτή η ομάδα είναι μυστική από το κοινό επειδή ο συντελεστής της **ϕ(n)** πρέπει να παραμείνει κρυμμένος. Το δημόσιο κλειδί αποτελείτε από έναν συντελεστή **n** και έναν δημόσιο εκθέτη **e**, ο οποίος συχνά ορίζεται στο 65537, καθώς είναι ένας πρώτος αριθμός που δεν χρειάζεται να είναι πολύ μεγάλος. Το ιδιωτικό κλειδί περιλαμβάνει έναν συντελεστή **n** και τον ιδιωτικό εκθέτη **d**, ο οποίος για να υπολογιστεί εφαρμόζεται ο εκτεταμένος ευκλείδειος αλγόριθμος για να προκύψει το πολλαπλασιαστικό αντίστροφο ως προς το πλήθος του **n**.

Τα παρακάτω βήματα εφαρμόζονται για να υπολογίσουμε το ζευγάρι ιδιωτικού και δημοσίου κλειδιού με τον RSA:

1. Αρχικά επιλέγονται οι δύο μεγάλοι πρώτοι αριθμοί, **p** και **q**.
2. Αφού υπολογίσουμε τους αριθμούς στην συνέχεια τους πολλαπλασιάζουμε για να υπολογίσουμε την πράξη $\boldsymbol{n = p \times q}$, όπου το **n** ονομάζεται συντελεστής κρυπτογράφησης και αποκρυπτογράφησης.
3. Στην συνέχεια επιλέγουμε έναν αριθμό **e** μικρότερο από **n**, έτσι ώστε ο **n** να είναι σχετικά πρώτος προς $\boldsymbol{(p-1)\times(q-1)}$**.** Που σημαίνει ότι τα **e** και $\boldsymbol{(p-1)\times(q-1)}$ δεν έχουν κοινό παράγοντα εκτός από 1. Επιλέγουμε **e** έτσι ώστε $\boldsymbol{1 < e < \varphi(n)}$, **e** να είναι πρώτος του **φ(n)**, $\boldsymbol{MK\Delta\big(e, d(n)\big) = 1}$
4. Εφόσον επαληθευτεί η ισότητα $\boldsymbol{n = p \times q}$, τότε το δημόσιο κλειδί αποτελείται από το ζευγάρι **<e, n>**. Ένα μήνυμα απλού κειμένου **m** κρυπτογραφείται χρησιμοποιώντας το δημόσιο κλειδί **<e, n>,** μέσο του ακόλουθου τύπου $\boldsymbol{C = m^e \bmod n}$, οπού **C** το τελικό κρυπτογραφημένο μήνυμα.

5. Το ιδιωτικό κλειδί υπολογίζετε , μέσο του ακόλουθου τύπου που χρησιμοποιείται για να υπολογίσουμε το **d** έτσι ώστε να ισχύει κάποια από τις παρακάτω ισότητες :

$$\boldsymbol{d \bmod \big((p-1)\times(q-1)\big) = 1} \text{ ή } \boldsymbol{d \bmod \varphi(n) = 1}$$

6. Ως ιδιωτικό κλειδί ορίζετε το ζευγάρι **<d,n>** . Ένα κρυπτογραφημένο μήνυμα **C** μπορεί να αποκρυπτογραφεί στο αρχικό μήνυμα **m** από την παρακάτω φόρμουλα : $\boldsymbol{m = C^d \bmod n}$.

2.3.2 Τρωτά σημεία της κρυπτογράφηση με τον RSA

Αδύναμη Γεννήτρια Τυχαίων Αριθμών – Υπάρχει περίπτωση κάποιος οργανισμός να επιλέξει μια αδύναμη γεννήτρια τυχαίων αριθμών, τότε οι πρώτοι αριθμοί που δημιουργούνται από την γεννήτρια αυτή είναι πολύ πιο εύκολο να υπολογιστούν, δίνοντας έτσι στους επιτιθέμενους ευκολότερο χρόνο να σπάσουν τον αλγόριθμο.

Παραγωγή αδύναμου κλειδιού – Τα κλειδιά που δημιουργούνται με τον RSA προϋποθέτουν ορισμένες απαιτήσεις σχετικά με τη δημιουργία τους. Στην περίπτωση που οι πρώτοι αριθμοί που έχουν επιλέγει είναι πολύ κοντά ή κάποιος από τους αριθμούς που συνθέτουν το ιδιωτικό κλειδί δεν είναι πολύ μεγάλος, τότε η διαδικασία ευρέσεις του κλειδί γίνεται πολύ πιο γρήγορη και εύκολη. Η ασφάλεια του RSA είναι αλληλένδετη με το μέγεθος του κλειδιού το οποίο πρέπει να είναι δύσκολο να σπάσει, όσο αυξάνεται το μέγεθος ενός κλειδιού αυξάνεται και η ασφάλεια του. Ερευνητές έχουν πετύχει να σπάσουν έναν αλγόριθμο RSA κλειδιού 768 bit αξιοποιώντας την αρχική παραγοντοποίηση ωστόσο χρειάστηκαν 2 χρόνια, χιλιάδες ανθρωποώρες και μια ιδιαιτέρα μεγάλη ποσότητα υπολογιστικής ισχύος, για αυτό και τα μήκη κλειδιών που χρησιμοποιούνται αυτή τη στιγμή στο RSA εξακολουθούν ακόμα να θεωρούνται σχετικά ισχυρά και ασφαλή. Παρόλα αυτά ο οργανισμός NIST προτρέπει το ελάχιστο μήκος κλειδιού να είναι τουλάχιστον 2048 bit, αλλά πολλοί οργανισμοί χρησιμοποιούν κλειδιά μήκους 4096 bit.

Επιθέσεις πλευρικού καναλιού – Οι επιθέσεις πλευρικού καναλιού είναι ένας τύπος επίθεσης που δεν αποκρυπτογραφεί άμεσα τον RSA, αλλά χρησιμοποιεί πληροφορίες από την υλοποίησή του για να δώσει στους εισβολείς υποδείξεις σχετικά με τη διαδικασία κρυπτογράφησης. Οι παραπάνω κατηγορία επιθέσεων συμπεριλαμβάνει τεχνικές όπως η ανάλυση της ποσότητας ισχύος που χρησιμοποιείται για εκτελεστή ο αλγόριθμος, η ανάλυση πρόβλεψης κλάδου, μια τεχνική κατά την οποία χρησιμοποιούνται μετρήσεις χρόνου εκτέλεσης για να μαντέψουν το ιδιωτικό κλειδί. Επίσης γνωστή τεχνική επίθεσης πλευρικού καναλιού αποτελεί η επίθεση χρονισμού, κατά την οποία ένας εισβολέας έχει τη δυνατότητα να μετρήσει τον χρόνο αποκρυπτογράφησης στον υπολογιστή του στόχου του για έναν αριθμό διαφορετικών αποσταλμένων κρυπτογραφημένων μηνυμάτων, οι παρακάτω πληροφορίες επιτρέπουν σε έναν εισβολέα να εξακριβώσει το ιδιωτικό κλειδί του στόχου. Μεγάλο πλήθος των υλοποιήσεις του RSA αποτρέπουν αυτήν την επίθεση εφαρμόζοντας μια διαδικασία γνωστή ως cryptographic blinding, οπού μια εικονική τιμή(one off value) προστίθεται κατά τη διαδικασία κρυπτογράφησης, η οποία αλλοιώνει τη συσχέτιση αποκρυπτογράφησης και απαιτούμενων πόρων.

Επίθεση παραγοντοποίησης – Η ασφάλεια στην κρυπτογράφηση RSA άγεται στην δυσκολία εύρεσης/υπολογισμού των πρώτων αριθμών p και q από το γινόμενο n. Εντούτοις, υπάρχει μια πιθανότητα οι πρώτοι αριθμοί να είναι πολύ κοντά ο ένας στον άλλο ή να μην είναι τυχαίοι και αρκετά μεγάλοι. Στην παρακάτω περίπτωση οι εισβολείς μπορούν να τους συνυπολογίσουν και στη συνέχεια να χρειαστούν ένα μικρό χρονικό διάστημα για να υπολογίσουν το ιδιωτικό κλειδί.

Επίθεση απλού κειμένου – Στην επίθεση απλού κειμένου ο επιτιθέμενος προσπαθεί να αντιστοιχίσει οποιοδήποτε μέρος ενός μηνύματος απλού κειμένου με το αντίστοιχο κρυπτογραφημένο κείμενο. Για παράδειγμα εάν ο επιτιθέμενος έχει στην κατοχή του ορισμένα μέρη του μηνύματος απλού κειμένου αρχικά θα το κρυπτογραφήσει για να λάβει το κρυπτογραφημένο κείμενο. Στη συνέχεια, μπορεί να το χρησιμοποίηση για να αναπαράγει και άλλα μέρη του μηνύματος απλού κειμένου. Αυτή η επίθεση μπορεί να αντιμετωπιστεί συμπεριλαμβάνοντας επιπλέον δεδομένων στο μήνυμα απλού κειμένου πριν το κρυπτογραφηθεί.

Επίσης ακόμα ένα παράδειγμα επίθεσης απλού κειμένου αποτελεί ο έλεγχος των πράξεων μετάθεσης που θα μπορούσαν να είχαν γίνει για τη δημιουργία του κρυπτογραφημένου κειμένου. Εάν λειτουργήσει η επίθεση αυτή ένας κακόβουλος χρήστης θα μπορούσε στη συνέχεια να αντιστρέψει τη διαδικασία για να λάβει το μήνυμα απλού κειμένου από το κρυπτογραφημένο κείμενο.

Επιλεγμένη επίθεση κρυπτογράφησης – Εφόσον ο RSA για την δημιουργία του ιδιωτικού κλειδιού εφαρμόζει τον ευκλείδειο αλγόριθμο, οι εισβολείς θα μπορούσαν εφαρμόζοντας τον εκτεταμένο ευκλείδειο αλγόριθμο να λάβουν από το κρυπτογραφημένο κείμενο το απλό μήνυμα κειμένου.

2.3.3 Ηλεκτρονική υπογραφή και επαλήθευση με τον RSA

Η γενική ιδέα της ηλεκτρονικής υπογραφής και επαλήθευσης μέσω αλγορίθμων ασύμμετρης κρυπτογράφησης θα αναλυθεί εκτενέστερα στην συνέχεια. Στο υποκεφάλαιο αυτό θα παρουσιαστεί πως υλοποιείτε αυτή ιδέα μέσω του RSA.

Το ζεύγος κλειδιών του RSA απαρτίζεται από ένα **δημόσιο κλειδί {n, e}** και ένα **ιδιωτικό κλειδί {n, d}**. Οι αριθμοί **n** και **d** είναι μεγάλοι ακέραιοι (π.χ. 2084 bit), ενώ ο αριθμός **e** είναι μικρός ακέραιος. Όλα τα ζεύγη κλειδιών RSA έχουν την ακόλουθη ιδιότητα, για όλα τα **m** που βρίσκονται στην περιοχή [0...n).

Για να υπογράψουμε ηλεκτρονικά ένα μήνυμα **m** με τον RSA με το ιδιωτικό εκθέτη **d** ακολουθούμαι τα παρακάτω βήματα:

1. Υπολογίζουμε το hash (κατακερματισμός)του μηνύματος **m**: $\boldsymbol{h = hash(m)}$.
2. Κρυπτογραφούμαι το **h** για να υπολογίσουμε την **ηλεκτρονική υπογραφή s** : $\boldsymbol{s = h^d \bmod n}$.

   Το hash του **h** πρέπει να βρίσκεται εντός των ορίων [0...n). Αντίστοιχα η υπογραφή **s** είναι ένας ακέραιος εντός των ορίων [0...n).

Στα σύστημα ασύμμετρου κλειδιού χρησιμοποιούμε το δημόσιο κλειδί για να κρυπτογραφήσουμε ένα μήνυμα και ένα ιδιωτικό κλειδί για να αποκρυπτογραφήσουμε το αντίστοιχο μήνυμα. Ωστόσο, στις ψηφιακές υπογραφές, το ιδιωτικό κλειδί χρησιμοποιείται για την κρυπτογράφηση της υπογραφής και το δημόσιο κλειδί για την αποκρυπτογράφηση της. Εφόσον το ζεύγος δημοσίου/ιδιωτικού κλειδιού λειτουργούν παράλληλα μεταξύ τους, η αποκρυπτογράφηση του ιδιωτικού κλειδιού με το δημόσιο κλειδί σημαίνει ότι το έγγραφο έχει υπογραφθεί από το αναμενόμενο ιδιωτικό κλειδί, επαληθεύοντας έτσι την προέλευση της υπογραφής.

Για να επαληθέψουμε μια ηλεκτρονική υπογραφή **s** με το δημόσιο εκθέτη **e** εφαρμόζουμε τα παρακάτω βήματα:

1. Υπολογίζουμε το hash του μηνύματος **m**: $\boldsymbol{h = hash(m)}$.
2. Αποκρυπτογραφούμε την ηλεκτρονική σφραγίδα **s** : $\boldsymbol{h' = s^e \bmod n}$.
3. Συγκρίνουμε τα **h** και **h'** , αν είναι ιδία σημαίνει πως ο χρήστης που έχει υπογράψει ηλεκτρονικά είναι αυτός που περίμενε ο παραλήπτης.

   Αν η ηλεκτρονική υπογραφή είναι σωστή τότε θα πρέπει να επαληθεύεται η παρακάτω σχέση : $\boldsymbol{h' = s^e \bmod n = \left(h^d\right)^e \bmod n = h}$.

2.3.4 Τρωτά σημεία της ηλεκτρονική υπογραφή και επαλήθευση με τον RSA

Επίθεση επιλεγμένου μηνύματος – Στην επίθεση επιλεγμένου μηνύματος, ο εισβολέας δημιουργεί δύο διαφορετικά μηνύματα, το M1 και M2, στην συνέχεια με κάποιο τρόπο καταφέρνει να πείσει τον γνήσιο χρήστη να υπογράψει και τα δύο μηνύματα χρησιμοποιώντας τον RSA. Έστω τα μήνυμα M1 και M2, ο εισβολέας θα υπολογίσει το νέο μήνυμα M = M1 x M2 ενώ στη συνέχεια για να παραπλανήσει τον παραλήπτη του μηνύματος θα ισχυριστεί ότι ο γνήσιος χρήστης έχει υπογράψει το μήνυμα M.

Επίθεση μόνο με κλειδί – Στην επίθεση αυτή, κάνουμε την υπόθεση ότι ο εισβολέας έχει πρόσβαση στο γνήσιο δημόσιο κλειδί ενός χρήστη και προσπαθεί να υποκλέψει ένα μήνυμα και την ψηφιακή υπογραφή που φέρει το μήνυμα. Στην συνέχεια ο εισβολέας δημιουργεί το νέο μήνυμα MM με τέτοιο τρόπο ώστε η ίδια υπογραφή S να φαίνεται να ισχύει για το MM. Ωστόσο, λόγο της άσχετα υψηλής πολυπλοκότητας η παράπονο επίθεση δεν είναι εύκολη στην υλοποίηση της.

Επίθεση σε γνωστά μηνύματα – Στην επίθεση γνωστού μηνύματος, ο εισβολέας επιχειρεί να εκμεταλλευτεί ένα χαρακτηριστικό του RSA όπου για δύο διαφορετικά μηνύματα που έχουν δύο εντελώς διαφορετικές υπογραφές θα μπορούν να συνδυαστούν με τέτοιο τρόπο που θα επέτρεπε αντίστοιχα και στις ηλεκτρονικές τους υπογραφές να συνδυαστούν. Για παράδειγμα, έστω ότι έχουμε δύο διαφορετικά μηνύματα M1 και M2 και αντίστοιχες ψηφιακές τους υπογραφές S1 και S2. Εφόσον M = (M1 x M2) mod n και S = (S1 x S2) mod n. Ο εισβολέας θα μπορεί να υπολογίσει M = (M1 x M2) mod n και αργότερα S = (S1 x S2) mod n για να πλαστογραφήσει κάποια υπογραφή.

Επιθέσεις σφάλματος (RSA-CRT attack) – Κάποιες φόρες έχουμε διάφορα σφάλματα που συμβαίνουμε κατά την εκτέλεση του RSA , η απλώς συμβαίνουν λόγου τυχαίων σφαλμάτων που μπορούν να συμβαίνουν στο υλικό που εκτελεί τον RSA , έστω ένα τυχαίο bit να μετατοπιστεί σε λάθος θέση την λάθος στιγμή και έχουμε αποτυχία στην δημιουργία της ηλεκτρονικής υπογραφής. Αυτό το φαινόμενο βοηθάει στην κρυπτανάλυση έτσι ώστε εφαρμόζοντας το Κινέζικο Θεώρημα Υπόλοιπου να κάνουμε κάποιες παρατηρήσεις στην λανθάνουσα υπογραφή και σε μια σωστή , με αποτέλεσμα να βρούμε πιο εύκολα τα ιδιωτικά κλειδιά του χρήστη που υπογράφει.

2.3.5 Πεδία εφαρμογής του RSA

Ο RSA είναι πλέον ένας παλιός αλγόριθμος κρυπτογράφησης επομένως χρησιμοποιείται συνήθως μαζί με άλλα σχήματα κρυπτογράφησης. Κύριο παράδειγμα χρήσης RSA αποτελεί η κρυπτογράφηση του ιδιωτικού κλειδιού για χρήση σε αλγόριθμους συμμετρικής κρυπτογράφησης και την ασφαλή κοινή χρήση του κλειδιού, ενώ παράλληλα τα πραγματικά ευαίσθητα δεδομένα κρυπτογραφούνται και προστατεύονται με συμμετρική κρυπτογράφηση. Κάποιες από τις κύριες εφαρμογές του RSA περιλαμβάνουν τη δημιουργία ασφαλούς σύνδεσης σε μη ασφαλή δίκτυα και τη δημιουργία ψηφιακών υπογραφών. Ο RSA δεν χρησιμοποιείται για την κρυπτογράφηση μηνυμάτων ή αρχείων επειδή άλλα συστήματα κρυπτογράφησης είναι πιο ασφαλή, πιο γρήγορα και απαιτούν λιγότερους πόρους, για αυτό και μπορούμε να εντοπίσουμε τον RSA σε εφαρμογές που απαιτείτε να δημιουργηθεί μια ασφαλή σύνδεση μεταξύ ξένων χριστών του δικτύου.

Τέτοιες εφαρμογές αποτελούν οι browser, πάροχοι email, εφαρμογές συνομιλίας μέσω του Διαδικτύου, υπηρεσίες cloud, υπηρεσίες VPN, συστήματα P2P και διάφορα άλλα κανάλια επικοινωνίας μέσω του Διαδικτύου. Οι browser χρησιμοποιούν το RSA για να αποτρέψουν επιθέσεις sniffing ή επιθέσεων man-in-the-middle μέσω της δημιουργίας ασφαλών συνδέσεων στο Διαδίκτυο. Η πληθώρα των συνδέσεων Διαδικτύου εφαρμόζουν το πρωτόκολλο SSL για να εξασφαλίσουν την ασφάλεια της κυκλοφορίας σε αυτό και ο RSA αποτελεί μέρος των χειραψιών SSL/TLS. Το OpenVPN χρησιμοποιεί τον RSA για την ανταλλαγή κλειδιών και την ασφαλή επικοινωνία μεταξύ του πελάτη VPN και του διακομιστή VPN όταν συναπτέ μια σύνδεση VPN. Τις περισσότερες φορές οι υπηρεσίες VPN προστατεύουν τα δεδομένα που έχουν πραγματικοί αξία με διαφορετικές μεθόδους και αλγορίθμους κρυπτογράφησης. Όσον αφορά τις ψηφιακές υπογραφές ο RSA μπορεί να εφαρμοστεί σε ένα ευρύ φάσμα εφαρμογών, από το ηλεκτρονικό ταχυδρομείο έως τις τραπεζικές συναλλαγές και τις ηλεκτρονικές αγορές. Τα email για παράδειγμα υπογράφονται συχνά ψηφιακά χρησιμοποιώντας κρυπτογράφηση PGP που εφαρμόζει τον αλγόριθμο RSA για τη δημιουργία ψηφιακής υπογραφής.

2.3.6 Quadratic Sieve και General Number Field Sieve

Οι εξειδικευμένοι αλγόριθμοι Quadratic Sieve και General Number Field Sieve δημιουργήθηκαν για την αντιμετώπιση του προβλήματος της παραγοντοποίησης μεγάλων πρώτων αριθμών. Οι δύο αυτοί αλγόριθμοι είναι αρκετά ταχύτεροι και απαιτούν λιγότερη υπολογιστική ισχύει από την αφελή προσέγγιση της απλής εικασίας ζευγών γνωστών πρώτων. Οι παραπάνω αλγόριθμοι παραγοντοποίησης γίνονται πιο αποδοτική και αποτελεσματικοί καθώς το μέγεθος των αριθμών που συνυπολογίζονται μεγαλώνει. Το χάσμα μεταξύ της δυσκολίας παραγοντοποίησης και πολλαπλασιασμού μεγάλων αριθμών συρρικνώνεται καθώς το μήκος bit του κλειδιού αυξάνεται. Οι δύο αυτοί αλγόριθμοι σε συνδυασμό με την συνεχόμενη αύξηση των διαθέσιμων υπολογιστικών πόρων για την αποκρυπτογράφηση μεγάλων αριθμών, οδηγεί στην συνεχόμενη αύξηση του μεγέθους των κλειδιών που πρέπει να χρησιμοποιούνται. Ωστόσο η συνέχει και γρήγορη αύξηση των κλειδιών δεν είναι μια βιώσιμη κατάσταση για συσκευές χαμηλής κατανάλωσης που έχουν περιορισμένη υπολογιστική ισχύ και κινητές συσκευές. Η δυσκολία της παραγοντοποίησης μεγάλων πρώτων αριθμών δεν είναι βιώσιμη μακροπρόθεσμα. Το γεγονός αυτό καθιστά τον RSA ως ένα αδύναμο και αβέβαιο σύστημα για το μέλλον της κρυπτογραφίας.

Ο Carl Pomerance παρουσίασε έναν από τους πιο αποτελεσματικούς αλγόριθμους παραγοντοποίησης γενικής χρήσεις των δεκαετιών 1980 και 1990 ο αλγόριθμος αυτός είναι γνωστός ως Quadratic Sieve (QS). Αλλά ακόμα και σήμερα ο αλγόριθμος αυτός εξακολουθεί να αποτελεί μια από της κύριες μεθόδους που εφαρμόζετε για παραγοντοποίηση ακέραιων αριθμών μεταξύ 50 και 110 ψηφίων. Ο QS στον πυρήνα του είναι ουσιαστικά ο αλγόριθμος του Dixon με την έννοια ότι εφαρμόζει εξαρτήσεις, βάσεις παραγόντων και ομαλότητα μεταξύ των διανυσμάτων πάνω από το $\frac{Z}{2Z}$ για τον παραγωγή των τετραγώνων τους. Μέσω ωστόσο μιας μικρής τροποποίησης του πολυώνυμου f(x), ο QS σε αντίθεση με την μέθοδο του Dixon, βρίσκει τις ομαλές τιμές με εξαιρετικά πιο γρήγορο τρόπο.

Η ταχύτερη γνωστή μέθοδος για την παραγοντοποίηση μεγάλου ακέραιου ονομάζεται General Number Field Sieve (GNFS), όπου μεγάλος ακέραιος θεωρούμαι αριθμούς με πάνω από 110 ψηφία. Αυτό καθιστά τον GNFS τον καλύτερο αλγόριθμο και πιο ευρέος εφαρμοσμένο για την αποκωδικοποίηση κλειδιών του RSA. Ο GNFS έχει χρησιμοποιηθεί για να συνυπολογιστεί ένας αριθμός 130 ψηφίων ο οποίος είχε προταθεί από την Αμερικάνικη υπηρεσία RSA, ο αριθμός αυτός αποτέλεσε τον μεγαλύτερο αριθμό κρυπτογραφικής σημασίας που έχει πότε παραγοντοποίηθει. Το λεγόμενο «special» Number Field Sieve (SNFS) είναι μια εξειδικευμένη έκδοση του GNFS, το οποίο είναι ασυμπτωτικά ταχύτερο από το GNFS για παραγοντοποίηση ακεραίων που εκφράζονται στη μορφή $(r \times e) \pm s$ όπου r, e, s $\in Z$ και e > 0. Ο αλγόριθμος ενθυλακώνει αποτελέσματα και ιδέες από ένα τεράστιο πλήθος από διάφορους τομείς των μαθηματικών, της επιστήμης των υπολογιστών, της γραμμική άλγεβρα, της θεωρίας γραφημάτων, των πεπερασμένων πεδίων, της θεωρεία αλγεβρικών αριθμών αλλά ακόμα και πραγματική και σύνθετη ανάλυση όλα τα παιδία αυτά παίζουν ιδιαίτερα ζωτικό ρόλο στο GNFS.

2.4 Ηλεκτρονικές υπογραφές με τον DSA

Ο αλγόριθμος DSA είναι ένα FIPS πρότυπο καθώς επίσης και ένα κρυπτοσύστημα δημόσιου κλειδιού που χρησιμοποιείται για ψηφιακές υπογραφές. Βασίζετε πάνω στις μαθηματικές έννοιες του modular exponentiation και το πρόβλημα του διακριτού λογαρίθμου (discrete logarithm problem) για να δημιουργήσει δύο ψηφιακές υπογραφές. Οι περισσότεροι αλγόριθμοι που χρησιμοποιούνται για την δημιουργία ψηφιακών υπογραφών ακολουθούν την τυπική τεχνική οπού υπογράφουν το hash ενός μηνύματος με την ιδιωτική υπογραφή του αποστολέα. Ωστόσο ο DSA διαφοροποιείται καθώς δημιουργεί δύο υπογραφές ενσωματώνοντας δύο πολύπλοκες και μοναδικές λειτουργίες υπογραφής και επαλήθευσης. Ο DSA ξεκινά ιδιωτικό με τον αποστολέα και είναι δημόσιος στο τέλος του παραλήπτη. Ο αποστολέας βάζει μια υπογραφή στο μήνυμα, αλλά ο καθένας μπορεί να το λάβει.

Ο DSA περιλαμβάνει τις παρακάτω τέσσερις διαδικασίες:

1. Δημιουργία κλειδιού
2. Διανομή κλειδιού
3. Υπογράφει μηνύματος
4. Επιβεβαίωση υπογραφής

1) Δημιουργία κλειδιού

Για την δημιουργία του κλειδιού στον DSA υπάρχουν δύο βήματα: δημιουργία παραμέτρων και per-user key.

1.1) Δημιουργία παραμέτρων

- Στην αρχή κάποιος χρήστης πρέπει να διαλέξει μια κρυπτογραφική συνάρτηση κατακερματισμού **(H)** μαζί με το μήκος εξόδου σε bit **|H|**. Αν το μήκος εξόδου **|H|** είναι μεγαλύτερη τότε επιλέγεται το μήκος μέτρου **N**.
- Έπειτα, επιλέγουμε ένα κλειδί που έχει μήκος **L.** Όπου το **L** χρειάζεται να είναι πολλαπλάσιο του 64 και να έχει μέγεθος μεταξύ 512-bit 1024-bit σύμφωνα με το μήκος του αρχικού DSS. Ο NIST ωστόσο προτείνει για την μέγιστη ασφάλεια τα μήκη κλειδιών 2048-bit ή 3072.
- Σύμφωνα με το με το πρότυπο FIPS 186-4 οι τιμές των **L** και **N** πρέπει να επιλέγονται ανάμεσα των (1024, 60), (2048, 224), (2048, 256) ή (3072, 256). Επίσης, ένας χρήστης θα πρέπει να επιλέξει το μήκος του συντελεστή **N** με τέτοιο τρόπο ώστε το μήκος του συντελεστή **N** να είναι μικρότερο από το μήκος κλειδιού **L** (**N<L**) και μικρότερο από και ίσο με το μήκος εξόδου **|H|** (**N<=|H|**).
- Στην συνέχεια ο χρήστης επιλέγει έναν πρώτο αριθμό **q** μεγέθους N bit και έναν άλλο πρώτο αριθμό **p** μεγέθους **L** bit έτσι ώστε το **p-1** να προκύπτει πολλαπλάσιο του **q**. Μετά διαλέγουμε τον αριθμό **h** ως ένας ακέραιος από τη λίστα (2……..p-2).
- Αφού βρούμε τις τιμές **p** και **q** υπολογίζουμε την τιμή **g** εφαρμόζοντας τον παρακάτω τύπο:

$$g = \frac{h^{p-1}}{q \times mod(p)}$$

1.2) Per-user key

Για να υπολογιστούν οι βασικές παράμετροι για έναν μεμονωμένο χρήστη, πρέπει πρώτα να επιλέγει ένας ακέραιος **x (ιδιωτικό κλειδί)** από τη λίστα (1…….q-1), στη συνέχεια υπολογίζετε το **δημόσιο κλειδί y** από τον τύπο:

$$y = g^x \times mod(p)$$

2) Διανομή κλειδιού

Ο χρήστης που έχει υπογράψει πρέπει να διανέμει το δημόσιο κλειδί **y** στο δίκτυο. Τώρα που το δημόσιο κλειδί **y** υπάρχει στο δίκτυο μπορούμε να υπογράψουμε ένα μήνυμα **m** , ακλουθώντας τα παρακάτω βήματα.

3) Υπογραφή μηνύματος

- Αρχικά περνάμε το αρχικό μήνυμα **M** από μια συνάρτηση κατακερματισμού για να πάρουμε τον κατακερματισμό (**h**).
- Στην συνέχεια περνάμε το **h** ως είσοδο σε μια συνάρτηση υπογραφείς η οποία δίνει ως έξοδο τις μεταβλητές **s** και **r** .
- Χρειάζεται να επιλέξουμε έναν τυχαίο ακέραιο αριθμός **k** οπού **0<k<q** .
- Υπολογίζουμε την τιμή **r** , μέσου του τύπου $\boldsymbol{r = \left(g^k \bmod p\right) \bmod q}$.
- Υπολογίζουμε την τιμή **s** , μέσου του τύπου $\boldsymbol{s = (k^{-1} \times (H(m) + (x \times r)) \bmod q.}$
- Υπογραφή αποτελεί ο συνδυασμός **{r,s}** . Στην συνέχεια αποστέλλουμε στον παραλήπτη το **{m,r,s}** οπού **m** το αρχικό μήνυμα.

4) Επιβεβαίωση υπογραφής

Μπορούμε να επιβεβαιώσουμε ότι ένα μήνυμα **m** έχει υπογραφεί από μια υπογραφεί **{r,s}** ακλουθώντας τα παρακάτω βήματα:

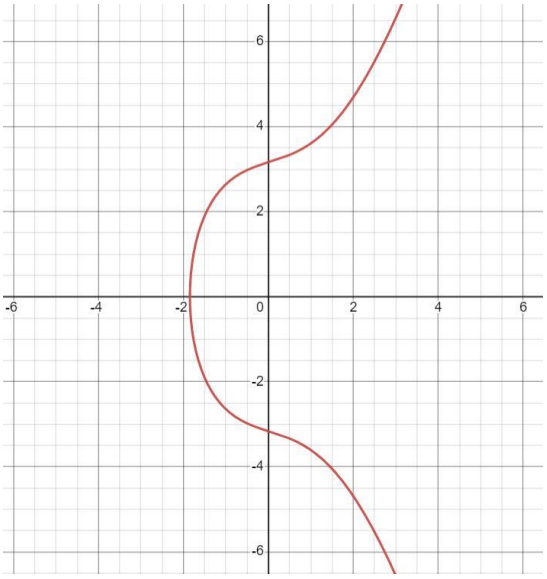
- Ελέγχουμε ότι **0 < r < q** και **0 < s < q**.
- Υπολογίζουμε τις τιμές **w,u_1,u_2** και **υ**, μέσω τον παρακάτω τύπον:
  - $\boldsymbol{w = s^{-1} \bmod q.}$
  - $\boldsymbol{u_1 = (H(m) \times w) \bmod q.}$
  - $\boldsymbol{u_2 = (r \times w) \bmod q.}$
  - $\boldsymbol{u = [(g^{u_1} \times y^{u_2}) \bmod p] \bmod q.}$
  - Αν **u = r** τότε μόνο η υπογραφή είναι έγκυρη.

2.5 Κρυπτογραφία ελλειπτικών καμπυλών(ECC)

Ο ECC είναι ένας κρυπτογραφικός αλγόριθμος δημόσιου κλειδιού που χρησιμοποιείται συνήθως για λόγους ασφαλείας, όπως ο έλεγχος ταυτότητας, η κρυπτογράφηση και οι ψηφιακές υπογραφές. Ο ECC χρησιμοποιεί τη θεωρία της ελλειπτικής καμπύλης για τη δημιουργία κλειδιών σε αντίθεση με τους προηγούμενους αλγόριθμους που βασιζόντουσαν στην παραγοντοποιήσει μεγάλων πρώτων αριθμών. Το γεγονός ότι ο ECC βασίζει την ασφάλεια του στις ελλειπτικές καμπύλες καθιστά την διαδικασία υλοποίηση του χρονοβόρα και πιο περίπλοκη από τον RSA για την υλοποίηση κρυπτοσυστημάτων που έχουν τις ίδιες δυνατότητες, επίσης απαιτούνται τεχνικές γνώσεις υψηλότερου επιπέδου για να εφαρμοστεί με επιτυχία. Στον RSA για λόγους που έχουν αναφερθεί στο Υποκεφάλαιο 2.3.6 απαιτεί μεγάλου μεγέθους κλειδιά για να θωρείτε ασφαλής, σε συγκρίσει με τον ECC που δεν υπάρχουν ακόμα γνωστοί αλγόριθμοι που να επιλύουν το πρόβλημα πιο αποδοτικά με αποτέλεσμα να χρειάζεται μικρότερου μεγέθους κλειδιά για το ίδιο επίπεδο ασφαλείας, για παράδειγμα κλειδί 384-bit ECC παρέχει την ιδία ασφάλεια με κλειδί 7670-bit RSA. Το μικρό μέγεθος των κλειδιών καθιστούν το ECC πιο γρήγορο και πιο ενεργειακά αποδοτικό αλγόριθμο κρυπτογράφησης δημόσιου κλειδιού.

Το σύνολο των σημείων που ικανοποιούν μια συγκεκριμένη μαθηματική εξίσωση ονομάζεται ελλειπτική καμπύλη. Η γενική εξίσωση για μια ελλειπτική καμπύλη έχει την μορφή:

$$y^2 = x^3 + ax + b$$



Σχήμα 2-1: Ελλειπτική καμπύλη, α=2 και b=10

Πολλαπλασιάζοντας ένα σημείο της καμπύλης με κάποιον αριθμό έχει ως αποτέλεσμα την δημιουργία κάποιου αλλού σημείου στην καμπύλη, ωστόσο είναι πολύ δύσκολο να βρεθεί ποιος αριθμός χρησιμοποιήθηκε, ακόμα κι αν είναι γνωστό το αρχικό σημείο και το αποτέλεσμα. Οι εξισώσεις που βασίζονται σε ελλειπτικές καμπύλες ένα διαθέτουν ένα χαρακτηριστικό που της καθιστά αρκετά χρήσιμο για τους σκοπούς της κρυπτογραφίας: είναι σχετικά εύκολο να εκτελεστούν και εξαιρετικά δύσκολο να αντιστραφούν (one way function). Ο διακριτός λογάριθμος της ελλειπτικής καμπύλης (elliptic curve discrete logarithm) αποτελεί το δύσκολο πρόβλημα πάνω στο οποίο στηρίζεται ο ECC για την δημιουργία του ζεύγους δημόσιου, ιδιωτικού κλειδιού.

Η βασική διαδικασία για να δημιουργήσουμε το ζευγάρι δημοσίου ιδιωτικού κλειδιού βασισμένη στον ECC είναι η παρακάτω:

- Διαλέγουμε μία ελλειπτική καμπύλη.
- Διαλέγουμε ένα σημείο **G** πάνω στην καμπύλη το οποίο είναι ίδιο για όλους τους χρήστες.
- Διαλέγουμε έναν αρκετά μεγάλο αριθμό **n** το οποίο είναι και το μυστικό κλειδί.
- Χρησιμοποιώντας πρόσθεση σημείων , προσθέτουμε το **G** στον εαυτό του **n** φορές.
- Η τιμή της συνταγμένης στο άξονα x του αποτελέσματος $\boldsymbol{G} \times \boldsymbol{n}$ αποτελεί το δημόσιο κλειδί.

Η εκτέλεση της πράξει του πολλαπλασιασμού σημείων για n=256 για παράδειγμα χρειάζεται απλά να υπολογιστεί $G + (2 \times G) + (4 \times G) + (8 \times G)$….. το οποίο στην χειρότερη περίπτωση θα είναι ίσο με το μέγιστο μεγέθους του κλειδιού στην περίπτωση του παραδείγματος 256. Η πράξη αυτή για μεγάλα n απαιτεί μια σχετικά μεγάλη υπολογιστική ισχύει ωστόσο όπως αναφέρθηκε ο ECC παρέχει μεγαλύτερη ασφάλεια για μικρότερου μεγέθους κλειδιά σε συγκρίσει με τον RSA για παράδειγμα όποτε η σχετικά μεγάλη υπολογιστική ισχύει θεωρείτε αμελητέα. Τέλος για να υπολογίσει κάποιος επιτιθέμενος το μυστικό κλειδί από την στιγμή που δεν υπάρχει ακόμα κάποιο γνωστό μαθηματικό τρικ για να επίσπευση την αναζήτηση χρειάζεται να ελέγξει όλους τους ενδιάμεσους αριθμούς, και για n=256 δηλαδή για κλειδί μεγέθους 256bits χρειάζεται να ελεγχθούν $2^{256} = 10^{77}$ αριθμοί.

2.5.1 Elliptic-curve Diffie–Hellman (ECDH)

Ο Elliptic Curve Diffie-Hellman (ECDH), αποτελεί μια παραλλαγή του Diffie Hellman σε συνδυάζει με κάποια από τα χαρακτηριστικά του ECC. Ο ECDH επιτρέπει σε δύο άγνωστα μέρη ενός δικτύου που δεν έχουν προηγούμενος γνωριστεί ξανά, να δημιουργήσουν ένα κοινόχρηστο μυστικό κλειδί σε ένα μη ασφαλές κανάλι. Ο ECDH είναι πολύ παρόμοιο με τον κλασικό αλγόριθμο DH, αλλά αντί για modular exponentiations εφαρμόζει πολλαπλασιασμό σημείων ECC, ο ECDH στηρίζεται στην ιδιότητα των σημείων ECC : (a * G) * b = (b * G) * a .Οπού a,b τυχαίοι ακέραιοι αριθμοί και G ένα σημείο πάνω στην καμπύλη.

Μέτα την εκτέλεση του αλγόριθμου ο Bob θα δημιουργήσει ένα δημόσιο κλειδί $\mathbf{Q_B}$ και ένα ιδιωτικό κλειδί $\mathbf{d_B}$ αντίστοιχα η Alice θα δημιουργήσει ένα ζευγάρι δημόσιου , ιδιωτικού κλειδιού $\mathbf{Q_A}$ και $\mathbf{d_A}$. Στην συνέχεια ανταλλάσσουν τα δημόσια κλειδιά τους και ένα κοινό κλειδί $\mathbf{d_A \times d_B \times G}$ όπου **G** είναι ένα σημείο πάνω στην ελλειπτική καμπύλη.

1. Η ελλειπτική καμπύλη που θα επιλέξουμε είναι η Curve 25519 ή X25519 με αλγεβρική μορφή $y^2 = x^3 + 486662x^2 + x$.
2. Ο Bob επιλέγει ένα τυχαίο αριθμό $\mathbf{d_B}$ ως ιδιωτικό κλειδί.
3. Ενώ υπολογίζει το δημόσιο κλειδί $\mathbf{Q_B}$ με τον τύπο:$\boldsymbol{Q_B = d_B \times G}$
4. Η Alice επιλέγει ένα τυχαίο αριθμό $\mathbf{d_A}$ ως ιδιωτικό κλειδί.
5. Ενώ υπολογίζει το δημόσιο κλειδί $\mathbf{Q_A}$ με τον τύπο:$\boldsymbol{Q_A = d_A \times G}$
6. Στη συνέχεια πραγματοποιείται η ανταλλαγεί των δημόσιων κλειδιών μεταξύ της Alice και του Bob. Αφού πραγματοποιηθεί η ανταλλαγή η Alice θα χρησιμοποιήσει το δημόσιο κλειδί του Bob και το δικό της ιδιωτικό της κλειδί για να υπολογίσει:
$$\boldsymbol{SharekeyAlice = d_A \times Q_B}$$
7. Αντίστοιχα ο Bob χρησιμοποιώντας το δημόσιο κλειδί της Alice και δικό του ιδιωτικό κλειδί θα υπολογίσει:
$$\boldsymbol{SharekeyBob = d_B \times Q_A}$$
8. $\boldsymbol{SharekeyAlice = SharekeyBob = (d_B \times d_A \times G)}$

2.5.2 Elliptic Curve Digital Signature Algorithm (ECDSA)

Ο αλγόριθμος ψηφιακής υπογραφής ECDSA είναι ένα κρυπτογραφικά ασφαλές σχήμα ψηφιακής υπογραφής, που βασίζεται στον ECC.Ο ECDSA βασίζεται στα μαθηματικά των κυκλικών ομάδων ελλειπτικών καμπυλών σε πεπερασμένα πεδία και στη δυσκολία του προβλήματος ECDLP (elliptic-curve discrete logarithm problemelliptic-curve discrete logarithm problem). Ο ECDSA στηρίζεται στον πολλαπλασιασμό σημείων EC και υλοποιείται μέσο των παρακάτω βημάτων.

1) Δημιουργία κλειδιών

- Επιλεγούμε ένα σημείο **G** πάνω ECC καμπύλη **.**
- Η μεταβλητή **n** αποτελεί το πλήθος υποομάδων των σημείων EC, που παράγονται μέσο του **G**, το **G** επίσης ορίζει το μήκος που θα έχουν τα ιδιωτικά κλειδιά. Το **n** είναι ένας πρώτος αριθμός.
- Ιδιωτικό κλειδί (ακέραιος) τυχαίος αριθμός από το 0 έως **n-1**: **privkey .**
- Δημόσιο κλειδί (EC σημείο) $\boldsymbol{pubkey} = \boldsymbol{privkey} \times \boldsymbol{G}$.

2) Υπογραφεί μηνύματος

Ο ECDSA κατά την υπογράφει ενός μηνύματος δέχεται ένα μήνυμα **mes** και ένα ιδιωτικό κλειδί **privkey** και συνδυάζοντας αυτά τα δύο παράγει μια υπογράφει που αποτελείτε από δύο ακέραιους **{r,s}** . Τα βήματα που ακολουθεί ο ECDSA για να υπογράψεις είναι τα παρακάτω:

- Υπολογίζει το **hash** του μηνύματος με κάποια συνάρτηση κατακερματισμού π.χ SHA-256: **h**=hash(**mes**).
- Στην συνέχεια παράγει έναν τυχαίο αριθμό $\boldsymbol{k} \in [0, \boldsymbol{n} - 1]$.
- Υπολογίζετε ένα τυχαίο σημείο $\boldsymbol{R} = \boldsymbol{k} \times \boldsymbol{G}$**.** Από το νέο σημείο χρειάζεται μόνο η συντεταγμένη στο άξονα x για υπολογιστή ο πρώτος ακέραιος από το ζευγάρι που αποτελεί την υπογράφει **r=R.x** .
- Υπολογίζετε στην συνέχεια και ο επόμενος ακέραιος από το ζευγάρι που αποτελεί την υπογράφει το **s** , μέσω του τύπου: $\boldsymbol{s} = \boldsymbol{k}^{-1} \times (\boldsymbol{h} + \boldsymbol{r} \times \boldsymbol{privkey}) \ \boldsymbol{mod}\ \boldsymbol{n}$.
- Ο ECDSA επιστρέφει το ζευγάρι **{r,s}** .

3) Επαλήθευση υπογραφής

Ο ECDSA κατά την επαλήθευση μιας υπογραφής δέχεται ένα μήνυμα **mes** , ένα ιδιωτικό κλειδί **privkey** και την παραγόμενη υπογραφή δηλαδή το ζευγάρι ακέραιων **{r,s}** . Με δεδομένο αυτά ακολουθούνται τα παρακάτω βήματα για να γίνει επαλήθευση μιας υπογραφείς :

- Βρίσκουμε το **hash** του μηνύματος χρησιμοποιώντας την ιδία συνάρτηση κατακερματισμού που είχε επιλεχθεί κατά την υπογραφεί **h**=hash(**mes**).
- Υπολογίζετε μια τιμή $\mathbf{s_1}$ μέσο του τύπου $\boldsymbol{s_1} = \boldsymbol{s}^{-1}\ \boldsymbol{mod}\ \boldsymbol{n}$.
- Υπολογίζετε ένα νέο σήμειο **R'.** $\boldsymbol{R'} = (\boldsymbol{h} \times \boldsymbol{s_1}) \times \boldsymbol{G} \times (\boldsymbol{r} \times \boldsymbol{s_1}) \times \boldsymbol{pubkey}$.
- Από το σημείο **R'** χρειάζεται μόνο η συντεταγμένη στο άξονα x **r'=R'.x** .
- Τέλος για να ελέχθη η υπογράφει πραγματοποιείτε ο έλεγχος **if** $\boldsymbol{r} = \boldsymbol{r'}$, αν ισχύει η συνθήκη αυτή το μήνυμα έχει υπογράφει από τον αναμενόμενο αποστολέα.

2.5.3 Τρωτά σημεία του ECDH

Επίθεση στην hash function – Εάν το SHA-1 δεν είναι ανθεκτικό, ένας εισβολέας Ε μπορεί να είναι σε θέση να πλαστογραφήσει τις υπογραφές του Α με τον παρακάτω τρόπο:

- Ένας επιτιθέμενος Ε επιλέγει έναν τυχαίο αριθμό $\boldsymbol{l}$. Στην συνέχεια το **r** υπολογίζεται ως οι συντεταγμένες **x** του $\boldsymbol{pubkey} - (\boldsymbol{l} \times \boldsymbol{G})$ μειωμένου συντελεστή **n**.
- Ο Ε καθορίζει το $\boldsymbol{e} = (\boldsymbol{r} \times \boldsymbol{l})\boldsymbol{mod}\ \boldsymbol{n}$ και θέτει το **s=r**.
- Εάν ο Ε μπορεί να εντοπίσει ένα μήνυμα **m** έτσι ώστε $\boldsymbol{e} = \boldsymbol{SHA} - \boldsymbol{1}(\boldsymbol{m})$, τότε το **(r,s)** αποτελεί μια έγκυρη υπογραφή για το μήνυμα **m.**

Η εύρεση μιας σύγκρουσης για το SHA-1 παραμένει ο πιο γρήγορος γνωστός τρόπος που υπάρχει για πραγματοποίηση επίθεση στο ECDSA χρησιμοποιώντας τις δυνατότητες του SHA-1.

Implementation Attack – Επιθέσεις που κάνουν χρήση αδύναμων ή ψεύτικων τυχαίων αριθμών, ανάλυση διαφορικών σφαλμάτων, ανάλυση διαφορικής ισχύος και επιθέσεις χρονισμού, αποτελούν κάποιες από τις Implementation Attack επιθέσεις που είναι εν δύναμη δυνατό να εφαρμοστούν σε εφαρμογές ECDSA.

DSKS – Σε μια επίθεση DSKS θεωρούμε ότι ο Bob, του οποίου το δημόσιο κλειδί διαθέτη πιστοποιητικό, έχει δρομολογήσει στην Alice ένα υπογεγραμμένο μήνυμα m το οποίο έχει υπογράψει χρησιμοποιώντας το ιδιωτικό κλειδί του s. Ένας επιτυχημένος εισβολέας σε μια επίθεση DSKS για παράδειγμα ο Chris είναι σε θέση να δημιουργήσει ένα δικό του πιστοποιημένο δημόσιο κλειδί το οποίο για το ίδιο μήνυμα m επαληθεύει πως έχει υπογράφει από το ιδιωτικό κλειδί s. Επίσης σε μια DSKS απαιτεί απόδειξη γνώσης του αντίστοιχου ιδιωτικού κλειδιού από τα αρχεία CA για να αποδώσει το πιστοποιητικό για το δημόσιο κλειδί. Μια επίθεση DSKS στο ECDSA μπορεί να πραγματοποιηθεί μόνο εάν καθένας από τους χρήστης επιλέγει το δικό του σημείο παραγωγής στην καμπύλη, ωστόσο, εάν το σημείο παραγωγής είναι κοινή για όλους τους χρήστες, αυτή η επίθεση δεν θα λειτουργήσει.

2.5.4 Πεδία εφαρμογής του ECC

Η κρυπτογραφία που βασίζετε στον ECC θα μπορούσε να πει κάποιος ότι είναι σχετικά καινούρια ωστόσο λόγου τον προτερημάτων της σε συγκρίσει με τον RSA έχει γίνει αρκετά διαδεδομένη σε σύγχρονες και ταχέως αναπτυσσόμενες τεχνολογίες. Πρωτοκόλλα που χρησιμοποιούν τον ECC χρησιμοποιούνται για ηλεκτρονικές τραπεζικές συναλλαγές όταν για παράδειγμα πραγματοποιείτε μια πληρωμή στο διαδίκτυο, τα στοιχεία της κάρτας προστατεύονται με ECC από τον προμηθευτή. Σε εφαρμογές email, το ECC χρησιμοποιείται για την κρυπτογράφηση του email έτσι ώστε κανείς να μην μπορεί να το διαβάσει κατά τη μεταφορά τους, το PGP πρωτόκολλο είναι μία από τις πιο δημοφιλείς λύσεις κρυπτογράφησης email που εξυπηρετούν αυτόν τον σκοπό και βασίζετε στον ECDH. Επίσης ο ECDH εξασφαλίζει την ανωνυμία στον web browser Tor, μερικές ακόμα υπηρεσίες που βασίζονται στον ECDH αποτελούν το iMessage της Apple. Ενώ ακόμα κάποιες από τις υπηρεσίες και τα πρωτοκόλλα που βασίζονται στον ECDA αποτελούν το TLS διάδοχος του SSL που χρησιμοποιείτε ως πρωτόκολλο κρυπτογραφήσεις τις συνδέσεις μεταξύ προγραμμάτων περιήγησης Ιστού και μιας εφαρμογής Ιστού , επίσης η κρυπτογραφημένη σύνδεση ενός ισότοπου HTTPS πραγματοποιείται μέσω υπογεγραμμένων πιστοποιητικών με χρήση του ECDSA, μια ακόμα χρήση του ECDSA συναντάται στην τεχνολογία Bitcoin οπού χρησιμοποιείται για να υπογράφουν συναλλαγές που

πραγματοποιήθηκαν μεταξύ χριστών. Όλοι οι σύγχρονοι browser υποστηρίζουν κάθε πρωτόκολλο ή τεχνολογία του διαδικτύου που χρησιμοποιούν κρυπτογραφία βασιζόμενοι στον ECC. Τέλος η κρυπτογραφία που βασίζετε σε ελλειπτικές καμπύλες έχει γίνει το στάνταρ σε συσκευές IoT διότι η ασφάλεια και η σχετικά μικρή απαίτηση σε υπολογιστικούς πόρους την καθιστούν ιδανική για μικρές ηλεκτρονικές συσκευές που δεν διαθέτουν μεγάλους πόρους.

2.5.5 Γιατί ο ECC δεν έχει αντικαταστήσει ακόμα τον RSA

Αν αναλογιστεί κάνεις τα προτερήματα που έχουν οι αλγόριθμοι που βασίζονται στο ECC απέναντι στον RSA και ειδικά στην σύγχρονη εποχή που το IoT αρχίζει να εδραιώνεται ακόμα περισσότερο θα αναρωτηθεί γιατί η ECC ασύμμετρη κρυπτογράφηση δεν έχει επικράτηση ακόμα. Η απάντηση ωστόσο στο ερώτημα αυτό δεν είναι εύκολη διότι ενώ από τεχνικής απόψεις και σε θεωρητικό επίπεδο οι αλγόριθμοι ασύμμετρης κρυπτογραφήσεις παρά τα προβλήματα που μπορούν να αντιμετωπίζουν είναι αξιόπιστη , κατά την εφαρμογή τους στον αληθινό κόσμο προέκυψαν κάποια θέματα που καθυστέρησαν την εδραίωση τους ως κυρίαρχα στάνταρ και τα οποία έχουν πλήξει τον κύρος τους, στους κύκλους των κρυπταναλυτών. Το πρώτο πρόβλημα αφόρα τα πρωτοκόλλα τα οποία πρότεινε ο NIST για αλγόριθμους που βασίζονται σε ECC. Μετά από ανάλυσης στην γεννήτρια Dual Elliptic Curve Deterministic Random Bit Generator (Dual_EC_DRBG) υπήρχαν υποψίες πως η γεννήτρια αυτή έχει δημιουργηθεί εσκεμμένα ώστε να διαθέτει ένα μυστικό τρωτό σημείο (backdoor), που έχει ως επακόλουθο η ακολουθία των αριθμών που επιστρέφονται να μπορούν να προβλεφθούν πλήρως από κάποιον με τον σωστό μυστικό αριθμό, λόγου αυτού του κενού ασφάλειας η εταιρία RSA (RSA Cybersecurity and Digital Risk Management Solutions) η οποία κατασκευάζει προϊόντα κρυπτογραφίας από 1982 αναγκάστηκε να αποσύρει αρκετά από τα προϊόντας από την αγορά. Δεν έχει αποδειχθεί αν αυτό το κενό ασφάλειας ήταν εσκεμμένο ωστόσο και μόνο ή οι υποψίες έχουν στρέψει αρκετούς κρυπτοαναλύτες στην δημιουργία πρωτοκόλλων ανεξάρτητων από αυτούς που προτείνει ο NIST. Ακόμα ένας λόγος που καθυστέρησε την διάδοση των αλγορίθμων κρυπτογραφίας βασιζομένους στις ECC , αφόρα τα διπλώματα ευρεσιτεχνίας που κάλυπτα τις τεχνολογίες και τα πρωτόκολλα που εφάρμοζαν τους αλγορίθμους αυτούς , προγραμματιστές και μηχανικοί ασφάλειας που ήθελαν να εφαρμόσουν τις νέες αυτές τεχνολογίες δεν μπορούσαν ακόμα διότι φοβόντουσαν κάποια πιθανή αγωγή. Όπως έγινε το 2007 οπού η Certicom έχανε μήνυση στην Sony για παραβίαση διπλώματος ευρεσιτεχνίας που αφορούσε τεχνολογίες που έκαναν χρήση ελλειπτικών καμπυλών. Ωστόσο πολλά από αυτά τα προβλήματα έχουν λυθεί διότι νέα ποιο αξιόπιστα πρωτοκολλά έχουν δημιουργηθεί και πολλές από τις νέες τεχνολογίες δεν καλύπτονται από διπλώματα ευρεσιτεχνίας. Μπορεί ο ECC να μην αντικαταστήσει πλήρης τον RSA , ωστόσο σίγουρα θα αποτελέσει το μεταβατικό στάδιο σε μια νέα γένια αλγορίθμων ασύμμετρης κρυπτογράφησης/δημοσίου κλειδιού που ονομάζονται μετά κβαντικοί αλγόριθμοι κρυπτογράφησης και οι οποίοι θα αναλυθούν εκτενέστερα στο Κεφάλαιο 4ο.

Κεφάλαιο 3 : Η επίδραση των κβαντικών υπολογιστών στην κρυπτογραφία

3.1 Κβαντικός Υπολογιστής

Σύμφωνα με την κβαντική μηχανική ο φυσικός κόσμος δεν είναι ντετερμινιστικός, αλλά πιθανολογικός με εγγενή αβεβαιότητα. Ένας από τους πρωτοπόρους του τομέα ο φυσικός Richard Feynman το 1982 ανέλυσε τις δυνατότητες που είχαν τα κβαντομηχανικά φαινόμενα να χρησιμοποιηθούν για την προσομοίωση ενός κβαντικού συστήματος πιο αποτελεσματικά από μια αφελή προσομοίωση σε έναν κλασικό υπολογιστή. Στην συνέχεια το 1993 οι ερευνητές Bernstein και Vazirani δημιούργησαν έναν νέο κβαντικό αλγόριθμο οποίος πρόσφερε μια εκθετική επιτάχυνση σε σχέση με οποιοδήποτε άλλο κλασικό αλγόριθμο την για εκτέλεση της αναδρομικής δειγματοληψίας Fourier. Ο αλγόριθμος αυτός είναι σημαντικός διότι απέδειξε ότι ένας κβαντικός υπολογιστής (QC quantum computer) παραβιάζει την εκτεταμένη θέση Church-Turing , μια θέση που αποτελεί το θεμέλιο της επιστήμης των υπολογιστών, ακόμα και σήμερα η κβαντική υπολογιστές αποτελούν το μοναδικό γνωστό μοντέλο υπολογισμού που παραβιάζουν αυτή την θέση τουλάχιστον ως προς τον χρόνο επίλυσης κάποιον προβλημάτων διότι προβλήματα που είναι αποδειγμένα μη επιλύσαμε από έναν κλασικό υπολογιστή είναι επίσης μη επιλύσαμε και από έναν κβαντικό. Ωστόσο η εξάπλωση της έννοιας της κβαντικής υπολογιστικής πέρα από τον ακαδημαϊκό χώρο επήλθε το 1994, όταν ο Peter Shor απέδειξε ότι χρησιμοποιώντας ένα θεωρητικό λειτουργικό κβαντικό υπολογιστή πολλά προβλήματα που απαιτούσα έναν ασύλληπτα μεγάλο χρονικό διάστημα για να υπολογιστούν, μπορούν να υπολογιστούν αστρονομικά πιο γρήγορα. Οι νέοι κβαντικοί αλγόριθμοι του Shor υπολόγιζαν την παραγοντοποιήσει μεγάλων ακέραιων και την γρήγορη επίλυση διακριτών λογάριθμων. Ακόμα και σε θεωρητικό επίπεδο αυτοί οι νέοι αλγόριθμοι αποτελούσαν και αποτελούν ένα μεγάλο πρόβλημα στους καθιερωμένους αλγορίθμους ασύμμετρης κρυπτογράφησης που παρουσιαστήκαν στο Κεφάλαιο 2ο. Η απειλή που παρουσίαζαν αυτές οι νέες ανακαλύψεις οδήγησαν πολλά κράτη, οργανώσιμους αλλά και ιδιωτικές εταιρίες στην συστηματική και συνέχει επένδυση για νέες έρευνες στο τομέα αυτό. Ωστόσο παρόλο την πυροδότηση του πρώτου κύματος ενθουσιασμού για την έρευνα πάνω στους κβαντικούς υπολογιστές που επέφεραν οι ανακαλύψεις του Shor πλέον με την προώθηση για χρήση αλγορίθμων κρυπτογράφησης ανθεκτικούς απέναντι στους κβαντικούς υπολογιστές οι κινητήρια δύναμη για έρευνα πάνω στην κβαντική υπολογιστική δεν βασίζετε τόσο πολύ μόνο στην χρήση τους για κρυπτανάλυση.

3.1.1 Βασικές αρχές λειτουργίας ενός QC και προκλήσεις

Σε έναν κλασικό υπολογιστή για να αναπαραστήσουμε τις τιμές με τις οποίες λειτουργεί χρησιμοποιούμε τα bits, σε ένα κβαντικό υπολογιστή χρησιμοποιούμε τα κβαντικά bit ή qubits. Ένα bit μπορεί να έχει την τιμή 0 ή την τιμή 1, ωστόσο ένα qubit έχει την δυνατότητα και μπορεί να αντιπροσωπεύει τις τιμές 0 ή 1, ή ένα γραμμικό συνδυασμό και των δύο, η ιδιότητα αυτή ονομάζεται κβαντική υπέρθεση αυτό συνεπάγεται ότι οι κβαντικοί υπολογιστές μπορούν να πραγματοποιήσουν πολλαπλούς υπολογισμούς με πολλαπλές εισόδους ταυτόχρονα. Ενώ σε ένα κλασικού υπολογιστή η καταστεί που μπορεί να βρίσκεται καθορίζεται από τις δυαδικές τιμές μιας συλλογής bit, σε οποιαδήποτε χρονική στιγμή η κατάσταση ενός κβαντικού υπολογιστή με τον ίδιο αριθμό κβαντικών bit μπορεί να καλύπτει όλες τις πιθανές καταστάσεις ταυτόχρονα του αντίστοιχου κλασικού υπολογιστή με αποτέλεσμα να λειτουργεί σε έναν εκθετικά μεγαλύτερο

χώρο προβλημάτων διότι η κβαντική υπέρθεση προσφέρει την δυνατότητα σε μια ομάδα qubits να εξερευνούν διαφορετικά μονοπάτια μέσω υπολογισμών και όταν προγραμματίζονται σωστά, οι διαδρομές που οδηγούν σε λανθασμένες απαντήσεις αγνοούνται ενώ οι σωστές απαντήσεις παραμένουν επισημασμένες. Ωστόσο, η χρήση των δυνατοτήτων του κβαντικού υπολογιστή απαιτεί όλα τα qubits να είναι εγγενώς διασυνδεδεμένα (Quantum entanglement), πολύ καλά απομονωμένα από το εξωτερικό περιβάλλον και να ελέγχονται με μεγάλη ακρίβεια. Σε αυτήν την πρόταση άγεται η μεγαλύτερη πρόκληση στην δημιουργία ενός λειτουργικού και πρακτικού κβαντικού υπολογιστή ο <<θόρυβος>>.

Ένα qubit μπορεί να είναι οποιοσδήποτε συνδυασμός ενός και μηδέν ωστόσο για να ελεγχτεί το φαινόμενο αυτό τα qubits απαιτούνται να είναι απόλυτα απονεμένα από το φυσικό χώρο που βρίσκονται. Ωστόσο η απολυτή απομόνωση των qubits είναι ένα αρκετά δύσκολο μηχανικό επίτευγμα με αποτέλεσμα σε επίπεδο hardware οι QCs να παράγουν σφάλματα στην έξοδο. Ωστόσο ακόμα και σε ένα θεωρητικά τέλεια απομονωμένο hardware λόγου διάφορων κβαντικών φαινόμενων τα οποία δεν είναι ακόμα πλήρης αντιληπτά οι κβαντικές πύλες δεν μπορούν να απορρίψουν μικρά σφάλματα (θορύβους) που παράγονται στα φυσικά κυκλώματα. Με αποτέλεσμα τα μικρά σφάλματα που μπορούν να εμφανιστούν στην εκτέλεση κβαντικών πράξεων ή τυχόν τυχαία σήματα που εμφανίζονται στο φυσικό σύστημα να οδηγήσουν τελικά σε εσφαλμένες εξόδους, επίσης όσο αυξάνεται ο αριθμός των qubit αυξάνεται και το επίπεδο του θορύβου. Για αυτό ένας από τους πιο σημαντικούς παραμέτρους σχεδιασμού ενός QC αποτελεί το ποσοστό σφάλματος.

Η ισχύς των αλγορίθμων που τρέχουν σε έναν QC προέρχεται εν τέλει από την εκθετική πολυπλοκότητα των κβαντικών συστημάτων. Η κατάσταση ενός συστήματος που περιλαμβάνει n qubits περιγράφεται και μπορεί να κωδικοποιήσει $N = 2^n$ μιγαδικούς συντελεστές, επιπλέον η εφαρμογή κάθε στοιχειώδους πύλης σε δύο qubit για παράδειγμα ενημερώνει τους 2^n μιγαδικούς αριθμούς που περιγράφουν την κατάσταση, με αποτέλεσμα να φαίνεται ότι ο αλγόριθμος εκτελεί 2^n υπολογισμούς σε ένα μόνο βήμα. Ωστόσο, όταν στο τέλος του υπολογισμού, όταν μετρούνται τα n qubits, το αποτέλεσμα είναι μόνο n κλασικά bit , διότι η μέτρηση της κατάστασης ενός κβαντικού υπολογιστή έχει ως αποτέλεσμα να «καταρρεύσει» από την μεγάλη κβαντική κατάσταση σε ένα μόνο κλασικό αποτέλεσμα. Πρακτικά λόγου του φαινομένου αυτού μπορούμε να εξαγάγουμε μόνο την ίδια ποσότητα δεδομένων από έναν κβαντικό υπολογιστή που θα μπορούσαμε από έναν κλασικό υπολογιστή που έχει το ίδιο μέγεθος. Η δυσκολία σχεδιασμού πρακτικών κβαντικών αλγορίθμων συναντάται στο ιδανικό συνδυασμό των δύο αυτών φαινομένων. Δηλαδή ένας κβαντικός αλγόριθμος θα πρέπει να εκτελεί εργασίες των οποίων οι λειτουργικές λύσεις χρησιμοποιούν τον παραλληλισμό που προσφέρει ένας QC και παράλληλα θα πρέπει να παράγει μια τελική κβαντική κατάσταση που έχει μεγάλη πιθανότητα να παρέχει πολύτιμες πληροφορίες κατά τη μέτρηση της. Η πιο αποδοτικοί αλγόριθμοι για να το επιτύχουν αυτό εκμεταλλεύονται το κβαντικό φαινόμενο <<quantum interference>> .

Αν συνυπολογίσει κάποιος τα μοναδικά χαρακτηριστικά που παρουσιάζουν οι κβαντικοί υπολογιστές αλλά και τις προκλήσεις που πρέπει να αντιμετωπιστούν για να φτιαχτεί ένας πρακτικά λειτουργικός. Είναι απίθανο να αντικαταστήσουν τους κλασσικούς υπολογιστές, διότι για την ορθή και ομαλή λειτουργία τους απαιτητέ ένας σημαντικός αριθμός κλασικών υπολογιστών για να ελέγχουν τις λειτουργίες και να πραγματοποιούν υπολογισμούς που απαιτούνται για την εφαρμογή της κβαντικής διόρθωσης σφαλμάτων. Πρακτικά οι QCs σχεδιάζονται ως συσκευές ειδικού σκοπού που λειτουργούν συμπληρωματικά με τους κλασικούς υπολογιστές. Κανένας δεν μπορεί να προβλέψει με ακρίβεια τον χρονικό ορίζοντα που χρειάζεται για την ανάπτυξη ενός πρακτικού QC , ωστόσο βάση των διαθέσιμων πληροφοριών σχετικά με τον τομέα των QCs κανένας επιστήμονας ή ερευνητής που ασχολείται με τον τομέα αυτό, δεν

βλέπει κάποιο θεμελιώδες πρόβλημα που μπορεί να οδήγηση στην αδυναμία δημιουργίας ενός μεγάλου μεγέθους κβαντικού υπολογιστή ο οποίος παρουσιάζει ανεκτικότητα στα σφάλματα. Ωστόσο, υπάρχει ένας σημαντικός αριθμός από τεχνικές προκλήσεις στο δρόμο για την οικοδόμηση ενός τέτοιου συστήματος και για την ανάπτυξή του με πρακτικό πλεονέκτημα για μια χρήσιμη εργασία.

3.1.2 QCs και κρυπτανάλυση

Οι QCs έχουν μεγάλο αντίκτυπο κυρίως στους αλγόριθμους ασύμμετρης κρυπτογραφίας οι οποίοι φαίνεται να είναι και οι πιο ευάλωτοι σε γνωστούς κβαντικούς αλγόριθμους, κύριος τον αλγόριθμο του Shor. Μέσω του αλγόριθμου του Shor παρέχονται ταχύτεροι μέθοδοι οι οποίοι μπορούν να λύσουν το πρόβλημα διακριτής καταγραφής και να υλοποιήσουν την παραγοντοποιήσει μεγάλων ακεραίων σε εκθετικό χρόνο. Ένας πλήρης λειτουργικός κβαντικός υπολογιστής με περίπου 2300 qubits θα μπορούσε να σπάσει τον RSA αλγόριθμο με κλειδί 1024 bits , σε λιγότερο από μια ημέρα. Η σοβαρή αυτή απειλή ακόμα και σε θεωρητικό επίπεδο οδήγησε το οργανισμό NIST να ξεκινήσει το 2016 μια διαδικασία για την επιλογή και την τυποποίηση νέων αλγορίθμων ασύμμετρης κρυπτογράφησης η οποίοι είναι θωρακισμένοι από τους QCs , η διαδικασία αυτή ολοκληρώθηκε το Μάρτιο του 2022 και οι νέοι μετά-κβαντικοί αλγόριθμοι που έχουν προταθεί θα αναλυθούν στο 4ο Κεφάλαιο .

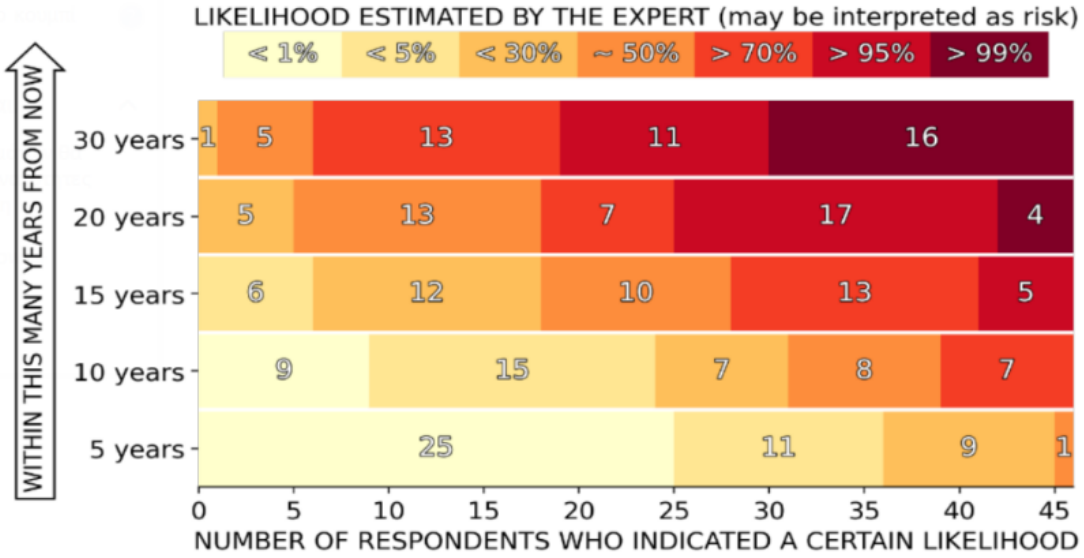
Cryptosystem	Category	Key Size	Security Parameter	Quantum Algorithm Expected to Defeat Cryptosystem	# Logical Qubits Required	# Physical Qubits Required[a]	Time Required to Break System[b]	Quantum-Resilient Replacement Strategies
AES-GCM[c]	Symmetric encryption	128 192 256	128 192 256	Grover's algorithm	2,953 4,449 6,681	4.61×10^6 1.68×10^7 3.36×10^7	2.61×10^{12} years 1.97×10^{22} years 2.29×10^{32} years	
RSA[d]	Asymmetric encryption	1024 2048 4096	80 112 128	Shor's algorithm	2,050 4,098 8,194	8.05×10^6 8.56×10^6 1.12×10^7	3.58 hours 28.63 hours 229 hours	Move to NIST-selected PQC algorithm when available
ECC Discrete-log problem[e-g]	Asymmetric encryption	256 384 521	128 192 256	Shor's algorithm	2,330 3,484 4,719	8.56×10^6 9.05×10^6 1.13×10^6	10.5 hours 37.67 hours 55 hours	Move to NIST-selected PQC algorithm when available
SHA256[h]	Bitcoin mining	N/A	72	Grover's Algorithm	2,403	2.23×10^6	1.8×10^4 years	
PBKDF2 with 10,000 iterations[i]	Password hashing	N/A	66	Grover's algorithm	2,403	2.23×10^6	2.3×10^7 years	Move away from password-based authentication

Εικόνα 3-1: Αλγόριθμοι ασύμμετρης κρυπτογράφησης και κβαντικοί αλγόριθμοι που τους απειλούν

Πηγή:
National Academies of Sciences, Engineering, and Medicine. 2019. Quantum Computing: Progress and Prospects. Washington, DC: The National Academies Press. https://doi.org/10.17226/25196, ISBN 978-0-309-47969-1 | DOI 10.17226/25196.
Σελίδα 98

Όπως παρατηρούμε και στην Εικόνα 3-1 πέρα από τον αλγόριθμο του Shor , υπάρχει και ο κβαντικός αλγόριθμος του Grover για κρυπτανάλυση αλγορίθμων ασύμμετρης κρυπτογράφησης. Ωστόσο παρόλο την θεωρητική μαθηματική υπεροχή που προσφέρει , πρακτικά όπως παρατηρούμε στην στήλη με τίτλο << Time Required to Break System>> είναι σχετικά αδύναμος

διότι σε υπολογιστές με την ίδια ισχύει σε συγκρίσει με τον αλγόριθμο του Shor για να χρησιμοποιηθεί για την κρυπτανάλυση των αλγορίθμων που θεωρητικά απειλεί απαιτούνται χιλιάδες χρόνια ενώ του Shor απαιτούνται κάποιες ώρες. Ωστόσο κανένας δεν μπορεί να προβλέψει με ακρίβεια πότε θα υλοποιηθεί ένας πρακτικός κβαντικός υπολογιστής ο οποίος θα μπορεί να τρέχει τόσο γρήγορα ένα κβαντικό αλγόριθμο για κρυπτανάλυση.



Εικόνα 3-2: Πίνακας που παρουσιάζει τις εκτιμήσεις των ειδικών για το πότε ένας QC θα μπορεί να σπάσει τον RSA-2048

Πηγή: Global Risk Institute, January 2022, 2021 Quantum Threat Timeline Report,
Authors: Dr. Michele Mosca , Dr. Marco Piani
https://globalriskinstitute.org/publication/2021-quantum-threat-timeline-report-global-risk-institute-global-risk-institute/
Σελίδα 24

Σύμφωνα με μια ερεύνα που πραγματοποιήθηκε από τον οργανισμό Global Risk Institute και κάποια από τα ευρήματα που προέκυψαν παρουσιάζονται στην Εικόνα 3-3. Ένα μεγάλο ποσοστό από το συνολικό πλήθος των ειδικών πάνω στο τομέα των κβαντικών υπολογιστών που ερωτήθηκαν στα πλαίσια τις έρευνας(27 στους 46, 56%) , θεωρούν με μεγάλη πιθανότητα (95% έως 99%) πως σε 30 χρόνια από το 2021 (έτος που πραγματοποιήθηκε η έρευνα) θα υπάρξει κβαντικός υπολογιστής που θα μπορεί να σπάσει τον RSA-2048 σε εύλογο χρονικό διάστημα. Ακόμα ωστόσο και αν χρειαστούν λίγο παραπάνω χρόνια η μεταβίβαση από ένα πρωτόκολλο κρυπτογράφησης/ασφάλειας σε ένα άλλο, είναι μια εξίσου χρονοβόρα διαδικασία. Για παράδειγμα το 2004 ανακαλύφθηκε ότι η συνάρτηση κατακερματισμού sha-1 δεν ήταν ασφαλής στην κατηγορία επιθέσεων << "collision" attacks >> . Σύμφωνα με τον NIST (Πηγή: NIST Retires SHA-1 Cryptographic Algorithm | NIST) ο sha-1 θα σταματήσει να χρησιμοποιείτε από όλα τα πρωτοκόλλα στης 31 Δεκεμβρίου του 2030 , δηλαδή μετά από 26 από το έτος που ανακαλύφθηκε ότι ήταν τρωτό. Φυσικά είδη το μεγαλύτερο μέρος του Διαδικτύου χρησιμοποιεί άλλες συναρτήσεις κατακερματισμού όπως για παράδειγμα το sha-256. Ρεαλιστικά ο NIST πιστεύει ότι η μεταβίβαση σε αλγορίθμους ασύμμετρης κρυπτογράφησης θωρακισμένους απέναντι σε QCs θα χρειαστεί 20 χρόνια. Ωστόσο συμβουλεύει οι διαδικασίες για τις απαραίτητες αλλαγές να ξεκινήσουν όσο πιο γρήγορα γίνεται.

3.2 Ο αλγόριθμος του Shor

Ο αλγόριθμος του Shor αναπτύχθηκε το 1994 από το μαθηματικό Peter Williston Shor , είναι ένας κβαντικός αλγόριθμος που επιτρέπει την εύρεση των πρώτων παραγόντων ενός ακέραιου αριθμού. Ο GNFS (Υποκεφάλαιο 2.3.6) αποτελεί τον καλύτερο αλγόριθμος σε κλασικό υπολογιστή για την εύρεση των πρώτον διαιρετέων ενός αριθμού N με χρονική πολυπλοκότητα $\exp(O\left(n^{\frac{1}{3}}\right))$, ενώ ο αλγόριθμος του Shor για το ίδιο πρόβλημα είχε αρχικά χρονική πολυπλοκότητα $O(n^3)$ και στην συνέχεια με βελτιώσεις από μαθηματικό και κρυπτογράφο Don Coppersmith στον αλγόριθμο η πολυπλοκότητα βελτιώθηκε σε $O(n^2 log[n])$. Πρακτικά ο αλγόριθμος του Shor είναι δύο αλγόριθμοι πολυωνυμικού χρόνου που χρησιμοποιούνται για την παραγοντοποιήσει και τον υπολογισμό διακριτών λογάριθμων, στον πυρήνα τους οι αλγόριθμοι αυτοί αποτελούν μια βελτιωμένη εφαρμογή του QFT (Quantum Fourier Transform).

Αρχικά ο Shor ανάγαγε το πρόβλημα εύρεσης των παραγόντων ενός αριθμού σε ένα νέο πρόβλημα εύρεσης επαναλαμβανόμενου μοτίβου, δηλαδή μια εργασία ιδανική για εφαρμογή του FT(Fourier Transform).Ο Shor κατάφερε να απόδειξη ότι το πρόβλημα παραγοντοποίησης ήταν ίδιο με το πρόβλημα εύρεσης της περιόδου σε μια ακολουθία αριθμών. Για να ισχύει αυτή η ισοδυναμία μεταξύ των δύο προβλημάτων θα πρέπει η ακολουθία αριθμών να είναι εκθετικά μεγαλύτερη από τον αριθμό των bit του αριθμού που πρέπει να παραγοντοποιηθεί. Λόγου αυτής της συνθήκης πρακτικά η ισοδυναμία των προβλημάτων δεν παρέχει κανένα ιδιαίτερο προβάδισμα σε ένα κλασικό υπολογιστή καθώς για να δημιουργηθεί μια ακολουθία 2n για έναν αριθμό n απαιτείτε εκθετικός χρόνος. Ωστόσο σε έναν QC μια εκθετικά μεγάλη ακολουθία μπορεί να κωδικοποιηθεί αποκλειστικά σε n qubits με αυτό τον τρόπο υπολογίζετε σε χρόνο πολυωνυμικό ως προς το n. Αφού έχει υπολογιστή η απαιτουμένη ακολουθία ο αλγόριθμος εφαρμόζει στην συνέχεια το QFT για την εύρεση της περιόδου. Τέλος ο αλγόριθμος επιστρέφει ένα δείγμα των πλατών FT, η επιθυμητή πληροφορία είναι πολύ πιθανό να έχει ληφθεί κατά την μέτρηση της εξόδου του αλγορίθμου.

3.2.1 Βήματα υλοποίησης του αλγόριθμου

Έστω ο ακέραιος **N** ο οποίος είναι το γινόμενο των πρώτων αριθμών **p** και **q**.

Βήμα 1ο

Επιλέγουμε ένα τυχαίο ακέραιο αριθμό **g** οπού 0<g<N. Το **g** αρχικά είναι ένας τυχαίος αριθμός ωστόσο από την εξίσωση $\mathbf{g^r = (N \times m) + 1}$, προκύπτει η σχέση $\left(\boldsymbol{g^{\frac{r}{2}} \pm 1}\right) = \mathbf{N \times m}$, όσο το **r** είναι ζυγός υπάρχει μια πιθανότητα να μετατρέψουμε τον τυχαίο αριθμό σε έναν πιθανό αριθμό οποίος έχει κοινό διαιρέτη με το N.

Βήμα 2ο

Εφόσον το **r** είναι άρτιος και εφόσον κανένας από τους αριθμούς $\left(g^{\frac{r}{2}} \pm 1\right)$ δεν είναι πολλαπλάσιο του **N** (οι πιθανότητα να ισχύουν ταυτόχρονα και οι δύο αυτοί περιορισμοί είναι 37,5 %) μπορούμε να εφαρμόσουν το Ευκλείδειο αλγόριθμο για να βρούμε τον μέγιστο κοινό διαιρέτη.

$$\mathbf{gcd}\left(g^{\frac{r}{2}} + 1\,, N\right) = n_1\,, \mathbf{gcd}\left(g^{\frac{r}{2}} - 1\,, N\right) = n_2$$

Οι αριθμοί $\mathbf{n_1}$ και $\mathbf{n_2}$ μπορεί να είναι τελικά και τα ζητούμενα **p** και **g** που ορίσαμε στην αρχή.

Τα παρακάτω βήματα δεν απαιτούν κανέναν κβαντικό υπολογισμό και εκτελούνται σε κλασικούς υπολογιστές. Ωστόσο έχει που απαιτεί να εκτελεστούν κβαντικοί υπολογισμοί είναι στον υπολογισμό της μεταβλητής **r**. Οι κβαντικοί υπολογισμοί που πραγματοποιούνται για να υπολογίσουμε το **r** μπορούν να περιγραφούν μέσω των βημάτων στο Υποκεφάλαιο 3.4.2 .

3.2.2 Quantum Fourier Transform

Μια από τις πιο σημαντικές ρουτίνες που εκτελούνται σε διάφορους κβαντικούς αλγορίθμους αποτελεί ο κβαντικός μετασχηματισμός Fourier (Quantum Fourier Transform,QFT). Η QFT είναι η κβαντική υλοποίηση του διακριτού μετασχηματισμού Fourier πάνω από τα πλάτη μιας κυματοσυνάρτησης. Η δειγματοληψία της εξόδου του μετασχηματισμού Fourier είναι χρήσιμο σε ορισμένες περιπτώσεις για την εύρεση δομής σε μια ακολουθία αριθμών. Το QFT βρίσκει αποτελεσματικά μοτίβα στην ακολουθία εισόδου και προσδιορίζει τη συχνότητα επανάληψης τους. Στον αλγόριθμο του Shor χρησιμοποιείται για την εκτίμηση της κβαντικής φάσης.

Το QFT εφαρμόζεται σε μια κβαντική κατάσταση $|X\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$ και την χαρτογράφει σε μια νέα κβαντική κατάσταση $|Y\rangle = \sum_{k=0}^{N-1} y_k |k\rangle$. Η μαθητικά μπορεί να εκφραστεί ως παρουσιάζεται παρακάτω.

- Ορίζουμε το QFT ως προς μια κβαντική σχέση $\{|x\rangle\} = \{|0\rangle, \dots, |q-1\rangle\}$

$$QFT: |x\rangle \rightarrow \frac{1}{\sqrt{q}} \sum_{y=0}^{q-1} \exp\left\{\frac{2\pi i}{q}(y \times x)\right\} |y\rangle = \frac{1}{\sqrt{q}} \sum_{y=0}^{q-1} \omega^{(x \times y)} |y\rangle$$

- Η εφαρμογή του QFT σε κάποια τυχαία κατάσταση $|\psi\rangle = \sum_x a_x |x\rangle$, πραγματοποιείται μέσω της φόρμουλας :

$$QFT(|\psi\rangle = \frac{1}{\sqrt{q}} \sum_{y=0}^{q-1} \beta_y |y\rangle$$

όπου οι μεταβλητές β_y αποτελούν οι διακεκριμένες τιμές από την εφαρμογή του FT στα πλάτη της α_x.

3.2.3 Κβαντικά βήματα υλοποίησης του αλγόριθμου

Η μεταβλητή **r** μπορεί να υπολογιστή πιο γρήγορα από ένα QC διότι για:
Για κάθε $\boldsymbol{x} \in \boldsymbol{R}^{+}$ *η πράξει* $\boldsymbol{g^x\ mod\ N}$, μετά από έναν αριθμό επαναλήψεων αρχίζει να παρουσιάζει μια περιοδικότητα. Στο αλγόριθμο του Shor ο έλεγχος για τον εντοπισμό της περιοδικότητας πραγματοποιείται σε ένα πλήθος αριθμών από το 1 έως το N^2. Η περιοδικότητα αυτή είναι και η ζητούμενη μεταβλητή **r .**

Δοθέντος μια περιοδική συνάρτηση $f: \{0, \dots., q-1\} \to \{0, \dots., q-1\}$, οπού $q=2^l$ οι συνθήκες περιοδικότητας και l το μέγεθος (πλήθος qubits) των καταχωρητών που θα χρησιμοποιηθούν κατά την εκτέλεση του αλγόριθμου σε έναν QC. Οι συνθήκες περιοδικότητας είναι :

$$f(a) = f(a+r), r \neq 0$$
$$f(a) \neq f(a+s)\ \forall s < r$$

Βήμα 1ο

Αρχικοποιούμε έναν QC με τις καταστάσεις: $|\Phi_l\rangle = |0\rangle^{\otimes 2l}$

Βήμα 2ο

Στην συνέχεια εφαρμόζουμε την πύλη Hadamard σε μια ομάδα από τα l qubits:

$$|\Phi_0\rangle = H^{\otimes l} \otimes \boldsymbol{1}^{\otimes \boldsymbol{l}} |0\rangle^{\otimes 2l} = \left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right)^{\otimes l} \otimes |0\rangle^{\otimes l} = \frac{1}{\sqrt{q}} \sum_{\boldsymbol{a=0}}^{\boldsymbol{q-1}} |\boldsymbol{0}\rangle\, |\boldsymbol{1}\rangle^{\otimes \boldsymbol{l}}$$

Βήμα 3ο

Εφαρμόζουμε την συνάρτηση f σε μια δεύτερη ομάδα από τα l qubits. Για τον αλγόριθμο του Shor η συνάρτηση $f = x^a\ mod\ N$.

$$|\Phi_1\rangle = U_f |\Phi_0\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle\, |f(a)\rangle$$

Βήμα 4ο

Πραγματοποιούμε μια μέτρηση στην συνάρτηση f(a). Η μέτρηση αυτή θα έχει ως αποτέλεσμα τα qubits του 4ου Βήματος να βρίσκονται σε μια κατάσταση:

$$|\Phi_1\rangle_z = \sqrt{\frac{r}{q}}\, \sum_{a:f(a)=z} |a\rangle \quad (1)$$

Ωστόσο η f είναι περιοδική και επιλέγουμε $a_0 = \min\{a | f(a) = z\}$ και εφόσον το r διαιρείται από το q , μπορούμε να ξανά γράψουμε την συνάρτηση (1) , στην παρακάτω μορφή:

$$|\Phi_1\rangle_z = \sqrt{\frac{r}{q}} \sum_{t=0}^{\frac{q}{r}-1} |a_0 + (t \times r)\rangle$$

Βήμα 5ο

Εφαρμόζουμε QFT:

$$|\tilde{\Phi}\rangle = QFT^{-1} = \sqrt{\frac{r}{q}} \sum_{t=0}^{\frac{q}{r}-1} \frac{1}{\sqrt{q}} \sum_{c=0}^{q-1} \exp\left\{-\frac{2\pi i}{q}(a_0 + rt)c\right\}|c\rangle$$

$$= \sqrt{\frac{r}{q^2}} \sum_{c=0}^{q-1} \exp\left\{-\frac{2\pi i}{q} a_0 c\right\} \underbrace{\sum_{t=0}^{\frac{q}{r}-1} \exp\left\{-\frac{2\pi i}{q}(t \times r \times c)\right\}}_{a_c} |c\rangle$$

Αν $r \times c = k \times q$ για κάποιο $k \in \mathbb{N}$, τότε :

$$a_c = \frac{q}{r}$$

Η πιθανότητα να μετρήσουμε μια συγκριμένη κβαντική κατάσταση $c' = \frac{kq}{r}$ είναι:

$$P[c'] = \left|\langle c'|\tilde{\Phi}\rangle\right|^2 = \frac{r}{q^2}|a_{c'}|^2 = \frac{r}{q^2} \times \frac{q^2}{r^2} = \frac{1}{r}$$

Βήμα 6ο

Η έξοδος του κβαντικού μέρους του αλγόριθμου είναι ένας φυσικός αριθμός της μορφής $\frac{kq}{r}$, $k \in \mathbb{N}$.

3.2.4 Ανάλυση κβαντικών βημάτων

Βήμα 1ο

Ορίζουμε δύο καταχωριτές ο πρώτος μεγέθους l qubits για τα δεδομένα (Φ_0) και έναν δεύτερο μεγέθους n=$\log_2 N$ qubits για τις πράξεις (Φ_1). Στην συνέχεια αρχικοποιούμε τον Φ_0 στις κβαντικές καταστάσεις

$$|\Phi_0\rangle = \underbrace{|0\rangle\,|0\rangle\,|0\rangle \ldots |0\rangle}_{l}$$

Βήμα 2ο

Περνάμε τα qubits του καταχωριτή Φ_1 από μια πύλη Hadamard και τα θέτουμε σε μια κατάσταση κβαντικής υπέρθεσης (superposition):

$$|\Phi_1\rangle = \underbrace{|0\rangle + |1\rangle + |2\rangle + |3\rangle \ldots |N^2\rangle}_{n}$$

Βήμα 3ο

Υψώνουμε τον αριθμό g στην Φ_1 ομάδα των qubits και διαιρούμαι με το N.

$$|\Phi_1\rangle = \underbrace{|\frac{q^0}{N}\rangle + |\frac{q^1}{N}\rangle + |\frac{q^2}{N}\rangle + |\frac{q^3}{N}\rangle \dots |\frac{q^{N^2}}{N}\rangle}_{n}$$

Στην συνέχεια αποθηκεύουμε το υπόλοιπο της διαίρεσης στην Φ_0 ομάδα από qubits.

$$|\Phi_0\rangle = \underbrace{|rem_0\rangle\,|rem_1\rangle\,|rem_2\rangle \dots |rem_{N^2}\rangle}_{l}$$

Σε αυτό το βήμα έχουμε καταφέρει μια κβαντική διαπλοκή (quantum entanglement) μεταξύ των ομάδων Φ_1 και Φ_0.

$$\underbrace{\Phi = |0\rangle|rem_0\rangle + |1\rangle|rem_1\rangle + |2\rangle|rem_2\rangle + |3\rangle|rem_3\rangle + \dots}_{l}$$

Βήμα 4ο

Στο βήμα αυτό καταμετράμε την κβαντική διαπλοκή μεταξύ των δύο ομάδων. Ωστόσο για να μην μετρήσουμε κάποια τυχαία τιμή , μετράμε μόνο κάποια κατάσταση $|rem\rangle$ στην Φ ομάδα. Μετρώντας στην ομάδα αυτή έχουμε μια κατάσταση $|rem\rangle$ η οποία εμφανίζεται επαναλαμβανόμενα με μια περίοδο r.

$$\underbrace{|i\rangle|rem\rangle + |j\rangle|rem\rangle + |k\rangle|rem\rangle \dots .}_{l} = \underbrace{|i, rem\rangle + |i + r, rem\rangle + |i + 2r, rem\rangle \dots .}_{l}$$

Η πραγματοποίηση μιας καταμέτρησης στην κατάσταση Φ έχει ως αποτέλεσμα το rem να είναι ίδιο οπότε μπορεί να αγνοηθεί έχοντας στο τέλος τις κβαντικές καταστάσεις:

$$\underbrace{|i\rangle + |i + r\rangle + |i + 2r\rangle \dots . (1).}_{l}$$

Βήμα 5ο

Εφαρμόζουμε QFT στην κατάσταση (1) του τέταρτου βήματος. Εφαρμόζοντας QFT θα έχουμε τις νέες καταστάσεις :

$$\underbrace{\widetilde{\Phi} = |c_1\frac{1}{r}\rangle + |c_2\frac{1}{r}\rangle + |c_3\frac{1}{r}\rangle \dots .}_{l}$$

Βήμα 6ο

Μετράμε την κατάσταση του πέμπτου βήματος και υπολογίζουμε το $\frac{1}{\frac{r}{1}} => \frac{r}{1} => r$.

3.2.5 Παράδειγμα εκτέλεσης αλγόριθμου

Έστω N=21 άρα η δύο πρώτοι αριθμοί που ψάχνουμε είναι το 3 και το 7.

Βήμα 1ο

Διαλέγουμε έναν τυχαίο ακέραιο αριθμό x , οπού 1<x<n. Επίσης το x δεν πρέπει να είναι διαιρέτης του N. Έστω x=2.

Βήμα 2ο

Υπολογίζουμε το q:

$$q = 2^l, \text{οπού } N^2 \leq q \leq 2N^2$$

Έστω l=9 qubits , άρα το q=512 .

Βήμα 3ο

Υπολογίζουμε το μέγεθος των καταχωρητών. Χρειαζόμαστε δύο καταχωριτές ένα μεγέθους l για τα δεδομένα (Φ_0) και έναν δεύτερο μεγέθους n=$\log_2$N=5 qubits για τις πράξεις (Φ_1). Στην συνέχεια αρχικοποιούμε τον Φ_0 στις κβαντικές καταστάσεις

$$|\Phi_0\rangle = |0\rangle_{r1}\,|0\rangle_{r2} = |0\rangle^{\otimes l} \Rightarrow |\Phi_i\rangle = \underbrace{|0\rangle\,|0\rangle\,|0\rangle \ldots |0\rangle}_{l=9\ qubits}$$

Βήμα 4ο

Στην συνέχεια εφαρμόζουμε την πύλη Hadamard σε όλα τα qubits του καταχωριτή Φ_1.

$$|\Phi_1\rangle = \frac{1}{\sqrt{q}} \sum_{\alpha=0}^{q-1} |\alpha\rangle|0\rangle \Rightarrow |\Phi_0\rangle = \frac{1}{\sqrt{512}} \sum_{a=0}^{511} |a\rangle|0\rangle$$

Βήμα 5ο

Αρχικοποιούμε τον καταχωρήτη Φ_1 με τις κβαντικές καταστάσεις υπερθέσεις που περιλαμβάνουν όλες τις καταστάσεις x^a mod q:

$$|\Phi_1\rangle = \frac{1}{\sqrt{q}} \sum_{a=0}^{q-1} |a\rangle|x^a \ mod\ 21\rangle = \frac{1}{\sqrt{512}} \sum_{\alpha=0}^{511} |\alpha\rangle|2^a \ mod\ 21\rangle =$$

$$= \frac{1}{\sqrt{512}} \underbrace{(|0\rangle|1\rangle + |1\rangle|2\rangle + |2\rangle|4\rangle + |3\rangle 8\rangle + |4\rangle|16\rangle + \cdots..)(1)}_{n=5\ qubits}$$

|0⟩|1⟩	|1⟩|2⟩	|2⟩|4⟩	|3⟩|8⟩	|4⟩|16⟩	|5⟩|11⟩	(1)
|6⟩|1⟩	|7⟩|2⟩	|8⟩|4⟩	|9⟩|8⟩	|10⟩|16⟩	|11⟩|11⟩	(2)
|12⟩|1⟩	|13⟩|2⟩	|14⟩|4⟩	|15⟩|8⟩	|16⟩|16⟩	|17⟩|11⟩	(3)
…….	……	…….	……	……	……	…..

Όπως παρατηρούμε στις καταστάσεις έτσι όπως έχουν καταχωρηθεί στον πίνακα εμφανίζετε ένα επαναλαμβανόμενο μοτίβο. Λόγου του μοτίβου αυτού μπορούμε να ξανά γράψουμε την σχέση (1) με την παρακάτω ισοδύναμη μορφή:

$$|\Phi_1\rangle = \frac{1}{\sqrt{512}}[\,(|0\rangle + |6\rangle + |12\rangle \dots + |510\rangle)|1\rangle + (|1\rangle + |7\rangle + |13\rangle \dots + |511\rangle)|2\rangle$$
$$+ (|2\rangle + |8\rangle + |14\rangle \dots + |506\rangle)|4\rangle + (|3\rangle + |9\rangle + |15\rangle \dots + |507\rangle)|8\rangle$$
$$+ (|4\rangle + |10\rangle + |16\rangle \dots + |508\rangle)|16\rangle + (|5\rangle + |11\rangle + |17\rangle \dots + |509\rangle)|11\rangle\,]$$

Βήμα 6ο

Εκτελούμε QFT σε μια από την καταστάσεις του Φ_1 , με την εφαρμογή του QFT μπορούμε να υπολογίσουμε το μοτίβα περιόδου των καταστάσεων. Διότι αυτό που μας ενδιαφέρει είναι η περίοδος μιας κατάστασης και όχι η τιμή της κατάστασης.

Έστω ότι εφαρμόζουμε QFT στην κατάσταση με τιμή 2.

$$QFT(|\Phi_1\rangle) => |\tilde{\Phi}\rangle = QFT\left(\frac{1}{\sqrt{86}}\sum_{a=0}^{85}|6a+1\rangle\right)|2\rangle$$
$$= \frac{1}{\sqrt{512}}\sum_{j=0}^{511}\left(\left[\frac{1}{\sqrt{86}}\sum_{j=0}^{511}e^{-2\pi i\frac{6ja}{512}}\right]e^{-2\pi i\frac{j}{512}}|j\rangle\right)|2\rangle$$

Βήμα 7ο

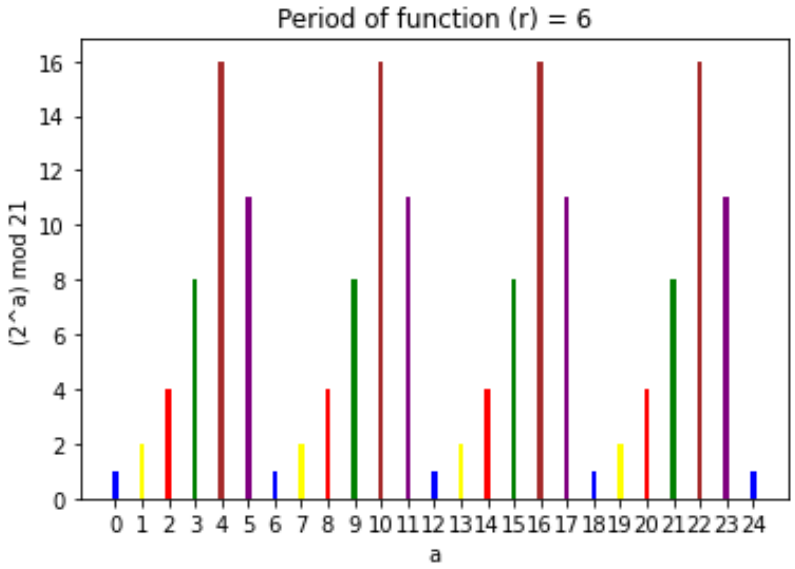
Υπολογίζουμε την πιθανότητα για μια κβαντική κατάσταση $|j, x^k \bmod n\rangle$. Για το παράδειγμα μας η σχέση διαμορφώνεται ως $j = |j, 2\rangle$

$$p(j) = \frac{1}{512 \times 86}\left|\sum_{a=0}^{85}e^{-2\pi i\frac{6ja}{512}}\right|^2$$

Οι πιο πιθανές τιμές για το j={0,85,171,256,341,427}

	J	Phase	Fraction	r
1	0	0/512=0	0	0
2	85	85/512=0.1660	1/6	6
3	171	171/512=0.3339	1/3	3
4	256	256/512=0.5	1/2	2
5	341	341/512=0.6660	2/3	3
6	427	427/512=0.8339	5/6	6

Όπως παρατηρούμε και στο πίνακα από πάνω δύο από τις τιμές του j μας επιστρέφουν το ζητούμενο r δηλαδή το 6 και το γνωρίζουμε αυτό διότι αν παρατηρήσουμε το παρακάτω διάγραμμα στο Σχήμα 3-1, ανά έξι αριθμούς επαναλαμβάνονται οι έξοδοι της εξίσωσης $2^{\alpha}\ mod\ 21$.



Σχήμα 3-1: Σχηματική απεικόνιση της περιοδικότητας για την συνάρτησης 2^{α} mod 21

Βήμα 8ο

Διαλέγουμε το r από τα αποτελέσματα που υπολογίσιμα :

Στην συνέχεια εφαρμόζουμε τα μη κβαντικά βήματα για να βρούμε τους ζητούμενους πρώτους αριθμούς.

$$(x^{\frac{r}{2}}\ mod\ n) - 1 = (11^3\ mod\ 21) - 1 = 7$$

$$(x^{\frac{r}{2}}\ mod\ n) + 1 = (11^3\ mod\ 21) + 1 = 9$$

Υπολογίζουμε τους διαιρέτες gcd(7,21)=7 και gcd(9,21)=3

Ο αλγόριθμος κατάφερε να βρει τους πρώτους αριθμούς για N=21 => 21=7 x 3.

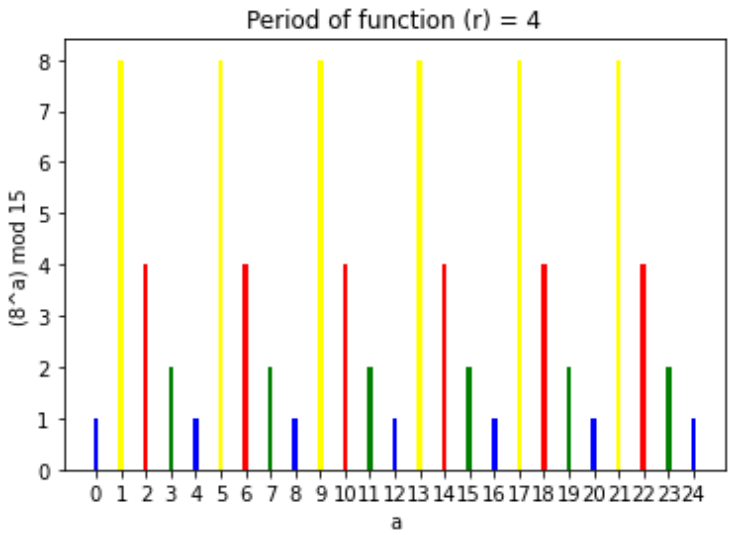
3.3 Η βιβλιοθήκη Qiskit

Το Qiskit αποτελεί ένα ανοικτού κώδικα kit ανάπτυξης κώδικα , το οποίο αναπτύχθηκε και συντηρείται από το τμήμα Έρευνας και Ανάπτυξης της IBM , η πρώτη έκδοση του Qiskit έγινε διαθέσιμο το 2016 για την γλώσσα προγραμματισμού Python. Το Qiskit παρέχει εργαλεία για τη δημιουργία και το χειρισμό κβαντικών προγραμμάτων σε επίπεδο κυκλωμάτων, παλμών και αλγορίθμων αλλά και την εκτελέσει σε πρωτότυπες κβαντικές συσκευές που βρίσκονται στο IBM Quantum Experience ή σε προσομοιωτές QC σε τοπικό υπολογιστή. Κύριος στόχος του Qiskit αποτελεί η δημιουργία μιας στοίβα λογισμικού που μπορεί να εξυπηρετήσει οποιοδήποτε επιθυμεί να χρησιμοποιήσει έναν QC. Ααανεξάρτητα από το επίπεδο δεξιοτήτων ή την περιοχή ενδιαφέροντός του χρήστη, το Qiskit επιτρέπει τον εύκολο προγραμματισμό πειραμάτων και εφαρμογών και την εκτέλεση τους σε πραγματικούς QCs ή κλασικούς προσομοιωτές. Μέσω του Qiskit μπορεί να αναπτυχθεί κβαντικό λογισμικό τόσο σε επίπεδο κώδικα μηχανής του OpenQASM που αποτελεί μια επιτακτική γλώσσα προγραμματισμού για την περιγραφή κβαντικών κυκλωμάτων, όσο και σε αφηρημένα επίπεδα το οποίο αποτελεί μια εναλλακτική κατάλληλη για τελικούς χρήστες χωρίς τεχνογνωσία κβαντικών υπολογιστών.

3.3.1 Υλοποίηση του αλγορίθμου του shor με χρήση της Qiskit

Στο παρακάτω υποκεφάλαιο θα χρησιμοποιήσουμε τμήματα κώδικα που προσφέρονται από την επίσημη ιστοσελίδα της Qiskit (Πηγή κώδικα: https://learn.qiskit.org/course/ch-algorithms/shors-algorithm) για εκπαιδευτικούς και ερευνητικούς σκοπούς. Ο όρος κομμάτια κώδικα είναι ιδανικός για να περιγράψει τον προγραμματισμό σε ένα QC. Οι κώδικες που γράφονται για QCs μηχανήματα δεν συγγράφονται σε γλώσσες υψηλού επίπεδου αλλά σε γλώσσες περιγραφείς υλικού (HDL) όπως για παράδειγμα οι γλώσσες Verilog και MIPS. Όπως και στις δύο διάσημες γλώσσες για τους κλασικούς υπολογιστές χρησιμοποιούμε κείμενο για να περιγράψουμε κλασικά κυκλώματα , στην συνέχεια θα χρησιμοποιήσουμε την γλώσσα Python σε συνδυασμό με την βιβλιοθήκη Qiskit για να περιγράψουμε κβαντικά κυκλώματα.

Για το παράδειγμα μας θα χρησιμοποιήσουμε το N=15 και αναζητούμε τους αριθμούς 3 και 5 οπού είναι και οι πρώτοι διαιρέτες του αριθμού 15 . Ως αρχική τυχαία επιλογή διαλέγουμε το αριθμό 8.



Σχήμα 3-2: Σχηματική απεικόνιση της περιοδικότητας για την συνάρτησης 8^a mod 15

Όπως παρατηρούμε στο Σχήμα 3-2 ο αλγόριθμος περιμένουμε να επιστρέψει τον αριθμό 4.

Το παρακάτω κομμάτι περιγράφει ένα κβαντικό κύκλωμα που υπολογίζει την πράξει *a mod* 15, το οποίο είναι βελτιστοποιημένο για τις τυχαίες τιμές {2,4,7,8,11,13}.

```
import numpy as np
from qiskit import QuantumCircuit, Aer, transpile
from qiskit.visualization import plot_histogram
import pandas as pd
from fractions import Fraction

# Specify variables
Q_COUNT = 10  # number of counting qubits
Q=15
e=8


def c_amod15(i, power):
    """Controlled multiplication by e mod 15"""
    if a not in [2,4,7,8,11,13]:
        raise ValueError("'e' must be 2,4,7,8,11 or 13")
    k = QuantumCircuit(4)
    for _iteration in range(power):
        if e in [2,13]:
            k.swap(2,3)
            k.swap(1,2)
            k.swap(0,1)
        if e in [7,8]:
            k.swap(0,1)
            k.swap(1,2)
            k.swap(2,3)
        if e in [4, 11]:
            k.swap(1,3)
            k.swap(0,2)
        if e in [7,11,13]:
            for q in range(4):
                1.x(q)
    k = k.to_gate()
    k.name = f
    c_k = k.control()
    return c_k
```

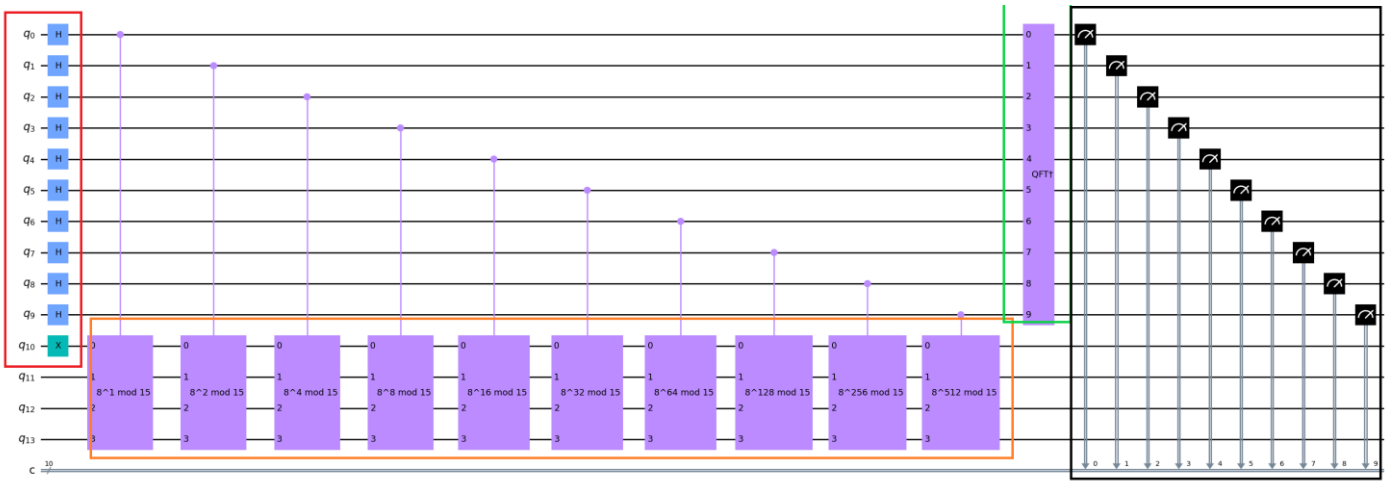
Το παρακάτω μπλοκ κώδικα περιγράφει ένα κβαντικό κύκλωμα που εκτελεί QFT.

```

def qft_dagger(q):
    """q-qubit QFTdagger the first q qubits in circ"""
    nc = QuantumCircuit(q)
        for qubit in range(q//2):
        nc.swap(qubit, q-qubit-1)
    for i in range(q):
        for j in range(i):
            nc.cp(-np.pi/float(2**(i-j)), j, i)
        nc.h(i)
    nc.name = "QFT†"
    return nc

```

Τέλος συνδυάζουμε τα παραπάνω δύο μπλοκ κώδικα και παράλληλα ρυθμίζουμε το τελικό κύκλωμα που θα τρέξει τον αλγόριθμο του Shor.

```
# Create QuantumCircuit with N_COUNT counting qubits
nc = QuantumCircuit(Q_COUNT + 4, Q_COUNT)

# Initialize counting qubits
for i in range(Q_COUNT):
    nc.h(i)

# And auxiliary register in state |1>
nc.x(Q_COUNT)

for i in range(Q_COUNT):
    nc.append(c_amod15(e, 2**i),
              [i] + [j+Q_COUNT for j in range(4)])

# Do inverse-QFT
nc.append(qft_dagger(Q_COUNT), range(Q_COUNT))

# Measure circuit
nc.measure(range(Q_COUNT), range(Q_COUNT))
nc.draw(fold=-1)

```

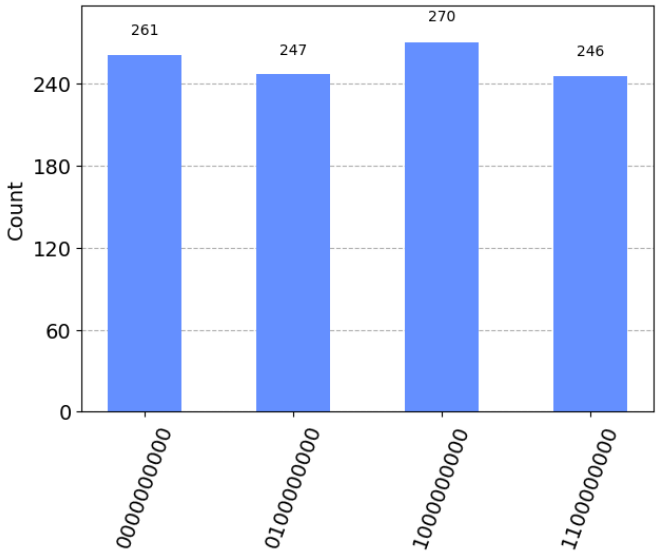
Το κύκλωμα του κώδικα έχει την μομφή που απεικονίζετε στο Σχήμα 3-3.



Σχήμα 3-3: Κβαντικό κύκλωμα που εκτελεί τον αλγόριθμο του Shor για N=15.

Αναλύοντας το Σχήμα 3-3 παρατηρούμε τέσσερις ζώνες . Η πρώτη (κόκκινο ορθογώνιο) είναι τα qubits $|\Phi\rangle$ τα οποία είναι εκείνα που θα εξεργαστούμε όπως περιγράφεται στα Υποκεφάλαια 3.2.3 και 3.2.4 , στην συνέχεια είναι τα qubits τις εξόδου $I\tilde{\Phi}\rangle$, του αλγόριθμου (μαύρο ορθογώνιο) τα οποία είναι εκείνα τα οποία όταν μετρήσουμε θα έχουμε πιθανότητα την πληροφορία που χρειαζόμαστε δηλαδή την περίοδο της συνάρτησης. Τέλος στο κίτρινο και στο πράσινο ορθογώνιο απεικονίζονται σχηματικά τα κυκλώματα που περιγράφονται με τα μπλοκ κώδικα.

3.3.2 Ανάλυση αποτελεσμάτων εξόδου του αλγορίθμου



Σχήμα 3-4: Αποτελέσματα εξόδου αλγορίθμου

Αναλύοντας τα αποτελέσματα του ιστογράμματος που απεικονίζετε στο Σχήμα 3-4 , προκύπτει ο παρακάτω πίνακας:

	Register Output	Phase	Fraction	r
0	1100000000(bin)=768(dec)	768/1024=0.75	3/4	4
1	0100000000(bin)=256(dec)	256/1024=0.25	1/4	4
2	1000000000(bin)=512(dec)	512/1024=0.5	1/2	2
3	0000000000(bin)=0(dec)	0/1024=0.0	0/1	1

Πρακτικά αυτό που μετράμε στο τέλος του αλγόριθμου είναι το πλήθος των συγκριμένων φάσεων που έχουν τα qubits. Στην συνέχεια την φάση αυτή την μετατρέπουμε σε κλάσμα και τέλος από τα κλάσματα που έχουμε ένας από τους παρανομαστές μπορεί να είναι το ζητούμενο r. Στο παράδειγμα μας βρέθηκε και ήταν το r=4. Στο υποκεφάλαιο 3.2.1 αναλύθηκε για πιο λόγο ο αλγόριθμος έχει μια πιθανότητα 37,5% να επιστρέψει την σωστή περίοδο μιας συνάρτησης , σε περίπτωση που δεν έχουμε βρει την σωστή περίοδο ξανά τρέχουμε τον ίδιο αλγόριθμο που έχουμε σχεδίαση.

Κεφάλαιο 4 : Αλγόριθμοι κρυπτογραφήσεις θωρακισμένοι από τους QCs

4.1 Μέτα κβαντικοί αλγόριθμοι κρυπτογραφίας(PQC)

Στην μετά κβαντική κρυπτογραφία(Post-quantum cryptography) υποθέτουμε ότι ο εισβολέας έχει πρόσβαση σε έναν λειτουργικό QCs,τα μετά κβαντικά κρυπτοσυστήματα θα πρέπει να παραμείνουν ασφαλής ακόμα και σε αυτό το σενάριο. Ο παραπάνω ερευνητικός τομέας είναι σχετικά πρόσφατος αλλά, έχει καταφέρει να σημειώσει κάποιες επιτυχίες στην εύρεση δύσκολων μαθηματικών προβλημάτων και την δημιουργία κρυπτογραφικών συστημάτων γύρω από αυτές. Για τις οποίες δεν γνωρίζουμε κάποιο κβαντικό αλγόριθμο που προσφέρει κάποια σημαντική επιτάχυνση στην παραβίαση τους.

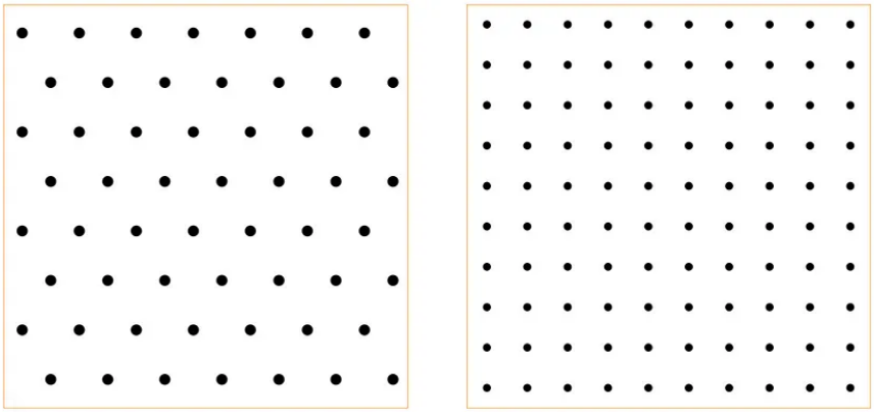
Η ανάπτυξη ενός νέου κρυπτογραφικού συστήματος απαιτεί χρόνια ερευνών και δοκιμών από διαφόρους ερευνητές και επιστήμονες. Κάποιες από τις προτάσεις για PQCs έχουν περάσει από πολλά χρόνια εξετάσεων, ωστόσο επέφεραν σοβαρό κόστος, ιδίως στο φόρτο που πρέπει να διαχειριστούν τα δίκτυα. Άλλες προτάσεις αποτελούν πιο ρεαλιστικές για την ανάπτυξη ενός πρακτικού μετά κβαντικού κρυπτοσυστήματος, αλλά η ασφάλειά τους αποτελεί ακόμα ασαφής και πιθανά κάποιες από αυτά μπορεί στο μέλλον να παρουσιάσουν κάποιο κενό ασφάλειας. Για παράδειγμα οι μετά κβαντικοί αλγόριθμοι κρυπτογράφησης παραμένουν ακόμα τρωτοί στις επιθέσεις που περιγράψαμε στα υποκεφάλαια 2.2.2, 2.3.2 και 2.5.3 για τους κλασικούς αλγορίθμους ασύμμετρης κρυπτογράφησης, μπορεί ένας μετά κβαντικός αλγόριθμος να είναι ασφαλής σε brute force επιθέσεις από κβαντικούς και κλασικούς υπολογιστές αλλά υπάρχουν και άλλων ειδών επιθέσεις που μπορούν να πραγματοποιηθούν από ένα κλασικό υπολογιστή όπως για παράδειγμα οι επιθέσεις πλευρικού καναλιού. Η ανάπτυξη ενός κρυπτογραφικού συστήματος εγείρει ερωτήματα για το κατά πόσο ο πραγματικός κόσμος εναρμονίζεται με τα μαθηματικά μοντέλα του συστήματος για τις πραγματικές δυνατότητες των χρηστών να καταφέρουν να τρέξουν το κρυπτοσυστήματα και τις δυνατότητες που διαθέτει ο εισβολέας για να το παραβίαση. Μεγάλο μέρος της κρυπτογραφικής έρευνας στοχεύει στην εύρεση της μέγιστης ασφάλειας που μπορεί να επιτευχθεί υπό διάφορους περιορισμούς στο πραγματικό κόστος υλοποίησης ενός κρυπτοσυστήματος.

Πολυάριθμοι φορείς και οργανισμοί τυποποίησης προτείνουν την άμεση μετάβαση σε κρυπτοσυστήματα που να μην παραμένουν τρωτά απέναντι σε επιθέσεις από κβαντικούς υπολογιστές. Η συσπείρωση τόσο πολλών οργανισμών γύρω από την άμεση ανάγκη για νέους αλγορίθμους κρυπτογράφησης αποτελεί ένα σημαντικό βήμα προς την ανάπτυξη. Διότι πολλές εφαρμογές κρυπτογραφίας απαιτούν από όλα τα μέρη να χρησιμοποιούν το ίδιο κρυπτογραφικό σύστημα, επομένως η τυποποίηση αποτελεί απαραίτητη προϋπόθεση για την ευρεία ανάπτυξη κρυπτογραφικών πρωτοκόλλων βασισμένων σε μετά κβαντικούς αλγορίθμους κρυπτογράφησης, πολλές φορές τα de facto πρότυπα τίθενται από την ευρεία χρήση τους και όχι από φορείς τυποποίησης. Ωστόσο οι επίσημες διαδικασίες τυποποίησης θεωρείτε ευρέως αποδεκτό ότι μειώνουν τυχόν κενά ασφάλειας. Ο οργανισμός NIST είχε ανοίξει πρόσκληση υποβολής υποψηφίων PQC αλγόριθμων για τυποποίηση, με προθεσμία υποβολής το Νοέμβριο του 2017 ένω τον Ιούλιο του 2022 ανακοίνωσε τέσσερις υποψήφιους προς τυποποίηση μετά κβαντικούς αλγόριθμους κρυπτογράφησης για διάφορες λειτουργίες(Πηγή: NIST Announces First Four Quantum-Resistant Cryptographic Algorithms | NIST). Στην συνέχεια της εργασίας θα αναλυθούν

οι παρακάτω τέσσερις αλγόριθμοι διότι ο NIST όντας κρατικός οργανισμός των ΗΠΑ έχει ως επακολουθώ, πολλά από τα πρότυπα που επιβάλει να υιοθετούνται και να εφαρμόζονται σε μερικές από τις μεγαλύτερες εταιρίες του κόσμου στο κομμάτι του software άλλα και του hardware, όποτε είναι βέβαιο και αναμενόμενο ότι ως αλγόριθμοι θα υπάρξουν σε διάφορα συστήματα στο μέλλον. Υπάρχουν και άλλοι φορείς τυποποίησης που έχουν προτείνει πρότυπα, για παράδειγμα το πρότυπο SC27 WG2 από τον διεθνή οργανισμό ISO (Πηγή: ISO/IEC JTC 1/SC 27 - Information security, cybersecurity and privacy protection) ή το πρότυπο KMIP από τον οργανισμό OASIS OPEN(Πηγή: OASIS Key Management Interoperability Protocol (KMIP) TC | OASIS (oasis-open.org)) .

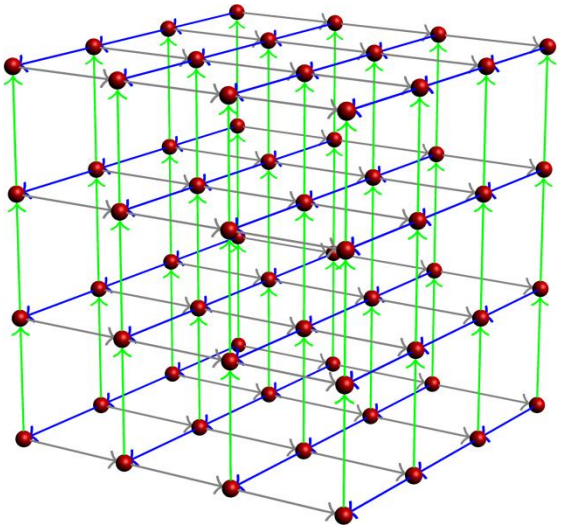
4.2 Μέτα κβαντικοί αλγόριθμοι κρυπτογραφίας βασισμένοι σε συστήματα πλέγματος (LBC)

Ένα «πλέγμα» στα αγγλικά «lattice» αποτελεί ένα διακριτό σύνολο σημείων στο χώρο τα οποία επεκτείνονται στο άπειρο με την ιδιότητα ότι το άθροισμα δύο σημείων του πλέγματος είναι επίσης ένα σημείο στο πλέγμα. Τα πλέγματα μπορούμε να τα συναντήσουμε φυσικά σε διάφορους κλάδους των μαθηματικών και της φυσικής. Μαθηματικά ένα πλέγμα $\mathcal{L}$ του συνόλου $\mathbb{R}^n$ είναι εξ ορισμού μια διακριτή υποομάδα του $\mathbb{R}^n$, ωστόσο για τις ανάγκες δημιουργίας ενός αλγορίθμου κρυπτογραφίας LBC χρειαζόμαστε μόνα τα πλέγματα πλήρους κατάταξης δηλαδή πλέγματα που εκτείνονται στο $\mathbb{R}^n$ αλλά έχουν μόνο πραγματικούς συντελεστές. Πρακτικά για τις ανάγκες της κρυπτογραφίας ερευνώνται μόνο τα πλέγματα που ανήκουν στο σύνολο $\mathcal{L} \subseteq \mathbb{Z}^n$.



Εικόνα 4-1: Διαφορετικού μεγέθους δισδιάστατα πλέγματα

Πηγή: What is Lattice-Based Cryptography & Why You Should Care | by Wickr | Wickr Crypto + Privacy Blog | Medium



Εικόνα 4-2: Τρισδιάστατο πλέγμα

Πηγή: graphics - How to make a 3D Lattice? - TeX - LaTeX Stack Exchange

4.2.1 SVP

Η εύρεση ενός «σύντομου» διανύσματος σε κάποιο δεδομένο πλέγμα αποτελεί ένα από τα ποιο γνωστά υπολογιστικά προβλήματα στα πλέγματα. Όλοι οι τρέχοντες κλασικοί αλγόριθμοι για να επιλύσουν το παραπάνω πρόβλημα απαιτούν εκθετικό χρόνο για πολυδιάστατα πλέγματα, κάποια στοιχεία επίσης υποδηλώνουν ότι το πρόβλημα χρειάζεται εκθετικό χρόνο και σε έναν κβαντικό υπολογιστή για να επιλυθεί. Αρκετά κρυπτοσυστήματα LBC έχουν δημιουργηθεί έχοντας ως βάση για την ασφάλεια τους το SVP. Μαθηματικά το SVP ορίζεται με το παρακάτω ορισμό:

Ένα πλέγμα $\mathcal{L}$ ορίζεται ως το σύνολο των γραμμικών συνδυασμών πάνω από τους ακέραιους $\mathbb{Z}$ ενός συνόλου γραμμικών ανεξάρτητων διανυσμάτων $b_1, \dots \dots, b_2 \epsilon \mathbb{R}^m$.

$$\mathcal{L} := \left\{ \sum_{i=1}^{n} a_i b_i : a_i \in \mathbb{Z}, i = 1, \dots \dots, n \right\}$$

Δοθέντος μιας βάσης $\mathbb{B}$ ενός πλέγματος , $\mathcal{L}$ πρέπει να βρεθεί ένα μη μηδενικό $v \in \mathcal{L}$ τέτοιο ώστε $\|v\| \leq \gamma \lambda_1(\mathcal{L})$

Η βάση ενός πλέγματος $\mathcal{L}$ είναι μια σειρά $\mathbb{B} = (b_1, b_2, \dots, b_n)$ τέτοια ώστε

$$\mathcal{L} = \mathcal{L}(\mathbb{B}) = \mathbb{B} \times \mathbb{Z}^n = \{ \textstyle\sum_{i=1}^{n} c_i b_i : c_i \in \mathbb{Z} \}$$

Μπορεί το πρόβλημα αυτό να φαίνεται σχετικά εύκολο να λυθεί. Ωστόσο, αν η βάση που δίνεται αποτελείται από μακρά διανύσματα, δεν είναι αμέσως σαφές πώς μπορούν να συνδυαστούν για να δημιουργηθεί ένα σημείο με μικρές μόνο συντεταγμένες, δηλαδή ένα σύντομο διάνυσμα. Επιπλέον, όταν πρόκειται για εφαρμογές που προορίζονται να χρησιμοποιηθούν για πρακτικές εφαρμογές κρυπτογραφίας, τα πλέγματα έχουν υψηλές διαστάσεις (π.χ. πλέγμα με 10.000 διαστάσεις), πρακτικά αυτό σημαίνει ότι τα σημεία μας έχουν στην πραγματικότητα 10.000 συντεταγμένες. Η εύρεση ενός συνδυασμού των βασικών διανυσμάτων που κάνει ταυτόχρονα και τις 10.000 συντεταγμένες που δημιουργούνται μικρές αποδεικνύεται αρκετά

δύσκολο, τόσο πολύ που καθιστά την λύση του προβλήματος αρκετά δύσκολη και κανένας δεν γνωρίζει κάποιο πρακτικό κβαντικό αλγόριθμο που θα μπορούσε να λύση γρήγορα αυτό το πρόβλημα . Ο ποιο αποδοτικός κλασικός αλγόριθμος για το SVP είναι ο LLL (Lenstra–Lenstra–Lovász lattice basis reduction algorithm) με πολυπλοκότητα χρόνου $O(d^5 n log_B^3)$, οπού B το μεγαλύτερο σε μήκος b_i .

4.2.2 SIS

Το SIS πρόβλημα ανάγεται σε μια ειδική υποκατηγορία του SVP το SVP_γ . Στο SVP_γ πρόβλημα το ζητούμενο είναι η εύρεση ενός σύντομου διανύσματος μέσα σε ένα πλέγμα, όπου το μέγεθος του είναι το πολύ ένας πολλαπλασιαστικός παράγοντας, ο οποίος προσεγγίζει το $\gamma \geq 1$του πιο μικρού μη μηδενικού διανύσματος που υπάρχει είδη στο πλέγμα που πραγματοποιείτε η αναζήτηση. Το SIS πρόβλημα αποτελεί την αναζήτηση για λύση στο SVP_γ σε μια ειδική κατηγορία πλεγμάτων τα g-ray. Μαθηματικά το SIS_{n,m,q,N^β} πρόβλημα ορίζεται :

Δοθέντος ένα $A \subseteq \mathbb{Z}_q^{m \times n}$ να βρεθεί το πιο σύντομο $\vec{z} \in \mathbb{Z}^m$ στο g-ary πλέγμα $A^\perp(A)$, το οποίο ικανοποιεί τις σχέσεις $A\vec{z} \equiv \vec{0}\, mod\ q$ και $\|z\| \leq \beta$.

Το SIS πρόβλημα είναι ιδανικό για ηλεκτρονικές υπογραφές διότι η συνάρτηση $\vec{x} \to A\vec{x}$ είναι μια πολύ καλή μονόδρομη συνάρτηση κατακερματισμού ανθεκτική σε συγκρούσεις.

4.2.3 LWE

Το LWE (μάθηση με σφάλμα) είναι η επίλυση ενός είδους τυχαίων γραμμικών εξισώσεων κάτω από μια δεδομένη κατανομή πιθανοτήτων. Το πρόβλημα αυτό μπορεί να θεωρηθεί και ως διπλή μορφή του προβλήματος SIS, η πολυωνυμική ισοδυναμία μεταξύ του μέσου προβλήματος LWE και των σκληρών lattice προβλημάτων είναι μια γενίκευση της αναγωγής Ajtai. Η εφαρμογές της LWE στην σύγχρονη LBC κρυπτογραφία είναι αρκετά σημαντικές, όπως για παράδειγμα η πλήρως ομομορφική κρυπτογράφηση που βασίζεται στο LWE.

Για να κατανοήσει κάποιος το πρόβλημα LWE, πρέπει πρώτα να κατανοήσει το πρόβλημα LPE:

Έστω $\mathbb{Z}_2 = \{0,1\}$ ένα πεπερασμένο πεδίο με δύο στοιχεία $n \geq 1\ και\ \varepsilon \geq 0$. Η κατανομή του ξ με παράμετρο το ε στο $\mathbb{Z}_2$ πεπερασμένο πεδίο είναι:

$$\Pr\{\xi = 0\} = 1 - \varepsilon, \Pr\{\xi - 1\} = \varepsilon$$

Εφόσον $a, b \in \mathbb{Z}_2$, η πιθανότητα το a και b να έχουν την ίδια ισοτιμία είναι 1-ε δηλαδή :

$$\Pr\{\alpha \equiv b\ (mod\ 2)\} = 1 - \varepsilon => a \equiv_\varepsilon b$$

Το LWE ορίζεται ως: Δοθέντος ένα ανεξάρτητο διάνυσμα $m = \{a_1, a_2, \dots, a_m\}, a_i \in \mathbb{Z}_2^n$ ομοιόμορφα κατανεμημένο στο $\mathbb{Z}_2^n$, και $b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} \in \mathbb{Z}_2^m$, να βρεθεί ένα διάνυσμα $s \in \mathbb{Z}_2^n$, τέτοιο ώστε για μια τυχαία συναφείς εξίσωση m ισχύει συνεχόμενα:

$$\begin{cases} b_1 =_\varepsilon < a_1, s > (mod\ 2) \\ b_2 =_\varepsilon < a_2, s > (mod\ 2) \\ \qquad\qquad . \\ \qquad\qquad . \\ b_m =_\varepsilon < a_m, s > (mod\ 2) \end{cases}$$

, οπού $< a_m, s >$ αποτελεί το εσωτερικό γινόμενο δύο διανυσμάτων στο $\mathbb{Z}_2^n$.

Το LWE αποτελεί την γενίκευση του προβλήματος LPE από το mod 2 στη γενική περίπτωση mod q.

Αρχικά, ορίζουμε την τυχαία εξίσωση με σφάλμα στον ακέραιο δακτύλιο $\mathbb{Z}$. Έστω δύο θετική ακέραιοι $n \geq 1, q \geq 2$, $\mathbb{Z}_q$ το υποσύνολο mod q του δακτυλίου $\mathbb{Z}$ και χ η κατανομή πιθανότητας στο $\mathbb{Z}_q$.

Έστω $\alpha, b \in \mathbb{Z}, e \in \mathbb{Z}$, εάν

$$\Pr\{a \equiv b + e\ (mod\ q)\} = \chi(e) => a \equiv_\chi\ b + e\ (mod\ q), or\ a =_\chi\ b + e$$

Ο τύπος ονομάζεται εξίσωση τυχαίας σύμπτωσης με σφάλμα κάτω από χ, και μπορεί να γραφτεί μερικές φόρες και ως a = b + e. Με βάση την παραπάνω τυχαία εξίσωση συνάφειας δίνουμε τον ορισμό στο LWE πρόβλημα.

4.2.4 Module-LWE

Έστω ένα δειγματικό σύνολο $(\alpha_i, b_i = a_i s + e_i), i = 1, \dots., m$ (1) , οπού τα αᵢ επιλέγονται από την ομοιόμορφη κατανομή στο $\mathbb{R}_q^l$,το s είναι ένα στοιχείο από το $\chi_k(\mathbb{R}_q^l)$ και τα eᵢ είναι στοιχεία από το $\chi_k(\mathbb{R}_q)$. Το MLWE καθορίζεται ως μια διαδικασία διακρίσεις των στοιχείων που λαμβάνονται από την κατανομή (1) και στοιχείων που λαμβάνονται από την ομοιόμορφη κατανομή $\mathbb{R}_q \times \mathbb{R}_q$, όπου το πρόβλημα θα πρέπει να λυθεί σωστά με το πλεονέκτημα να είναι μη αμελητέο σε σύγκριση με μια τυχαία εικασία, δηλαδή το πλεονέκτημα θα πρέπει να είναι χαμηλότερο οριοθετημένο από μια συνάρτηση που δεν είναι αμελητέα στην παράμετρο ασφαλείας.

Το M-LWE πρόβλημα αποτελεί μια βασική υπολογιστική υπόθεση της LBC το οποίο παραθέτει μια ενδιαφέρουσα αντιστάθμιση μεταξύ εγγυημένης ασφάλειας και συγκεκριμένης απόδοσης. Το πρόβλημα παραμετροποιείται από μια μυστική κατανομή καθώς και από μια κατανομή σφάλματος, υπάρχει ένα κενό μεταξύ των επιλογών αυτών των κατανομών για θεωρητικά αποτελέσματα υψηλής ασφάλειας και την επιτυχή δημιουργία πρακτικών σχημάτων. Μια τυπική σύνθεση του M-LWE περιλαμβάνει ομοιόμορφο μυστικό modulo q και Gaussian error. Η ασφάλεια του MLWE στηρίζεται στην πραγματικότητα σε ένα πρόβλημα πλέγματος, δηλαδή στο SVP πρόβλημα. Για αυτό και θεωρητικά, η ανάκτηση ενός μηνύματος από ένα κρυπτογραφημένο κείμενο ή αντίστοιχα κάποιο ιδιωτικό από ένα δημόσιο κλειδί, θα πρέπει να είναι το ίδιο δύσκολη όσο και η επίλυση μιας μεγάλης παρουσίας SVP, ο υπολογισμός αυτός θα να πρέπει να είναι

αδύνατος , ακόμη και για έναν κβαντικό υπολογιστή, η αναγωγή του MLWE σε SVP έχει μια πολύ καλά θεωρητική τεκμηρίωση. Τέλος το LWE λειτουργεί με διανύσματα ακεραίων ενώ το M-LWE λειτουργεί με διανύσματα πολυωνύμων, τα διανύσματα πολυωνύμων είναι δύσκολα στην διαχείριση τους, ωστόσο προσφέρουν πολύ καλή ισορροπία μεταξύ ταχύτητας, μεγέθους και ασφάλειας.

4.2.5 NTRU

Το NTRU είναι ένα νέο κρυπτοσύστημα δημόσιου κλειδιού που βασίζεται σε δύσκολο πρόβλημα πλέγματος. Προτάθηκε από τους θεωρητικούς επιστήμονες υπολογιστών Hoffstein, Piper και Silverman το 1996. Στην ουσία ο κρυπτογραφικός σχεδιασμός του NTRU αποτελεί μια γενίκευση του RSA σε πολυώνυμα, γι' αυτό και σαν κρυπτοσύστημα βασίζεται σε πολυωνύμα δακτυλίων. Το κύριο χαρακτηριστικό του αποτελεί η απλή παραγωγή κλειδιών καθώς επίσης ο αλγόριθμος κρυπτογράφησης και αποκρυπτογράφησης είναι αρκετά πιο γρήγορος από τον συνήθως χρησιμοποιούμενο RSA και τα κρυπτοσυστήματα βασισμένα ελλειπτικής καμπύλης που έχουν αναλυθεί στο υποκεφάλαιο 2.5, πέρα από τα πλεονεκτήματα στην ταχύτατα ο NTRU ως κρυπτοσύστημα μπορεί να αντισταθεί σε επιθέσεις από ένα κβαντικό υπολογιστή για αυτό και θεωρείται ως ένας πιθανός αντικαταστήσεις του RSA για την κρυπτογραφία δημόσιου κλειδιού στη μετά κβαντική εποχή.

Το NTRU παράγει μυστικά το δημόσιο κλειδί έτσι ώστε η αποκρυπτογράφηση να είναι αποδοτική. Αρχικά ο αποστολέας παραλαμβάνει ένα μικρό διάνυσμα της μορφής (g,3f), στην συνέχεια χρησιμοποιεί τον Ευκλείδειο αλγόριθμο για να υπολογίσει μια τιμή h τέτοια ώστε το πλέγμα να εμπεριέχει το διάνυσμα. Το δημόσιο κλειδί είναι ένα πολυώνυμο βαθμού p:

$$h = h_0 + h_1 + \cdots + h_{p-1}x^{p-1}$$

με κάθε συντελεστή να ανήκει στο πεδίο {0,1,….,q-1}. Το κρυπτογραφημένο μήνυμα που δημιουργείται από το ιδιωτικό κλειδί είναι επίσης ένα πολυώνυμο c στο ίδιο πεδίο με το q. Για το ιδιωτικό κλειδί ο αποστολέας διαλέγει δύο πολυώνυμα d,c με μικρούς συντελεστές και υπολογίζει:

$$c = (h \times d) + e \; mod \; x^p - 1 \bmod q$$

Έστω ένα πλέγμα $\mathcal{L}\,(u, v)$ πολυωνυμικού βαθμού p με συντελεστές έτσι ώστε:

$$(h \times u) - v \; mod \; x^p - 1 \; mod \; q = 0$$

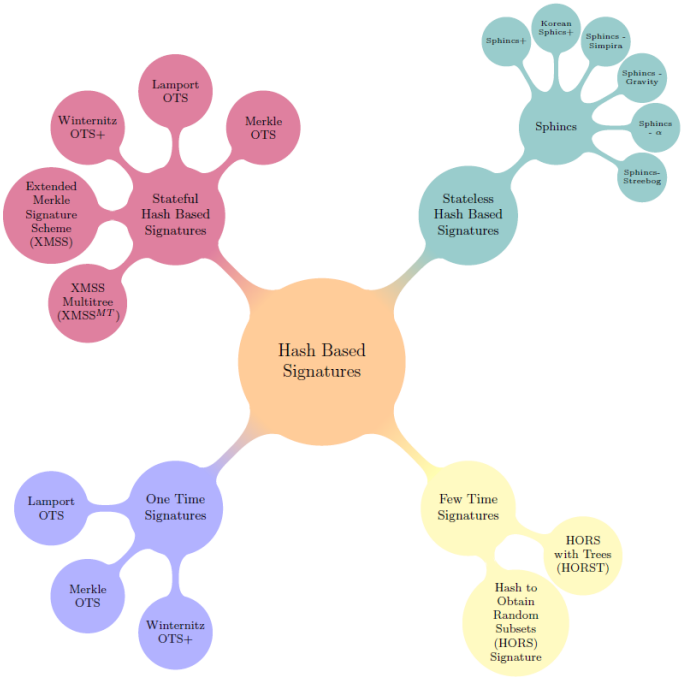
το $\mathcal{L}$, είναι ένα 2p διαστάσεων πλέγμα και εν περιέχει σημεία κοντά (d,c-e). Ένας επιτιθέμενος δεν μπορεί να βρει τα μυστικά κλειδιά d,e έχοντας ως δεδομένο το κρυπτογραφημένο μήνυμα c και το δημόσιο κλειδί h διότι είναι το ίδιο δύσκολο με το εντοπίσει κάποιος σε ένα πολυδιάστατο πλέγμα ένα σημείο πολύ κοντά σε ένα δεδομένο σημείο μέσα στο πλέγμα.

4.2.6 Πλεονεκτήματα και μειονεκτήματα των LBC αλγορίθμων

Αρχικά τα μαθηματικά προβλήματα που αφορούν τα lattice αποτελούν μερικά από τα πιο καλά, μελετημένα προβλήματα. Τα πλέγματα έχουν μελετηθεί από τους μαθηματικούς ειδή από τις αρχές του 1800 , αυτή η λεπτομέρεια είναι σημαντική διότι ένα μεγάλο μέρος από πιθανά όρια ή τρωτά σημεία που υπάρχουν είναι ήδη γνωστά και έχουν μελετηθεί εκ βάθους. Τα χρόνια μελέτης πάνω στα προβλήματα πλέγματος έχουν αποδείξει ότι είναι απίστευτα ευέλικτα όσον αφορά τους τύπους κρυπτογραφικών σχημάτων που μας επιτρέπουν να δημιουργήσουμε. Μπορούνε να αντικαταστήσουμε ουσιαστικά όλα τα ήδη υπάρχων τρωτών κρυπτογραφικών σχημάτων, αλλά επίσης επιτρέπουν και την δημιουργία εντελώς νέων κατηγόριων εξαιρετικά ισχυρών κρυπτογραφικών εργαλείων τα οποία είναι θωρακισμένα τόσο από κλασικούς όσο και από κβαντικούς υπολογιστές. Η σιγουριά που προσφέρουν τα προβλήματα πλέγματος στους κρυπταναλυτές αλλά και σε άλλους ερευνητές λόγου των χρονιών ερευνών πάνω σε αυτά σε συνδυασμό με την ευελιξία που διαθέτουν για τα κρυπτοσυστήματα που μπορούν να αναπτυχθούν πάνω σε αυτά έχουν οδήγηση τα LBC να αποτελούν τη μερίδα του λέοντος στις επιστημονικές δημοσιεύσεις που αφορούν τον τομέα της μετά κβαντικής κρυπτογραφίας. Ωστόσο υπάρχουν ακόμα κάποιες πιθανές επίθεσης κατά των διαφόρων LBC που εξετάζονται. Όπως, για παράδειγμα η cyclotomic δομή $x^p - 1$ που έχει χρησιμοποιηθεί ως μια επέκταση του αλγόριθμου του Shor για να σπάσει ορισμένα κρυπτοσυστήματα που βασίζονται σε πλέγμα. Επίσης επιθέσεις που λειτουργούν για αυθαίρετα πλέγματα, χωρίς να εκμεταλλεύονται οποιοδήποτε πολυωνυμική δομή και έχουν μικρότερους εκθέτες εξετάζονται για πιθανά κενά ασφάλειας που μπορούν να δημιουργήσουν. Τέλος δεν είναι πλήρως σαφές εάν η πρόσθετη δομή των υποκείμενων πλεγμάτων της μονάδας μπορεί να χρησιμοποιηθεί για τη διαμόρφωση καλύτερων επιθέσεων σε συγκεκριμένο στιγμιότυπο SVP.

4.3 Μετακβαντικά σχήματα υπογραφών βασισμένα σε κατακερματισμό (hash).

Οι αλγόριθμοι κρυπτογραφίας για υπογραφές οι οποίοι στηρίζονται σε κατακερματισμό παρέχουν έναν μηχανισμό υπογραφής που είναι καλά μελετημένος και ανθεκτικός σε επιθέσεις από κβαντικούς υπολογιστές εφόσον βέβαιος οι λειτουργίες κατακερματισμού είναι ασφαλείς δηλαδή χρησιμοποιείται μια ασφαλής συνάρτηση κατακερματισμού. Μετά κβαντικές ασφαλείς ψηφιακές υπογραφές προϋπήρχαν ήδη από τη δεκαετία του 1980, τα συστήματα αυτά βασίζονται σε τυπικές συναρτήσεις κατακερματισμού και δεν υπάρχει αμφιβολία σχετικά με την μετά κβαντική ασφάλειά τους. Το βασικό μειονέκτημα σε αυτά τα σχήματα κρυπτογραφίας το συναντάμε στην δημιουργία σχετικά μεγάλων υπογραφών, και επομένως περιορίζονται η εφαρμογή τους μόνο σε ορισμένες ρυθμίσεις. Μια τέτοια ρύθμιση είναι η υπογραφή ενός πακέτου λογισμικού ή μιας ενημέρωσης λογισμικού. Επειδή τα πακέτα λογισμικού τείνουν να είναι μεγάλα σε μέγεθος το πρόσθετο μήκος της υπογραφής δεν έχει μεγάλη σημασία. . Λόγου της υψηλής ασφάλειας από κλασικούς αλλά και κβαντικούς αλγορίθμους που προσφέρουν αυτά τα κρυπτοσυστήματα είναι πιθανό οι πωλητές λογισμικού να προχωρήσουν σε μετάβαση από τους αλγόριθμους ψηφιακής υπογραφής RSA και ελλειπτικής καμπύλης (ECDSA) σε υπογραφές που βασίζονται σε κατακερματισμό για υπογραφή λογισμικού.



Σχήμα 4-1: Ταξινόμηση κρυπτοσυστημάτων που βασίζονται σε κατακερματισμό

Πήγη: An Overview of Hash Based Signatures, Vikas Srivastava1⋆, Anubhab Baksi 2, and Sumit Kumar Debnath 1
1 National Institute of Technology Jamshedpur India, 2 Nanyang Technological University Singapore, URL: An Overview of Hash Based Signatures (iacr.org), Σελίδα 32

Όπως παρατηρούμε και στο Σχήμα 3-1 υπάρχουν πολυάριθμοι αλγόριθμοι κρυπτογραφίας βασισμένοι στον κατακερματισμό, ωστόσο τα διαθέσιμα πρότυπα για πρακτικά λειτουργικά κρυπτοσυστήματα αφορούν μόνο τις stateless εκδόσεις αυτών των αλγορίθμων, οι οποίες όμως παρουσιάζουν κάποια δύσκολα προβλήματα ανάπτυξης. Το SPHINCS+ είναι ένα από τα κρυπτοσυστήματα υπογραφών που βασίζεται σε stateless κατακερματισμό το οποίο έχει γίνει αποδεκτό από το NIST για τυποποιήσει ως αλγόριθμος υπογραφής γενικού σκοπού, περεταίρω ανάλυση του αλγόριθμου αυτού θα γίνει σε ερχόμενα κεφάλαια.

4.3.1 Stateless hash based signature

Τα συστήματα υπογραφών που βασίζονται σε stateless κατακερματισμό είναι συστήματα που δεν απαιτούν για την ενημέρωση του κλειδιού υπογραφής να πρέπει να γίνετε διαχείριση της κατάστασης του. Το κλειδί υπογραφής πλέον ορίζεται με κάποιο ψευδοτυχαία τρόπο κατά την υπογραφή ενός μηνύματος. Εφόσον τα σχήματα OTS εφαρμόζονται για την υπογραφή μηνυμάτων, απαιτούνται p^2 κλειδιά υπογραφής εφόσον πρόκειται να υπογράφουν p μηνύματα. Επομένως, σε συγκρίσει με τα stateful σχήματα υπογραφών η κλίμακα ολόκληρου του δέντρου είναι πολύ μεγαλύτερη από εκείνη των μηνυμάτων που πρέπει να υπογραφούν , ωστόσο τα κλειδιά δεν είναι απεριόριστα διότι υπάρχουν πληθώρα από βασικά ζητήματα που δημιουργούνται κατά το σχεδιασμό ενός συστήματα υπογραφών με αυτή την δυνατότητα.

Για να δημιουργηθεί ένα stateless σχήμα υπογραφών αρχικά χρειάζεται να χρησιμοποιηθεί ένας αποτελεσματικός αλγόριθμος για την επιλογή του κλειδιού. Το κλειδί μπορεί να επιλέγεται από το μια δεξαμενή κλειδιών υπογραφής τυχαία ή ψευδοτυχαία, σε αντίθεση με τα stateful σχήματα που το κλειδί υπογραφής επιλέγεται διαδοχικά από μια οντότητα που διαχειρίζεται όλα τα δυνατά κλειδιά υπογραφής. Πρακτικά η πλειοψηφία των stateless σχημάτων υπογραφής λαμβάνουν το μήνυμα προς υπογραφή ως μία είσοδο του αλγορίθμου PRG για την ψευδοτυχαία επιλογή του κλειδιού, ωστόσο, το υπέρ δέντρο συνεχίζει να παραμένει η πιο κοινή προσέγγιση για την δημιουργία απεριόριστου αριθμού και σχημάτων χωρίς κατάσταση. Για να για να επιτευχθεί αυτό χρειάζεται να δημιουργηθεί ένα ολόκληρο δέντρο σε αρκετά μεγαλύτερη κλίμακα για να αποφευχθεί η σύγκρουση μεταξύ κλειδιών δηλαδή ένα κλειδί να χρησιμοποιηθεί δύο φορές για την υπογραφή δύο διαφορετικών μηνυμάτων. Εφαρμόζοντας την τεχνική του υπέρ δέντρου σε υπογραφές που βασίζονται σε κατακερματισμό έχει ως αποτέλεσμα να υπολογιστεί μόνο ένα υπό δέντρο σε κάθε επίπεδο με αποτέλεσμα ο χρόνος δημιουργίας του κλειδιού να μειώνεται εκθετικά. Εκ του αποτελέσματος η τεχνική του υπέρ δέντρου συνεχίζει να παραμένει ένας εξαιρετικός εφαρμόσιμος τρόπος για την δημιουργία μεγάλου αριθμού κλειδιών σε stateless σχήματα υπογραφών παρόμοιο με το πλήθος κλειδιών που προσφέρουν τα stateful σχήματα υπογραφών. Είναι σημαντικό να διασφαλιστεί ότι δεν χρησιμοποιείται ποτέ ένα μόνο ζεύγος κλειδιών OTS για την υπογραφή δύο διαφορετικά μηνύματα, μια τεχνική που εφαρμόζεται για να υπογράφει ένα μήνυμα M χρησιμοποιώντας ένα δέντρο ύψους n ακολουθεί τα παρακάτω βήματα. Αρχικά υπολογίζεται ένα hash μεγέθους n-bit για το M. Το hash που προκύπτει είναι ένας ακέραιος h μεταξύ 0 και 2^n-1 στην συνέχεια χρησιμοποιείται το φύλλο στο δείκτη h για να υπογράφει το μήνυμα M. Στην πλήρη υπογραφή περιέχοντα όλα τα δημόσια κλειδιά OTS από το φύλλο h έως την ρίζα, καθώς επίσης και όλα τα δημόσια κλειδιά των αδελφών κόμβων που είναι προσαρτώμενα στο μονοπάτι αυτό, καθώς και τις εφάπαξ υπογραφές στο μήνυμα και στα δημόσια κλειδιά του μονοπατιού. Το OTS δημόσιο κλειδί της ρίζας γίνεται το συνολικό δημόσιο κλειδί, ενώ ως μυστικό κλειδί ορίζεται η τιμή που χρησιμοποιήθηκε για την ψευδοτυχαία δημιουργία όλων των άλλων ζευγών OTS κλειδιών του δέντρου. Τέλος μια εφαρμογή για stateless σχήμα υπογράφεις για να είναι εύχρηστη και αποτελεσματική θα πρέπει να εφαρμόζει διάφορες προσεγγίσεις για τη μείωση του κόστους αποθήκευσης και υπολογισμού των κλειδιών. Για παράδειγμα, στο κάτω μέρος ολόκληρου του δέντρου, τα σχήματα FTS χρησιμοποιούνται για την υπογραφή των μηνυμάτων αντί για την υπογραφή τους από τα σχήματα OTS, ενώ για την κατασκευή του υπό δέντρου σε κάθε στρώμα του εφαρμόζεται το PRG το οποίο δημιουργεί πιο αποδοτικά τους κόμβους κ.λπ. Επίσης για να μειωθούν οι απαιτήσεις σε πόρους που χρειάζονται για να τρέξει ο αλγόριθμος, η υπογραφή αποτελείται από το ευρετήριο του ιδιωτικού κλειδιού που χρησιμοποιείται για το FTS του μηνύματος, καθώς επίσης και τις υπογραφές αλλά και την αντίστοιχη διαδρομή ελέγχου ταυτότητας του δημόσιου κλειδιού της ρίζας σε κάθε βαθμίδα του δέντρου.

4.3.2 Πλεονεκτήματα και μειονέκτημα των stateless σχημάτων υπογραφείς

Για τα stateless σχήματα, χρειάζεται μια ενιαία δημιουργία κλειδιού OTS, η υπογραφή για να φτιαχτεί απαιτεί 2^n γενιές OTS κλειδιών και OTS υπογραφών. Αυτό μπορεί να επιτευχθεί σε εύλογο χρόνο για ασφαλείς παραμέτρους διότι τα κλειδιά είναι επίσης πολύ σύντομα. Η πολυπλοκότητα χρόνου για την δημιουργία ενός OTS δημόσιου κλειδιού είναι $O(n^2)$ και $O(n)$ για το μυστικό κλειδί. Παρόλα αυτά, το μέγεθος της υπογραφής είναι κυβικό ως προς την παράμετρο ασφαλείας. Για παράδειγμα, παίρνοντας n = 256 όπως κάνουμε για το SPHINCS-256 (SPHINCS$^+$) και εφαρμόζοντας κάποιες απλές βελτιστοποιήσεις, παράγεται μια υπογραφή το μέγεθος της οποίας εξακολουθεί να είναι πάνω από 1 MB. Παρόλο το σχετικά μικρό μέγεθος σε συγκρίσει με αλλά σχήματα υπογραφών μπορεί να προκαλέσει προβλήματα σε διάφορες εφαρμογές. Για παράδειγμα, στο λειτουργικό σύστημα Debian, το μέσο μέγεθος πακέτου είναι 1,2 MB και το διάμεσο μέγεθος πακέτου είναι μόλις 0,08 MB, το Debian έχει σχεδιαστεί έτσι ώστε να πραγματοποιεί συχνές ενημερώσεις, συνήθως αναβαθμίζοντας μόνο ένα πακέτο ή μερικά πακέτα κάθε φορά, και φυσικά κάθε αναβάθμιση πρέπει να ελέγχει τουλάχιστον μία νέα υπογραφή. Επίσης για παράδειγμα το μέγεθος μιας μέσης ιστοσελίδας είναι 1,8 MB και το HTTPS συνήθως στέλνει πολλές υπογραφές ανά σελίδα, ο ακριβής αριθμός είναι ανάλογος των ιστότοπων HTTPS που συνεργάζονται για την παραγωγή της σελίδας, πόσα πιστοποιητικά αποστέλλονται κ.λπ. Όπως παρατηρούμε ένα μέγεθος υπογραφής άνω του 1 MB θα υπερφόρτωση το δίκτυο με αποτέλεσμα να υπάρχουν ορατές καθυστερήσεις από τους χρήστες.

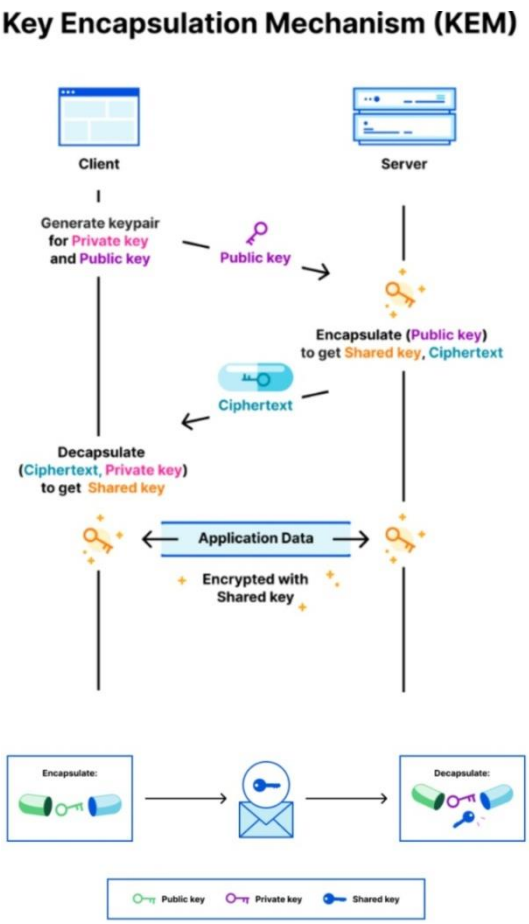
Αλλά σχήματα υπογράφεις έχουν ως δεδομένο έναν έμφυτο μηχανισμό για την παραγωγή και διαχείριση των κλειδιών που προορίζονται για τη δημιουργία της υπογραφής. Τα μετά κβαντικά σχήματα υπογραφών μέσα στα οποία εντάσσονται τα stateless σχήματα τα κλειδιά είναι αυτοδύναμα καθώς εφαρμόζουν την ίδια τη συνάρτηση κατακερματισμού για την ασφάλεια των δεδομένων και οι συναρτήσεις κατακερματισμού είναι ακόμη και εναλλάξιμες σε κάποιο βαθμό. Λόγου του μεγάλου ενδιαφέροντος που έχουν για πρακτικούς σκοπούς βελτιώνονται συνεχώς για παράδειγμα στο SPHINCS με την εισαγωγή του HORST, τα μεγέθη των κλειδιών είναι πολύ πιο ρεαλιστικά και αυτό εξαλείφει ένα σημαντικό μειονέκτημα των stateless σχημάτων υπογραφείς που αναλύσαμε προηγούμενος. Ωστόσο ένα από τα σημαντικότερα πλεονεκτήματος τους αποτελεί η αντοχή τους στους κβαντικούς υπολογιστές έναντι των κλασικών σχημάτων υπογράφεις που έχουν αναλυθεί στο Κεφάλαιο 2ο, πάρα τα προβλήματα που παρουσιάζουν τα σχήματα αυτά λόγο της ανάγκης για άμεση μεταβίβαση σε νέα πρότυπα κρυπτογραφίας βελτιώνονται συνέχεια με το SPHINCS$^+$ να προορίζεται άμεσα για αντικατάσταση των παλιών προτύπων. Βέβαια υπάρχουν και άλλοι αλγόριθμοι LBC για σφραγίδες όπως για παράδειγμα ο CRYSTALS-Dilithium και ο FALCON διότι το SPHINCS+ παράγει σχετικά μεγαλύτερες υπογραφές και πιο είναι αργός από τα άλλα δύο. Ωστόσο είναι πολύτιμος ως εφεδρικός, διότι βασίζεται σε μια διαφορετική μαθηματική προσέγγιση τις hash function σε αντίθεση με τις άλλες δύο επιλογές του NIST που βασίζονται μαθηματικά όπως προ είπαμε σε LBC προβλήματα.

4.4 CRYSTALS-Kyber

Το CRYSTALS-Kyber (Kyber) αποτελεί ένα PCQ σχήμα κρυπτογραφίας που επιλέχθηκε για τυποποίηση ως το πρώτο κβαντικό ανθεκτικό σχήμα PKE (τυποποιημένο ως KEM) από τον οργανισμό NIST(Πηγή: NIST Announces First Four Quantum-Resistant Cryptographic Algorithms | NIST). Είναι ένα LBC κρυπτοσύστημα διότι βασίζεται στο δύσκολο μαθηματικό πρόβλημα M-LWE, στον πυρήνα του, το Kyber είναι ουσιαστικά ένα LPR σύστημα κρυπτογράφησης μέσα σε μια ρύθμιση M-LWE. Το Kyber προορίζεται για γενικής χρήση ανταλλαγή κλειδιών σε εφαρμογές που απαιτείτε να δημιουργούν συχνά κλειδιά για λόγους ασφάλειας. Αυτό επιτυγχάνεται λόγου του μικρού μήκους σε σύγκριση με άλλους αλγόριθμους, που έχουν το ζευγάρι δημόσιου και ιδιωτικού κλειδιού καθώς επίσης και τις ταχύτητας του, μπορεί το μέγεθος των κλειδιών του να είναι μεγαλύτερο από αυτό των προ-κβαντικών σχημάτων κρυπτογραφίας αλλά είναι αρκετά μικρό για να μπορεί χρησιμοποιηθεί σε συστήματα που έχουν πραγματικές εφαρμογές. Τα αρχικά CRYSTALS σημαίνουν Cryptographic Suite for Algebraic Lattices, η σουίτα κρυπτοσυστημάτων περιλαμβάνει δύο αλγόριθμους τον Kyber καθώς επίσης και τον Dilithium για την δημιουργία ηλεκτρονικών υπογραφών.

4.4.1 Από το PKE στο KEM σχήμα

Όπως είδαμε στο πρώτο κεφάλαιο με τον RSA και το ECDH σε ένα PKE κρυπτογραφικό σύστημα, ο Bob χρησιμοποιεί το δημόσιο κλειδί της Alice για να κρυπτογραφήσει ένα μήνυμα m, το οποίο μόνο η Alice έχει την δυνατότητα να αποκρυπτογραφήσει για να λάβει πίσω το m, χρησιμοποιώντας το δικό της ιδιωτικό κλειδί. Δηλαδή ασύμμετρη κρυπτογράφηση: τα κλειδιά που χρησιμοποιούνται για την κρυπτογράφηση και την αποκρυπτογράφηση ενός μηνύματος είναι διαφορετικά. Αντίθετα, η KEM μέθοδο λειτουργεί διαφορετικά από μια ανταλλαγή κλειδιών, αν και έχει τον ίδιο σκοπό με την PKE δηλαδή να δημιουργήσει τελικά κάποιο κοινό μυστικό κλειδί ss μεταξύ των δύο μερών και να χρησιμοποιηθεί ως ένα κρυπτογραφικό πρωτόκολλο. Το KEM χρησιμοποιεί ασύμμετρη κρυπτογράφηση για να εξασφαλίσει την επικοινωνία μεταξύ της Alice και του Bob και στην συνέχεια μέσω του κρυπτογραφημένου μηνύματος να ανταλλάξουν το κλειδί (ενθυλάκωση),με αυτό τον τρόπο ένα κοινό τυχαία επιλεγμένο συμμετρικό κλειδί διαμοιράζεται μέσω του κρυπτογραφημένου μηνύματος από την Alice στον Bob το μήνυμα αυτό μπορεί να αποκρυπτογραφηθεί χρησιμοποιώντας το ιδιωτικό κλειδί. Σε ένα κλασικό KEM, η διαδικασία μπορεί να ξεκινήσει μέσω της ανταλλαγής ενός τυχαίου αριθμού ο οποίος αποστέλλεται από την Alice και τελικά θα κοινοποιηθεί με ασφάλεια στον Bob. Τέλος κάθε PKE με αρκετά μεγάλο χώρο απλού κειμένου μπορεί να μετατραπεί σε KEM καθώς επίσης κάθε KEM μπορεί να μετατραπεί σε PKE προσθέτοντας κάποια συμμετρική κρυπτογράφηση.



Σχήμα 4-2: σχηματική απεικόνιση του KEM κρυπτογραφικού συστήματος

Πηγή: Deep dive into a post-quantum key encapsulation algorithm (cloudflare.com)

4.4.2 Αλγόριθμος

Στο υποκεφάλαιο αυτό θα γίνει μια περιγράφει υψηλού επιπέδου για τον τρόπο λειτουργίας του αλγόριθμου τον οποίο βρίσκουμε στο επίσημο paper που κατατέθηκε στον NIST . Το paper μπορεί να βρεθεί στον παρακάτω σύνδεσμο https://pq-crystals.org/kyber/resources.shtml.

Στον αλγόριθμο Kyber το ιδιωτικό κλειδί **s** είναι ένα διάνυσμα πολυωνύμων, ενώ το δημόσιο κλειδί **t** είναι ένα διάνυσμα πολυωνύμων που υπολογίζεται από έναν πίνακα A που αποτελείται από τυχαία πολυώνυμα του ιδιωτικού κλειδιού **s** και κάποιο διάνυσμα για θόρυβο. Ο Kyber είναι ένας υβριδικός αλγόριθμος μεταξύ των σχημάτων PKE και KEM το οποίο προσπαθεί να κρυπτογραφήσει κάποιο μήνυμα **m** και να το μετατρέψει σε ένα κρυπτογραφημένο μήνυμα **ct**. Επίσης πρέπει για το KEM σχήμα, να πραγματοποιηθεί η ενθυλάκωση η όποια σαν διαδικασία παράγει ένα νέο κοινόχρηστο κλειδί **k** με ένα δημόσιο κλειδί το οποίο χρησιμοποιείται για να εξαχθεί το τελικό κοινόχρηστο κλειδί **K** και ένα κρυπτογραφημένο μήνυμα που αποστέλλεται στο άλλο μέρος της συνομιλίας. Τέλος για το PKE υπάρχει μια ακόμα περαιτέρω διαδικασία αποκρυπτογράφησης του **ct** μηνύματος για να λάβουμε πίσω το αρχικό μήνυμα **m**. Ενώ αντίστοιχα για το KEM, πραγματοποιείτε η αποκάλυψη από το κρυπτογραφημένο μήνυμα με ένα ιδιωτικό κλειδί του τελικό κοινόχρηστο κλειδιού **K**. Συνοψίζοντας το Kyber απαιτεί τρείς συναρτήσεις :

- Η πρώτη συνάρτηση χρειάζεται για να παράγει ένα ζεύγος κλειδιών.
- Η δεύτερη συνάρτηση δημιουργεί μια τυχαία τιμή **ss** και την κρυπτογραφεί σε ένα κρυπτογραφημένο μήνυμα **ct** με τη βοήθεια ενός δημόσιου κλειδιού **t**.
- Η τρίτη συνάρτηση αποκρυπτογραφεί το μήνυμα **ct** για να πάρει την τυχαία τιμή **ss** χρησιμοποιώντας το ιδιωτικό κλειδί **s**. Η συνάρτηση αυτή αποκαλύπτει και το τελικό κοινόχρηστο μυστικό κλειδί **K**.

Βήμα 1ο Δημιουργία κλειδιών

- Το ιδιωτικό κλειδί **s** στο ζεύγους κλειδιών για τον Kyber απαρτίζεται από πολυώνυμα με μικρούς συντελεστές, δηλαδή είναι ένα διάνυσμα «μικρών» πολυωνύμων.
- Το δημόσιο κλειδί στο Kyber αποτελείτε από δύο στοιχεία έναν πινάκα **A** αποτελούμενο από τυχαία πολυώνυμα και ένα διάνυσμα πολυνόμων **t.** Ωστόσο για να υπολογίσουμε το **t** χρειαζόμαστε ένα διάνυσμα **e** για να προσθέσουμε θόρυβο. Το διάνυσμα που χρησιμοποιείται για να δημιουργήσει θόρυβο αποτελείται από πολυώνυμα με μικρούς συντελεστές. Στην συνέχεια υπολογίζουμε το t μέσω του τύπου:

$$t = As + e$$

- Στο τέλος έχουμε το ζεύγος ιδιωτικού (**s**) και δημόσιου κλειδιού (**A** ,**t**). Η ασφάλεια του προβλήματος αυτού άγεται στο γεγονός ότι είναι δύσκολο κάποιος να υπολογίσει το **s** μέσω των **A** και **t**. Για να μπορέσει κάποιος να βρει το κλειδί θα πρέπει να λύσει το μαθηματικό πρόβλημα MLWE που είδαμε στο υποκεφάλαιο 4.2.3 .

Βήμα 2ο Κρυπτογράφηση

- Για την κρυπτογράφηση χρησιμοποιείται ένα πολυώνυμο $\mathbf{e_1}$ για να προσθέσει θόρυβο και ένα πολυωνυμικό διάνυσμα **r** για να πρόσθεση τυχαιότητα, τα πολυώνυμα αυτά δημιουργούνται τυχαία κάθε φορά που χρειάζεται να κρυπτογραφήσουμε ένα μήνυμα . Επίσης χρειαζόμαστε ένα ακόμα πολυώνυμο $\mathbf{e_2}$ για να προσθέσει και αυτό θόρυβο. Τα τρία αυτά πολυώνυμα είναι τυχαία μικρού μεγέθους.
- Στην συνέχεια για να κρυπτογραφήσουμε ένα μήνυμα, χρειάζεται να το μετατρέψουμε σε ένα πολυώνυμο. Αυτό επιτυγχάνεται αν εκμεταλλευτούμε την δυαδική αναπαράσταση του μηνύματος δηλαδή κάθε κομμάτι του μηνύματος χρησιμοποιείται ως συντελεστής. Για παράδειγμα έστω $m_b=(10)_{10}$ => $(1010)_2 \Leftrightarrow m_b=1x^3 + 0x^2 + 1x + 0$.
- Ωστόσο πριν κρυπτογραφήσουμε το μήνυμα m χρειάζεται να μεγαλώσουμε το μέγεθος του πολυωνύμου $\mathbf{m_b}$. Η διαδικασία αυτή επιτυγχάνεται πολλαπλασιάζοντας με το $\mathbf{m_b}$ με τον ποιο κοντινό ακέραιο από την διαίρεση q/2.

$$m = \left\lfloor \frac{q}{2} \right\rceil \times m_b$$

- Το m κρυπτογραφείτε μέσου του δημόσιου κλειδιού (**A,t**), για την διαδικασία της κρυπτογράφησης υπολογίζονται δύο τιμές ($\boldsymbol{u}, \boldsymbol{v}$) μέσω τον παρακάτω τύπων:

$$u = (A^T \times r) + e_1$$
$$v = (t^T \times r) + e_2 + m$$

- Το κρυπτογραφημένο μήνυμα που αποστέλλεται είναι οι τιμές $(\boldsymbol{u}, \boldsymbol{v})$.

Βήμα 3ο Αποκρυπτογράφηση

- Στην διαδικασία της αποκρυπτογράφησης ανακτούμε μέσω των τιμών $(\boldsymbol{u}, \boldsymbol{v})$ που στέλνονται ως κρυπτογραφημένο μήνυμα και του ιδιωτικού κλειδιού **s** το τελικό κοινόχρηστο κλειδί.
- Αρχικά απομακρύνουμε τον βουβό που έχουμε εισάγει στον μήνυμα εφαρμόζοντας την παρακάτω σχέση:

$$m_n = v - s^T u <=> m_n = e_1 r + e_2 + m + s^T e_1$$

- Αφού έχουμε απομακρύνει τον θόρυβο μπορούμε να ανακτήσουμε το αρχικό κλιμακωτό μήνυμα ελέγχοντας αν οι συντελεστές του $\mathbf{m_n}$ πολυωνύμου είναι πιο κοντά στο q/2 που σημαίνει πως ο αρχικός συντελεστής του $\mathbf{m_b}$ ήταν 1 η πιο κοντά στο 0 που σημαίνει πως αρχικός συντελεστής ήταν 0.
- Στην συνέχεια κλιμακώνοντας προς τα κάτω κατά 1/q το νέο στρογγυλοποιημένο πολυώνυμο προκύπτει το $\mathbf{m_b}$ μήνυμα που κρυπτογραφήθηκε.
- Τέλος διαβάζουμε τα bit και ανακτούμε το τελικό μήνυμα **m** που θέλαμε να στείλουμε

Οι παρακάτω μεταβλητές καθορίζουν και το μέγεθος των κλειδιών του Kyber:

	n	K	Q	n_1	n_2	d_u	d_v	Δ
Kyber$_{512}$	256	2	3329	3	2	10	4	2^{-139}
Kyber$_{768}$	256	3	3329	2	2	10	4	2^{-164}
Kyber$_{1024}$	256	4	3329	2	2	11	5	2^{-174}

- n: Ο μέγιστος αριθμός πολυωνύμων που χρησιμοποιούνται για τον αλγόριθμο.
- k: Αριθμός πολυωνύμων σε κάθε διάνυσμα
- q: Συντελεστής κλιμάκωσης παρανομαστών
- n_1 , n_2: Μεταβλητές που καθορίζουν ποσό μεγάλοι θα είναι οι συντελεστές για τα μικρά διανύσματα που χρησιμοποιούνται.
- d_u , d_v: Μεταβλητές που καθορίζουν πόσο θα συμπιεστούν οι τιμές (u, v).
- δ: Η πιθανότητα κατά την οποία το αποτέλεσμα της αποκρυπτογράφησης δεν είναι αυτό που είχε κρυπτογραφηθεί.

4.4.3 Παράδειγμα αλγόριθμου

Βήμα 1ο Δημιουργία κλειδιών

Ιδιωτικό κλειδί: **s**

$$\boldsymbol{s} = (-\boldsymbol{x}^3 - \boldsymbol{x}^2 + \boldsymbol{x}, -\boldsymbol{x}^3 - \boldsymbol{x})$$

Δημόσιο κλειδί: **(A,t)**

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{6x^3 + 16x^2 + 16x + 11} & \boldsymbol{9x^3 + 4x^2 + 6x + 3} \\ \boldsymbol{5x^3 + 3x^2 + 10x + 1} & \boldsymbol{6x^3 + x^2 + 9x + 15} \end{pmatrix}$$

Πριν υπολογίσουμε το **t** χρειαζόμαστε ένα διάνυσμα πολυωνύμων για θόρυβο έστω το διάνυσμα:

$$\boldsymbol{e} = (\boldsymbol{x}^2, \boldsymbol{x}^2 - \boldsymbol{x})$$

Στην συνέχεια υπολογίζουμε το **t**:

$$t = (A \times s) + e <=> \boldsymbol{t} = (\boldsymbol{16x^3 + 15x^2 + 7, 10x^3 + 12x^2 + 11x + 6})$$

Βήμα 2ο Κρυπτογράφηση

Πριν υπολογίσουμε το κρυπτογραφημένο μήνυμα χρειάζονται τρία διανύσματα πολυωνύμων για θόρυβο έστω τα διανύσματα:

$$\boldsymbol{r} = (-\boldsymbol{x}^3 + \boldsymbol{x}^2, \boldsymbol{x}^3 + \boldsymbol{x}^2 - \boldsymbol{1})$$
$$\boldsymbol{e_1} = (\boldsymbol{x}^2 + \boldsymbol{x}, \boldsymbol{x}^2)$$
$$\boldsymbol{e_2} = (-\boldsymbol{x}^3 - \boldsymbol{x}^2)$$

Στην συνέχεια μετατρέπουμε το μήνυμα που θέλουμε στην δυαδική του μορφή για να φτιάξουμε ένα πολυώνυμο. Το πολυώνυμο που προκύπτει το στρογγυλοποιούμαι κατά το ποιο κοντινό ακέραιο της πράξεις $\frac{q}{2}$. Έστω το μήνυμα 11 και q=17⇔$\frac{q}{2} = 9$:

$$(11)_{10} = (1011)_2 => \boldsymbol{m_b} = \boldsymbol{1x^3 + 0x^2 + 1x + 1}$$
$$m = \left\lfloor \frac{q}{2} \right\rceil \times m_b <=> 9 \times m_b <=> \boldsymbol{m_b} = \boldsymbol{9x^3 + 9x + 9}$$

Αφού έχουν οριστεί τα διανύσματα θορύβου και έχει υπολογιστή το διάνυσμα του μηνύματος που θέλουμε να κρυπτογραφήσουμε μπορούμε να προχωρήσουμε στην χρήση των δημόσιων κλειδιών για την κρυπτογράφηση. Το μήνυμα που αποστέλλεται αποτελείτε από τις παρακάτω δύο τιμές:

$$u = (A^T \times r) + e_1 <=> \boldsymbol{u} = (\boldsymbol{11x^3 + 11x^2 + 10x + 3, 4x^3 + 4x^2 + 13x + 11})$$
$$u = (t^T \times r) + e_2 + m <=> \boldsymbol{u} = (\boldsymbol{7x^3 + 6x^2 + 8x + 15})$$

Βήμα 3ο Αποκρυπτογράφηση

Πρώτα βρίσκουμε ένα μήνυμα το οποίο εν περιέχει θόρυβο.

$$m_n = v - (s^T \times u) <=> \boldsymbol{m_n = (7x^3 + 14x^2 + 7x + 5)}$$

Στην συνέχεια στρογγυλοποιούμε το m_n με το παρακάτω τρόπο:

- Αν ένας συντελεστής είναι πιο κοντά στο 9 γίνεται 9.
- Αν ένας συντελεστής είναι πιο κοντά στο 0 η μια μεταβλητή p το οποίο συμβολίζει ότι αριθμοί μεγαλύτερη του 9 και πιο κοντά στο p γίνονται πάλι 0. Έστω p=16.

Το νέο μήνυμα μετά την στρογγυλοποίηση γίνεται:

$$\boldsymbol{m_n = (9x^3 + 0x^2 + 7x + 1}$$

Στην συνέχεια κλιμακώνοντας προς τα κάτω το m_n κατά $\frac{1}{9}$ να παίρνουμε το αρχικό m_b μήνυμα:

$$m_b = \frac{1}{9}(9x^3 + 0x^2 + 9x + 9) = (1x^3 + 0x^2 + 1x + 1)$$

Έχοντας του τελικούς συντελεστές μπορούμε να ανακτήσουμε το τελικό κλειδί που έχει σταλεί το $(1011)_2 = (11)_{10}$.

*(Πηγή Kyber - How does it work? | Approachable Cryptography (cryptopedia.dev))

4.4.4 Τρωτά σημεία του Kyber

Στην επίσημη αναφορά που έχει υποβληθεί από την ομάδα ανάπτυξης του Kyber (Πηγή: https://pq-crystals.org/kyber/resources.shtml), αναλύουντο πως ο Kyber είναι θεωρητικά θωρακισμένος από κλασικές επιθέσεις που αναλυθήκαν στα υποκεφάλαια 1.2.2 , 1.3.2 και 1.5.3 αλλά και θωρακισμένος από τα προβλήματα που είδαμε στο υποκεφάλαιο 4.2.4 που θεωρητικά αντιμετωπίζουν οι LBC αλγόριθμοι. Αν εξαιρέσουμε κάποια μαθηματική ανάληψη στο μέλλον ή την ανακάλυψη ενός νέου κβαντικού αλγορίθμου, ο Kyber φαίνεται να είναι ένας καλά θωρακισμένος αλγόριθμος. Ωστόσο υπήρχε μια αναφορά για την δυνατότητα παραβιάσεις του αλγόριθμου με χρήσει Side-channel επιθέσεις υποβοηθούμενη από την χρήση μηχανικής μάθησης. Σύμφωνα με τα δύο paper από το KTH Royal Institute of, Stockholm, Sweden (1η Πηγή: Breaking a Fifth-Order Masked Implementation of CRYSTALS-Kyber by Copy-Paste (iacr.org), 2η Πηγή: https://eprint.iacr.org/2022/1692). Στα paper αυτά η ομάδα δοκίμασε επίθεση πλευρικής επίθεσης κατά την διαδικασία της ενθυλάκωσης οπού για κάθε bit όπως είδαμε και στα βήματα ανάπτυξης του αλγόριθμους στο υποκεφάλαιο 4.4.2 ο αλγόριθμος δημιουργεί μια μάσκα ανάλογα αν το bit είναι 1 ή 0. Όπως είναι λογικό η επεξεργασία του 0 και του 1 απαιτούν διαφορετική ενέργεια, οι συγγράφεις των paper βελτιστοποίησαν τις ειδή υπάρχων τεχνικές για την χρήση τεχνικής νοημοσύνης σε επίθεσης side channel. Εκπαιδεύοντας ένα μοντέλο επιβλεπόμενης μηχανικής μάθησης με κάποια είδη υπάρχων γνωστά ζευγάρια κλειδιών και μηνυμάτων το μοντέλο τους κατάφερε να φτιάξει ένα μοτίβο μεταξύ των εξομοιώσεων στην ενεργεία και των κλειδιών. Ωστόσο σύμφωνα με το παρακάτω άρθρο (Πηγή: No, AI did not break post-quantum cryptography (cloudflare.com)) , το οποίο έχει γραφτεί από δύο ειδικούς πάνω στην χρήση Al σε επιθέσεις side channel, οι συγγράφεις των paper κατάφεραν να ¨σπάσουν¨ εν μέρη των Kyber αλλά τα άρθρα στις εφημερίδες και τα blog που αναμετέδωσαν τα δεδομένα υπέρβαλαν με το

πραγματικό αντίτυπο που είχαν τα paper αυτά για την ασφάλεια του Kyber. Σύμφωνα με τους ερευνητές Δρ. Lejla Batina και Dr. Stjepan Picek οι συνθήκες πάνω στο οποίο έγινε το πείραμα ήταν κάτι παραπάνω από ιδεατές για να πετύχει αυτή η επίθεση. Αρχικά όπως επισημάνουν οι ερευνητές η εκπαίδευση του μοντέλου έγινε πάνω σε διάφορα κρυπτογραφημένα μηνύματα που είχαν κρυπτογραφηθεί με το ίδιο κλειδί το οποίο δεν είναι ρεαλιστικό σε ένα KEM διότι τα ζεύγη κλειδιών δημιουργούνται μόνο μια φόρα. Στην συνέχεια όπως παρατηρούν οι ερευνητές όλες οι επίθεσης και δοκιμές έγινα σε μια μονάδα οπού στο πραγματικό κόσμο θα ήταν άτοπο διότι ακόμα και δύο παρόμοια μοντέλα με το ίδιο hardware συνήθως έχουν αποκλίσεις στην διαχείριση της ενέργειας. Επίσης σύμφωνα με τους ερευνητές το hardware στο οποίο πραγματοποιήθηκαν οι δόκιμες ήταν αδύναμο και συναντάται για παράδειγμα σε έξυπνες κάρτες, στις οποίες γνωρίζουμε εξαρχής πως το κρυπτοσύστημα που τρέχουν και είναι βασισμένο στον αλγόριθμο Kyber έχει ρυθμιστεί για μέγιστη απόδοση και όχι μέγιστη ασφάλεια. Σύμφωνα με το παρακάτω άρθρο (Πηγή: Post-quantum algorithm vulnerable to side channel attacks, researchers | SC Media (scmagazine.com)) το οποίο παραθέτει την επίσημη γνώμη του NIST , δεν υπάρχει λόγος ανησυχίας ο NIST γνώριζε από το 2016 για την συγκεκριμένη αδυναμία του αλγόριθμου και τα τελικά κρυπτοσυστήματα για τυποποίηση έχουν ρυθμιστή και για αυτή την επίθεση.

4.5 CRYSTALS-Dilithium

Το CRYSTALS-Dilithium (Kyber) αποτελεί ένα PCQ σχήμα κρυπτογραφίας που επιλέχθηκε για τυποποίηση και προτείνεται από τον οργανισμό NIST ως κύριος αλγόριθμος για την δημιουργία ηλεκτρονικών υπογραφών. Όπως και το Kyber που αναλύσαμε στο υποκεφάλαιο 4.4 ανήκει στην σουίτα κρυπτοσυστημάτων CRYSTALS , επίσης είναι ένα LBC κρυπτοσύστημα διότι η ασφάλεια του αλγορίθμου βασίζεται στην επίλυση των δύσκολων μαθηματικών προβλημάτων SIS και M-LWE, η κατασκευή του βασίζεται στην τεχνική Fiat-Shamir με απολαβές. Οι αλγόριθμοι ηλεκτρονικών υπογραφών που βασίζονται σε πλέγματα είναι σχετικά γρήγοροι διότι ο πολλαπλασιασμός μεταξύ ενός πίνακα και ενός διανύσματος αποτελεί μια σχετικά γρήγορη υπολογιστική πράξη. Ωστόσο τα κλειδιά και οι υπογραφές είναι πολύ μεγάλες για παράδειγμα ο NIST για την μεγίστη ασφάλεια που μπορεί να προσφέρει ο Dilithium προτείνει κλειδιά μεγέθους 1427 bytes και υπογραφή μεγέθους 2701 bytes. Το μεγάλο μέγεθος των κλειδιών οφείλεται στο διάνυσμα μεγέθους m στοιχείων από το οποίο αποτελείται κάθε κλειδί, ενώ οι υπογραφές αποτελούνται από τα $m \times k$ στοιχεία. Παρόλα αυτά μέσω διαφόρων παραμετροποιήσεων ο Dilithium μπορεί να προσαρμοστεί σε διάφορες εφαρμογές, διότι για παράδειγμα μπορεί να περιοριστεί η ασφάλεια που προσφέρει αλλά να αυξηθεί η απόδοση του και να μειωθούν τα μεγέθη των κλειδιών και των υπογραφών τα διάφορα επίπεδα ασφάλειας που προσφέρει ο αλγόριθμος παρουσιάζονται στην Εικόνα 4-3.

NIST Security Level	2	3	5
Output Size			
public key size (bytes)	1312	1952	2592
signature size (bytes)	2420	3293	4595
LWE Hardness (Core-SVP and refined)			
BKZ block-size b (GSA)	423	624	863
Classical Core-SVP	123	182	252
Quantum Core-SVP	112	165	229
BKZ block-size b (simulation)	433	638	883
$\log_2$ Classical Gates (see App. C.5)	159	217	285
$\log_2$ Classical Memory (see App. C.5)	98	139	187
SIS Hardness (Core-SVP)			
BKZ block-size b	423 (417)	638 (602)	909 (868)
Classical Core-SVP	123 (121)	186 (176)	265 (253)
Quantum Core-SVP	112 (110)	169 (159)	241 (230)
Performance (Unoptimized Reference Code, Skylake)			
Gen median cycles	300,751	544,232	819,475
Sign median cycles	1,081,174	1,713,783	2,383,399
Sign average cycles	1,355,434	2,348,703	2,856,803
Verify median cycles	327,362	522,267	871,609
Performance (AVX2, Skylake)			
Gen median cycles	124,031	256,403	298,050
Sign median cycles	259,172	428,587	538,986
Sign average cycles	333,013	529,106	642,192
Verify median cycles	118,412	179,424	279,936
Performance (AVX2+AES, Skylake)			
Gen median cycles	70,548	153,856	153,936
Sign median cycles	194,892	296,201	344,578
Sign average cycles	251,144	366,470	418,157
Verify median cycles	72,633	102,396	151,066

Εικόνα 4-3: Παράμετροι που καθορίζουν το επίπεδο ασφάλειας στον Dilithium

Πήγη: CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation, October 1 2020, Shi Bai, Léo Ducas, Eik Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler and Damien Stehlé, URL: Dilithium – Resources (pq-crystals.org), Σελίδα 8

4.5.1 Fiat-Shamir

Τα περισσότερα πρωτόκολλα μηδενικής γνώσης (zero-knowledge proofs) ακολουθούν τα παρακάτω τρία βήματα:

- Ο χρήστης που θέλει να αποδείξει την ταυτότητα του(prover) , δεσμεύεται μέσω κάποιας τυχαίας τιμής που στέλνει στον χρήστη που θέλει να επαλήθευση την ταυτότητα του(verifier).
- Ο verifier απαντάει μέσω μιας ερώτησης προκλήσεις βασισμένη στην τιμή αυτή.
- Στο τέλος ο prover πρέπει να απαντήσει σωστά στην ερώτηση αυτή δεδομένου της τιμής που έστειλε.

Για παράδειγμα έστω ο Α (prover) που θέλει να αποδείξει στον Β (verifier) ότι όντως είναι ο Α. Στο παρακάτω υποθετικό παράδειγμα ο Α βρίσκεται σε ένα δωμάτιο με τρείς διακόπτες που ανάβουν εκάστοτε ο καθένας από ένα φως σε ένα άλλο δωμάτιο που βρίσκεται ο Β. Αν ο Α ανάψεις το τρίτο φως και ο Β τον ρωτήσει πιο φως είναι αναμμένο, ο Α θα πρέπει να μπορεί να του απαντήσει σωστά αφού αυτός γύρισε τον διακόπτη.

Η παραπάνω δομή απαιτεί μια απάντηση από τον verifier προτού μπορέσει να ολοκληρωθεί η απόδειξή της ταυτότητας του χρήστη, η αναμονή για την απάντηση δεν είναι ιδανική για τις περισσότερες εφαρμογές κρυπτογράφησης. Ωστόσο το 1986 οι Uriel Feige, Amos Fiat, και Adi Shamir πρότειναν μια γενική μέθοδο/μετατροπή στην οποία, αντί να στέλνει ο verifier μια τυχαία τιμή πρόκλησης στον prover, ο prover θα είναι σε θέση να μπορεί να υπολογίσει μόνος του τιμή/πρόκληση χρησιμοποιώντας μια τυχαία συνάρτηση, όπως για παράδειγμα μια κρυπτογραφική συνάρτηση κατακερματισμού. Ο παραπάνω μετασχηματισμός μπορεί να φαίνεται απλός, αλλά η σωστή και πρακτική εφαρμογή τείνει να είναι πολύ δύσκολη στην πράξη. Διότι ο prover παράγει την τιμή τυχαίας πρόκλησης χρησιμοποιώντας μια κρυπτογραφική συνάρτηση κατακερματισμού , ωστόσο ποιες θα είναι οι είσοδοι της συνάρτησης, αποδεικνύεται ότι εάν επιλέγουν λάθος εισόδους, συνήθως σημαίνει ότι το σύστημα δεν λειτουργεί σωστά. Το κάθε σύστημα μηδενικής γνώσης απαιτεί και διαφορετικές εισόδους, ωστόσο ο εμπειρικός κανόνας προτείνει να συμπεριλαμβάνονται όλες οι δημόσιες πληροφορίες και όλα τα στοιχεία μέσα στη συνάρτηση κατακερματισμού μέχρι το σημείο της διαδικασία επαλήθευση της ταυτότητας.

4.5.2 Αλγόριθμος

Για τον Dilithium αρχικά πρέπει να δημιουργηθεί ένας πινάκας **A** μεγέθους $k \times l$, οπού κάθε στοιχείο του πίνακα είναι ένα πολυώνυμο που ανήκει σε δακτύλιο πολυωνύμων R. Στην συνέχεια δημιουργούνται δύο ιδιωτικά διανύσματα **s1** και **s2** των οποία τα στοιχεία ανήκουν επίσης στα στοιχεία του R δακτυλίου. Το δημόσιο κλειδί για τον αλγόριθμο είναι ο πίνακας **A** και το $\boldsymbol{t} = (\boldsymbol{A} \times \boldsymbol{s1}) + \boldsymbol{s2}$. Αφού έχουν δημιουργηθεί τα δημόσια κλειδιά ο αλγόριθμος εκτελεί τα παρακάτω βήματα:

- Αρχικά ο prover θέλει να αποδείξει ότι γνωρίζει το μυστικό κλειδί (να αποδείξει την ταυτότητα του). Για αυτό και δημιουργεί ένα τυχαίο μυστικό **y**, στην συνέχεια υπολογίζει $\boldsymbol{A} \times \boldsymbol{y}$ και στην συνέχεια θέτει ως δεσμευτική τιμή **w1**,τα υψηλότερης τάξης bits των συντελεστών του διανύσματος που θα προκύψει από τον προηγούμενο πολλαπλασιασμό.
- Ο verifier αποδέχεται την δέσμευση **w1** και δημιουργεί μια πρόκληση **c**, για την δέσμευση αυτή.
- Στην συνέχεια ο prover δημιουργεί μια εν δύναμη υπογράφει $\boldsymbol{z} = \boldsymbol{y} + (\boldsymbol{c} \times \boldsymbol{s1})$ και παράλληλα πραγματοποιεί ελέγχους στα μεγέθη διάφορων παραμέτρων που ρυθμίζουν την ασφάλεια της υπογραφής.
- Τέλος ο verifier παραλαμβάνει την υπογραφή και υπολογίζει το **w1** ως τα υψηλότερης τάξης bits των συντελεστών του διανύσματος $(\boldsymbol{A} \times \boldsymbol{z}) - (\boldsymbol{c} \times \boldsymbol{t})$**.** Η απάντηση γίνεται δέκτη δηλαδή αποδεικνύετε η ταυτότητα του prover αν το **w1** είναι ίσο με το **w0** και όλοι οι συντελεστές του **z** έχουν μικρότερη τιμή από την παράμετρο ασφαλείας.

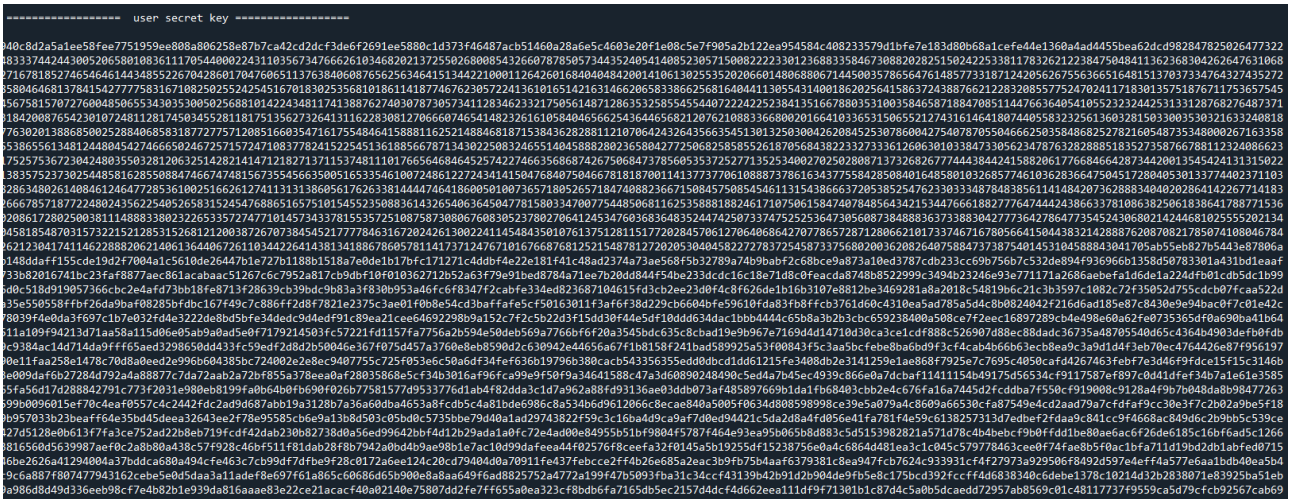
Χρησιμοποιώντας τον μετασχηματισμό Fiat–Shamir: αντί ο verifier να αποδεχθεί τη δέσμευση και να στείλει μια πρόκληση c, ο prover υπολογίζει την πρόκληση ως ένα κατακερματισμό $\boldsymbol{H}(\boldsymbol{M} \| \boldsymbol{w1})$ της τιμής **w1** και κάποιου μηνύματος **M.** Με την παρακάτω προσέγγιση ο χρήστης που έχει υπογράψει το μήνυμα **M** έχει δημιουργήσει ένα πρόβλημα βασισμένο στην θεωρεία πλεγμάτων οπού μόνο εκείνος ξέρεις την λύση. Πρακτικά η μετατροπή αυτή μετατρέπει τον Dilithium από ένα σχήμα μηδενικής γνώσης σε ένα κλασικό σχήμα ηλεκτρονικής υπογραφής , οπού το μήνυμα έχει υπογράφει από ένα ιδιωτικό κλειδί και όσοι έχουν πρόσβαση στο δημόσιο κλειδί μπορούν να επιβεβαιώσουν τον χρήστη του ιδιωτικού κλειδιού.

4.5.3 Παράδειγμα εκτέλεσης του Dilithium

Στο παρακάτω υποκεφάλαιο θα κάνουμε κάποιες πειραματικές εκτελέσεις του αλγόριθμου μέσω της βιβλιοθήκης ανοικτού κώδικα η οποία μπορεί να βρεθεί στο παρακάτω σύνδεσμο (dilithium-py/README.md at main · GiacomoPope/dilithium-py · GitHub), στην παρακάτω βιβλιοθήκη έχει αναπτυχθεί ο Dilithium σε γλώσσα Python. Έχοντας ως βάση την παρακάτω βιβλιοθήκη θα υποθέσουμε πως έχουμε μια εφαρμογή η οποία πιστοποιεί την είσοδο ενός χρήστη σε κάποια εφαρμογή διαχείρισης τραπεζικού λογαριασμού. Για την παρακάτω υλοποίηση έχουν γίνει δύο υποθέσεις: Πρώτον έχει επιλέγει ο Dilithium τρίτου επιπέδου ασφάλειας σύμφωνα με τα πρότυπα του NIST που αναλύθηκαν σε προηγούμενο υποκεφάλαιο, διότι προσφέρει μια μέση λύση μεταξύ ασφάλειας και αποδοτικότητας του αλγόριθμου. Δεύτερον ο υποτιθέμενος οργανισμός που κάνει επαλήθευση τον χρήστη δεν έχει κάπου αποθηκευμένα τον όνομα και τον κωδικό του χρήστη αλλά ένα hash που περιέχει αυτά τα στοιχεία και το δημόσιο κλειδί του χρήστη.

Όλα τα πειράματα θα εκτελεστούν σε μηχάνημα με 8Gb ram, intel core i3-10 gen, σε γλώσσα python 3.8 και windows 10 λειτουργικό σύστημα.

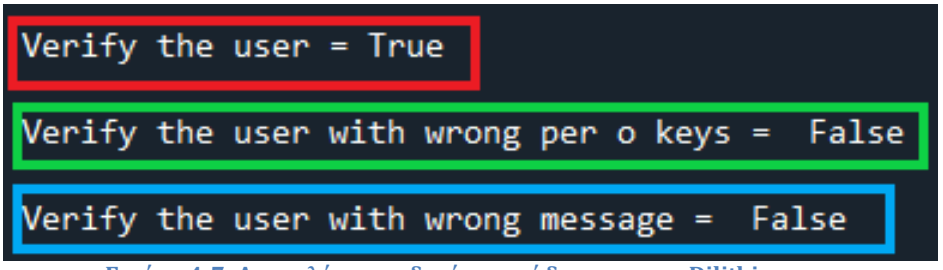
Εικόνα 4-4: Δημόσιο κλειδί για τον Delithium σε δεκαεξαδική μορφή



Εικόνα 4-5: Ιδιωτικό κλειδί για τον Dilithium σε δεκαεξαδική μορφή

Εικόνα 4-6: Dilithium υπογραφή σε δεκαεξαδική μορφή

Όπως παρατηρούμε και από τις Εικόνες 4-4, 4-5 και 4-6 τα κλείδα αλλά και το μήνυμα που παράγεται μετά το κατακερματισμό του αρχικού μικρού μηνύματος είναι πολύ μεγάλα ακόμα και σε αναπαράσταση δεκαεξαδικής μορφής.



Εικόνα 4-7: Αποτελέσματα δοκίμων κώδικα για τον Dilithium

Στην Εικόνα 4-7 , εμφανίζονται τα αποτελέσματα δοκίμων για διάφορες εκδοχές του αλγόριθμου. Στο κόκκινο ορθογώνιο απεικονίζεται η ιδεατή περίπτωση οπού το μήνυμα έχει υπογραφεί από τον χρήστη και γίνετε επιβεβαίωση του ιδίου μηνύματος. Στο πράσινο ορθογώνιο το μήνυμα είναι το ίδιο αλλά έχει υπογράφει από άλλο ιδιωτικό κλειδί οπότε δεν είναι δυνατή η επιβεβαίωση του χρήστη που επιθυμούμαι. Στο μπλε ορθογώνιο ο χρήστης που επιθυμούμαι έχει υπογράψει το μήνυμα, ωστόσο οι κωδικοί που έχει δώσει δεν είναι σωστή οπότε το hash που έχει αποθήκευση η τράπεζα και αυτό που παρήγαγε ο χρήστης δεν είναι ιδία.

4.5.4 Τρωτά σημεία του Dilithium

Σύμφωνα με το επίσημο έγγραφο (URL: Dilithium – Resources (pq-crystals.org)) που έχει κατατεθεί στον NIST για τον τρίτο γύρω έλεγχου των υποψήφιων PQC για προτυποποίηση, οι δημιουργεί του Dilithium ισχυρίζονται ότι προηγούμενα κενά ασφάλειας που είχε ο αλγόριθμος στους πρώτους δύο γύρους έχουν διορθωθεί. Ωστόσο την περίοδο που ολοκληρώθηκε ο τρίτος γύρος αξιολόγησης των αλγόριθμων εκδοθήκαν papers με πιθανές επίθεσης εναντία του Dilithium. Ομάδα ερευνητών σύμφωνα με την παρακάτω ερεύνα (URL: 2203.00637.pdf (arxiv.org)) στην οποία εφαρμόστηκε μια επίθεση γνωστή ως Signature Correction, κατάφερε χρησιμοποιώντας ελαττωματικές υπογραφές και το δημόσιο κλειδί να ανακτήσουν κάποια bits από το μυστικό κλειδί , συγκριμένα κατάφεραν να ανακτήσουν 1.851 bits από τα 3.072, μπορεί να

μην βρήκαν όλα τα bits ωστόσο κατάφεραν να υποβαθμίσουν την αρχική ασφάλεια που παρέχει το επιλεγμένο κλειδί. Μια άλλη ομάδα συμφώνα με την παρακάτω ερεύνα (URL: Profiling Side-Channel Attacks on Dilithium (iacr.org)) χρησιμοποίησαν μια επίθεση πλευρικού καναλιού, κατά την οποία, η ομάδα των ερευνητών εκμεταλλευόμενη μια μικρή διαρροή σε κάποια από τα bits του μυστικού κλειδιού εφάρμοσαν μηχανική μάθηση συνδυαστικά με διάφορους αλγορίθμους όπως γραμμικό ακέραιο προγραμματισμό και παλινδρόμηση ελάχιστων τετραγώνων, κατάφεραν να βρουν το τελικό μυστικό κλειδί. Ωστόσο έως τώρα που γράφεται η παρακάτω διπλωματική εργασία δεν υπάρχει κάποια απάντηση από τους δημιουργούς του αλγόριθμου για αυτές τις επιθέσεις, επίσης ούτε ο NIST έχει εκδώσει κάποια ανακοίνωση για πιθανά κενά ασφάλειας που μπορεί να έχει ο Dilithium.

4.6 FALCON

Ο αλγόριθμος Falcon είναι ένα LBC, το οποίο προορίζεται για την δημιουργία ηλεκτρονικών υπογραφών, βασίζεται στα δύσκολα μαθηματικό πρόβλημα SIS(Υποκεφάλαιο 4.2.3) πάνω από πλέγματα NTRU(υποκεφάλαιο 4.2.5). Σε συγκρίσει με αλλά QPC που προορίζονται για την δημιουργία ηλεκτρονικών υπογραφών, ο Falcon έχει σχεδιαστεί με γνώμονα να προσφέρει την ανώτερη προστασία χωρίς ωστόσο να θυσιάζει αλλά σημεία που μπορούν να επηρεάσουν την απόδοση του. Για αυτό και παρέχει τις μικρότερες υπογραφές και το μικρότερο συνδυασμένο μεγέθους υπογραφής και δημόσιου κλειδιού με αποτέλεσμα να είναι γρήγορος αλλά και οικονομικός ως προς την κατανάλωση πόρων αφού απαιτεί τον μικρότερο μέγεθος μνήμης για να τρέξει. Ο Falcon αποτελείτε ουσιαστικά από τα παρακάτω τρία στοιχεία:

$$Falcon = GPV\ framework + NTRU\ lattices + Fast\ Fourier\ sampling$$

4.6.1 GPV framework

Το GPV framework περιγραφή μια μεθοδολογία κατασκευής ασφαλών υπογραφών από ένα ζευγάρι ιδιωτικού και δημόσιου κλειδιού τα οποία αναπαριστώνται ως βάσεις για ένα χώρο πλέγματος σε μορφή μήτρας. Στόχος του framework αυτού είναι η δημιουργία σύντομων υπογραφών χωρίς ωστόσο αυτή η ταχύτητα να επηρεάζει την ασφάλεια που προσφέρει ο αλγόριθμος. Αρχικά το GPV χρειάζεται την επιλογή κάποιας κατηγορίας πλεγμάτων που διαθέτουν ισχυρά κρυπτογραφικά στοιχεία. Μέσω των πλεγμάτων αυτών θα πρέπει να δημιουργηθεί μια μικρή και μια μεγάλη βάση για το ίδιο πλέγμα, με τέτοιο τρόπο ώστε να είναι δύσκολο για οποιονδήποτε με πρόσβαση στην μεγάλη βάση να βρει κοντινά διανύσματα με την ιδία ακρίβεια που θα μπορούσε αν είχε πρόσβαση στην μικρή βάση. Τα LBC κρυπτοσυστήματα απαιτούν συνήθως διαστάσεις της τάξης μεγέθους n = 1024 για να είναι ασφαλή έναντι κλασικών και κβαντικών υπολογιστών, ωστόσο η αποθήκευση βάσεων τόσο υψηλών διαστάσεων είναι αρκετά κοστοβόρα, διότι το δημόσιο κλειδί που προκύπτει μπορεί εύκολα να έχει μέγεθος ενός megabyte. Το GPV προσφέρει μια λύση στο πρόβλημα αυτό , διότι σύμφωνα με το framework χρησιμοποιούνται δομημένα πλέγματα, όπου μια ολόκληρη βάση μπορεί να ληφθεί περιστρέφοντας τους συντελεστές μερικών από τα αρχικά διανύσματα της βάσης, ο Falcon χρησιμοποιεί πλέγματα NTRU τα οποία αποτελούν μια κατηγορία τέτοιων δομημένων πλεγμάτων. Η χρήση αυτών των πλεγμάτων έχει ως αποτέλεσμα την μείωση του μεγέθους του δημόσιου κλειδιού σε λιγότερο από 1,8 kilobyte. Στην συνέχεια το framework χρειάζεται έναν αλγόριθμος για τον υπολογισμό των στενών διανυσμάτων στο πλέγμα, ο αλγόριθμος αυτός ονομάζεται Fast Fourier Sampling το οποίο αποτελεί ένα υβρίδιο μεταξύ της δειγματοληψίας GPV και του γρήγορου μετασχηματισμού Fourier.

4.6.2 Αλγόριθμος

Για τον αλγόριθμο Falcon εργαζόμαστε σε έναν δακτύλιο $\mathcal{R} = \frac{\mathbb{Z}_q[x]}{x^n+1}$

Δημιουργία κλειδιού:

- Δημιουργούνται δύο πλέγματα $\boldsymbol{\mathcal{L}(A)}$ και $\boldsymbol{\mathcal{L}(B)}$ με συντελεστές που ανήκουν στο $\mathcal{R}$, έτσι ώστε $(A \times B) = 0$ και το B εν περιέχει μικρούς συντελεστές. Ο πίνακας **A** είναι το δημόσιο κλειδί και ο πίνακας **B** είναι το ιδιωτικό κλειδί.

Υπογραφεί ενός μηνύματος m με το ιδιωτικό κλειδί A

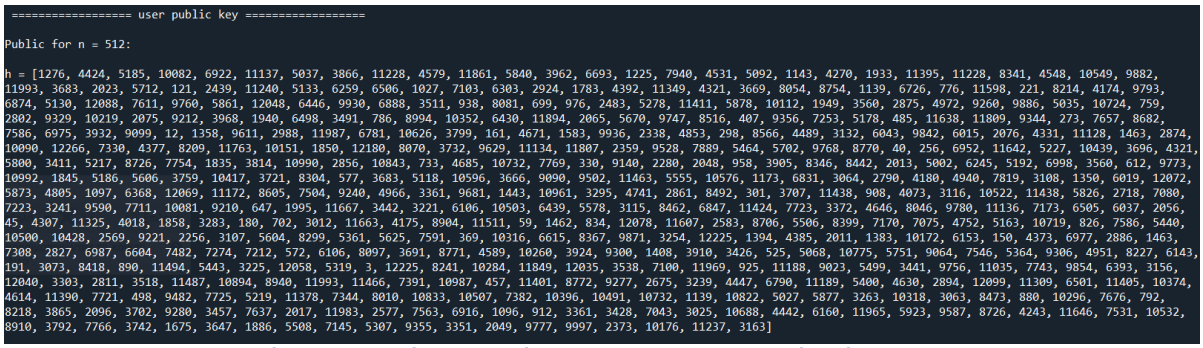
- Υπολογίζουμε το σημείο **c** τέτοιο ώστε: $c \times A = H(m)$, όπου **H** είναι μια συνάρτηση κατακερματισμού που στέλνει μια είσοδο σε ένα τυχαίο σημείο στο πλέγμα.
- Υπολογίζουμε ένα σημείο **v** το όποιο είναι ένα σημείο στο $\boldsymbol{\mathcal{L}(B)}$ κοντά στο σημείο **c**. Για το υπολογισμό του **c** μπορεί να χρησιμοποιηθεί ο Fast Fourier Sampling.
- Υπολογίζουμε την υπογραφεί **s** με τον τύπο: $s = c - v$

Επαλήθευση υπογραφής s μέσω του δημόσιου κλειδιού B

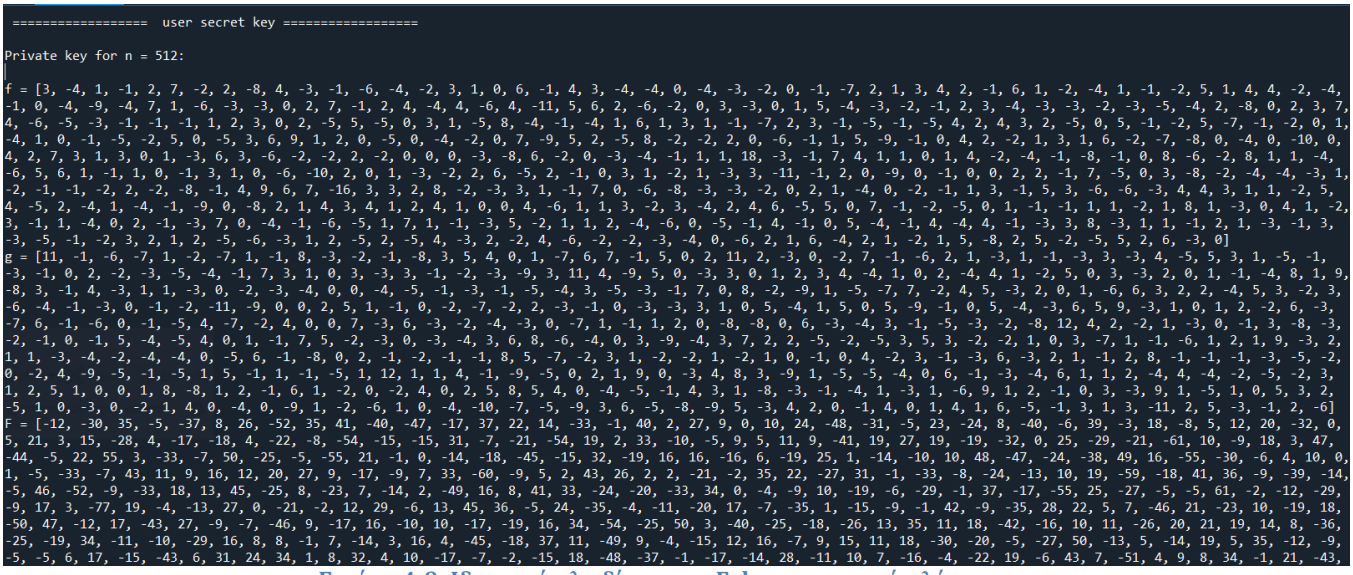
- Η υπογραφή γίνεται αποδεκτή αν το **s** δεν έχει μεγάλο μέγεθος.
- Επίσης ισχύει ο τύπος: $(s \times A) = H(m)$

4.6.3 Παράδειγμα εκτέλεσης του Falcon

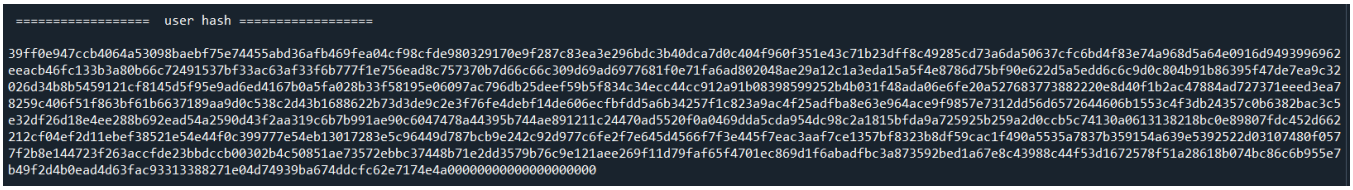
Στο παρακάτω υποκεφάλαιο θα κάνουμε κάποιες πειραματικές εκτελέσεις αλγόριθμου, μέσω της βιβλιοθήκης ανοικτού κώδικα η οποία μπορεί να βρεθεί στο παρακάτω σύνδεσμο (GitHub - tprest/falcon.py: A python implementation of the signature scheme Falcon), στην παρακάτω βιβλιοθήκη έχει αναπτυχθεί ο Falcon σε γλώσσα Python. Έχοντας ως βάση την παρακάτω βιβλιοθήκη θα υποθέσουμε πως έχουμε την ιδία εφαρμογή με το υποκεφάλαιο 4.5.3.



Εικόνα 4-8: Δημόσιο κλειδί για τον Falcon σε μορφή πλέγματος

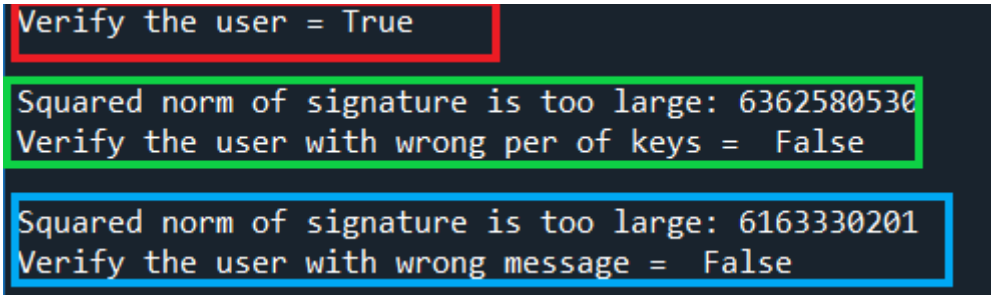


Εικόνα 4-9: Ιδιωτικό κλειδί για τον Falcon σε μορφή πλέγματος



Εικόνα 4-10: Falcon υπογραφή σε δεκαεξαδική μορφή

Εξετάζοντας τις Εικόνες 4-4,4-5,4-8 και 4-9 παρατηρούμε πως και ο Dilithium και Falcon έχουν μικρά δημόσια κλειδιά και μεγάλα ιδιωτικά. Ωστόσο ενδιαφέρον παρουσιάζει η οπτική σύγκριση στο μέγεθος των κλειδιών για το ίδιο επίπεδο ασφαλείας σύμφωνα με τα πρότυπα του NIST. Τα κλειδιά που παράγει ο Falcon όπως απεικονίζεται και στις Εικόνες 4-8(δημόσιο Falcon) και 4-9 (ιδιωτικό Falcon) είναι αρκετά πιο μικρά από τα εκείνα που παράγει ο Dilithium Εικόνες 4-4(δημόσιο Dilithium) και 4-5(ιδιωτικό Dilithium). Επίσης και οι τελικές υπογραφές που παράγουν έχουν μεγάλη διάφορα στο μέγεθος όπως παρατηρούμε και από στις Εικόνες 4-6(Dilithium υπογραφή) και 4-10(Falcon υπογραφή) , με την υπογραφή του Falcon να είναι πολύ μικρότερη από την αντιστοιχεί του Dilithium για το ίδιο μήνυμα.



Εικόνα 4-11: Αποτελέσματα δοκίμων κώδικα για τον Falcon

Στην Εικόνα 4-11 παρατηρούμε όπως ήταν αναμενόμενο να γίνει σωστή επαλήθευση του χρήστη μόνο όταν τα στοιχεία και το μήνυμα είναι σωστό.

4.6.4 Τρωτά σημεία του Falcon

Όπως είδαμε και στο υποκεφάλαιο 4.5.4 για τον Dilithium, οι δημιουργεί του Falcon στο επίσημο έγγραφο (URL: Falcon (falcon-sign.info)) που έχουν καταθέσει στον NIST για τον τρίτο γύρω έγκρισης των υποψήφιων PQC, ισχυρίζονται ότι ο Falcon είναι θωρακισμένος από γνώστες επιθέσεις. Ωστόσο μια ομάδα ερευνητών στο παρακάτω paper (URL: The Hidden Parallelepiped Is Back Again: Power Analysis Attacks on Falcon (d-nb.info)) εφάρμοσε μια ειδική κατηγορία των επιθέσεων πλευρικού καναλιού γνώστες ως ανάλυση ισχύεις (power analysis). Η επίθεση εφαρμόστηκε κατά την εφαρμογή του Gaussian sampling algorithm, οι ομάδα απέδειξε και επιβεβαιωθήκαν από μια δεύτερη ομάδα ερευνητών (URL: 224.pdf (iacr.org)) ότι κάποιος μπορεί να αντλήσει σημαντικές πληροφορίες που μπορούν να οδηγήσουν σε πλήρη ανάκτηση του μυστικού κλειδιού, αναλύοντας δείγματα από την ισχύει που χρειάζεται να εκτελεστεί ο προαναφερόμενος αλγόριθμος σε συνδυασμό με την εκτέλεση μιας παραλλαγή της παραμορφωμένης επίθεσης παραλληλεπίπεδου. Η παρακάτω επίθεση είναι αρκετά απαιτητική όσον αφορά τους υπολογιστικούς πόρους και τις μετρήσεις που απαιτεί να εκτελεστούν, ωστόσο επιτρέπει την αξιολόγηση μιας νέας ευπάθειας η οποία μπορεί να την αποτρέψεις με την μέθοδο του masking. Ο NIST όπως και στην περίπτωση του Falcon δεν έχει έκδοση κάποια επίσημη ανακοίνωση για την σημαντικότητα της παρακάτω επίθεσης.

4.6.5 Falcon vs Dilithium

Αντλώντας δεδομένα από τα επίσημα papers που έχουν κατατεθεί στον NIST από τους δημιουργούς των αλγορίθμων μπορούμε να κατασκευάσουμε τον παρακάτω πίνακα:

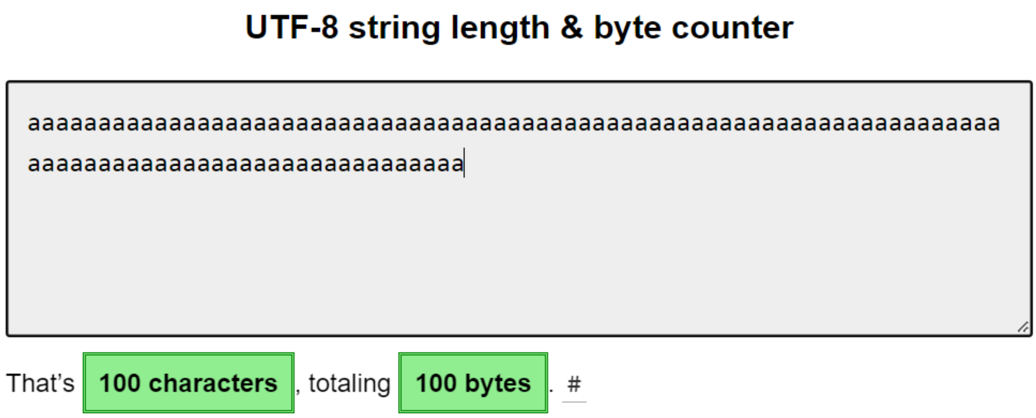
Αλγόριθμος	Επίπεδο Ασφαλείας	Δημόσιο Κλειδί(bytes)	Μέγεθος Υπογραφή(bytes)	n	q
Dilithium$_2$	2	1.312	2.420	256	8.380.417
Dilithium$_3$	3	1.952	3.293	256	8.380.417
Dilithium$_5$	5	2.592	4.595	256	8.380.417
Falcon$_{512}$	1	897	666	512	12.289
Falcon$_{1024}$	5	1.793	1.280	1.024	12.289

Αναλύοντας τις στήλες του πίνακα έχουμε αρχικά το επίπεδο ασφάλειας βάση κάθε εκδοχής των αλγορίθμων με βάση τα πρότυπα του NIST (URL: Security Requirements for Cryptographic Modules (nist.gov)). Στην συνέχεια έχουμε το μέγεθος σε bytes για το δημόσιο κλειδί και την υπογράφει που παράγει ο κάθε αλγόριθμος. Τέλος έχουμε τις μεταβλητές n και q, όπου n είναι το μέγεθος των διαστάσεων των πλεγμάτων και q είναι ένας πρώτος αριθμός για τις πράξεις module.

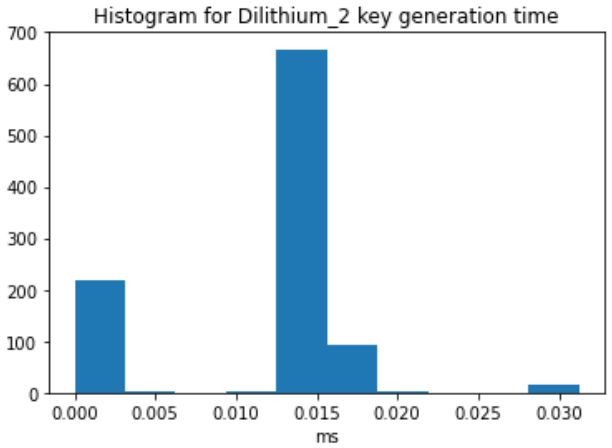
Παρατηρώντας τον πίνακα αλλά και τις εικόνες στα υποκεφάλαια 4.5.3 και 4.6.3 όλοι οι παράμετροι του Falcon είναι πολύ μέγεθος από εκείνων του Dilithium. Ωστόσο συμφώνα με τα ευρήματα μιας ερευνητικής ομάδας του τμήματος πληροφορικής του University of Colorado, Colorado Springs (URL: [1 Security Comparisons and Performance Analyses of Post-Quantum Signature Algorithms.pdf (uccs.edu)]) καθώς επίσης και από πειράματα που πραγματοποιήθηκαν με τους κώδικες που παρουσιάστηκαν στα υποκεφάλαια 3.5.3 και 3.6.3 ο Dilithium σε κάποιες λειτουργίες είναι πιο γρήγορος από τον Falcon. Το πόριζα αυτό προκύπτει εξετάζοντας τα δεδομένα από το πείραμα που πραγματοποίησε η ερευνητική ομάδα στο Colorado αλλά και μέσα από την διεξαγωγή του ιδίου πειράματος, του οποίου τα ευρήματα που προέκυψαν θα παρουσιαστούν στην συνέχεια.

Στο παρακάτω πείραμα όπως και στο paper της ερευνητικής ομάδας από το Πανεπιστήμιο του Colorado, πραγματοποιήθηκαν για 1000 φόρες σε ένα μήνυμα 100 bytes η εξής διεργασία: παραγωγή νέου δημόσιου και ιδιωτικού κλειδιού, υπογράφει του μηνύματος κάθε φόρα με το νέο ιδιωτικό κλειδί που δημιουργήθηκε και επιβεβαίωση του υπογεγραμμένου μηνύματος με το αντίστοιχο νέο δημόσιο κλειδί. Ως χρόνος εκτέλεσης κάθε διεργασίες θεωρείται η μέση τιμή μεταξύ των χιλίων χρονικών δοκίμων.

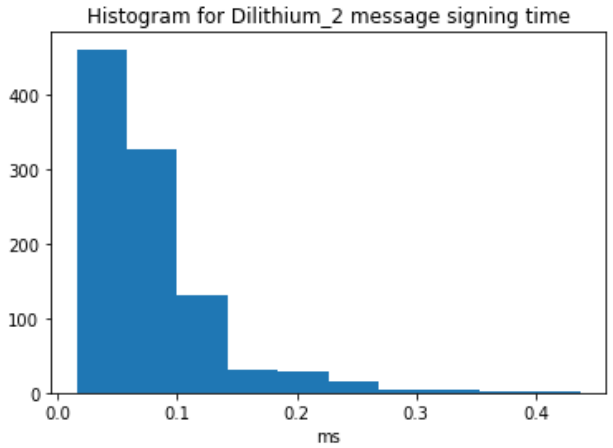
Αποτελέσματα:



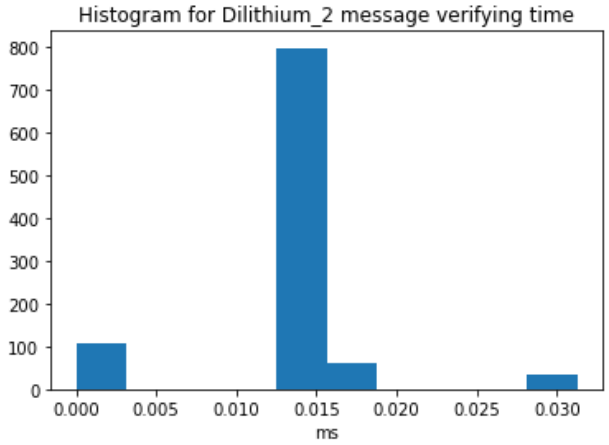
Εικόνα 4-12: Τυχαίο μήνυμα 100 bytes



Σχήμα 4-3: Dilithium 2 key generation time histogram



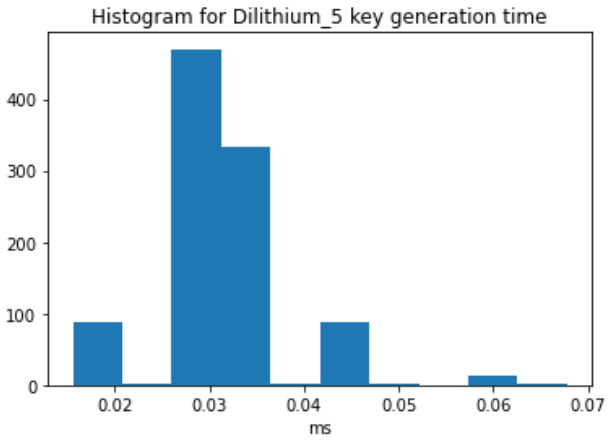
Σχήμα 4-4: Dilithium 2 message signing time histogram



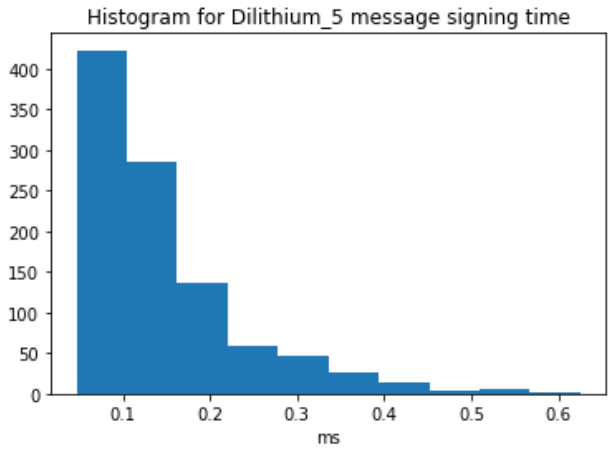
Σχήμα 4-5: Dilithium 2 message verifying time histogram

```
/n ==================== Caclulate the averange runing times ======================

Averange key generation time for  Dilithium_2 =  0.012439797878265381
Averange message siging time for  Dilithium_2 =  0.0742984426021576
Averange message verifying time for Dilithium_2 =  0.0144691481590271


Experiment time=  101.28639650344849
/n ==================== end ======================
```

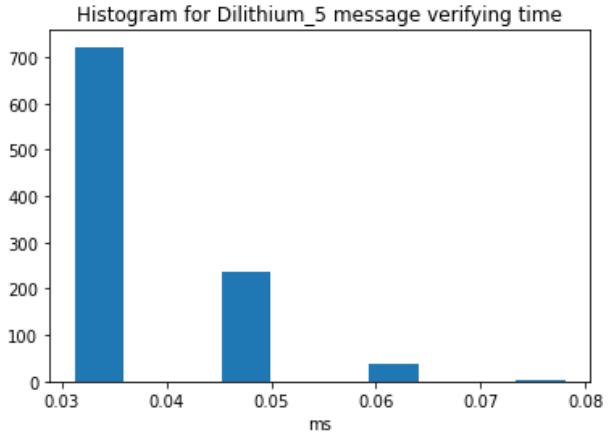
Εικόνα 4-13: Dilithium 2 , μέσοι χρόνοι διεργασιών



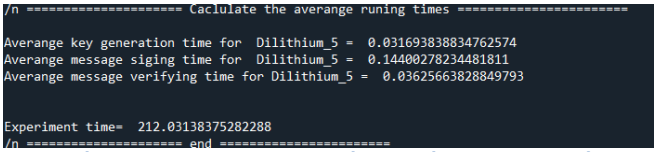
Σχήμα 4-6:Dilithium 5 key generation time histogram



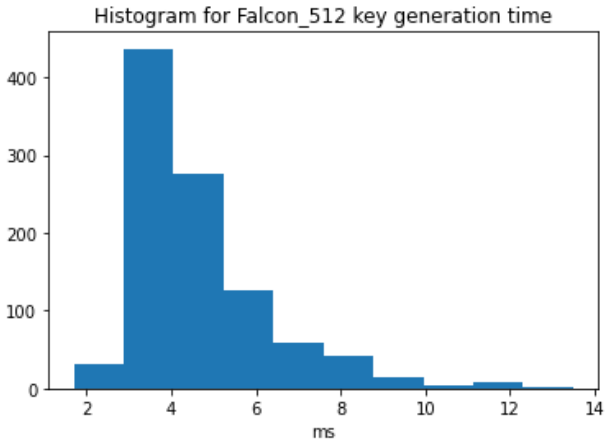
Σχήμα 4-7: Dilithium 5 message signing time histogram



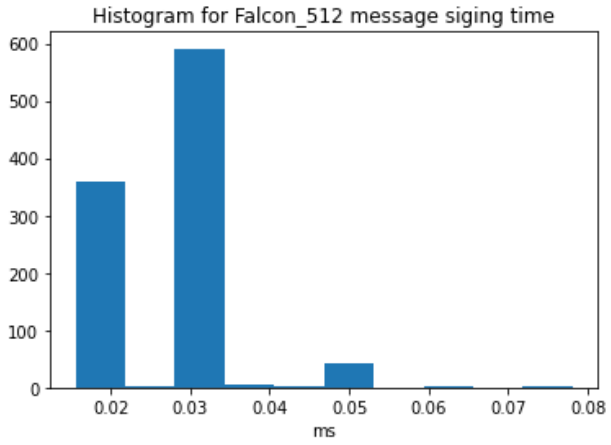
Σχήμα 4-8: Dilithium 5 message verifying time histogram



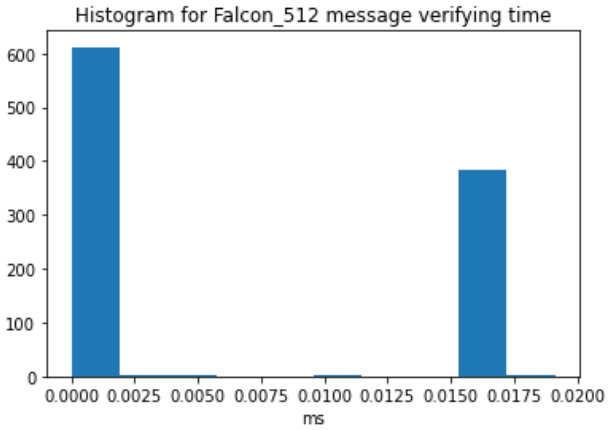
Εικόνα 4-14: Dilithium 5 , μέσοι χρόνοι διεργασιών



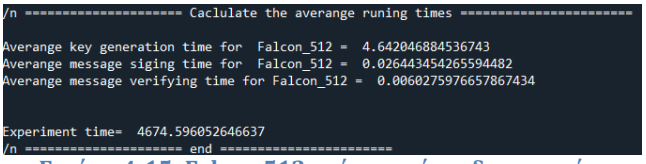
Σχήμα 4-9: Falcon 512 key generation time histogram



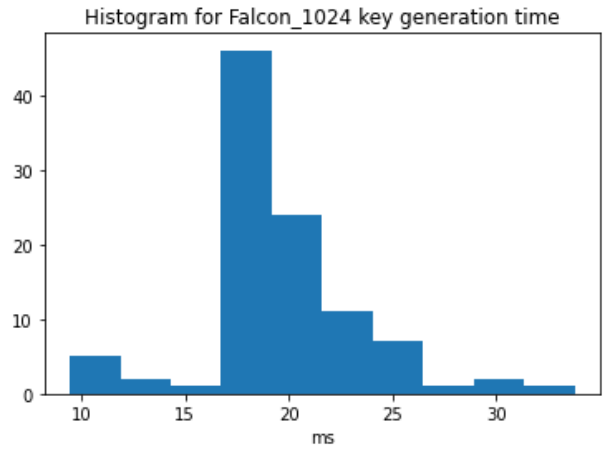
Σχήμα 4-10: Falcon 512 message signing time histogram



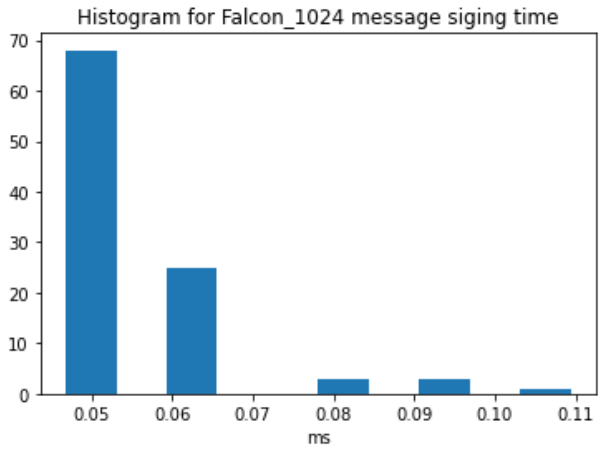
Σχήμα 4-11: Falcon 512 message verifying time histogram



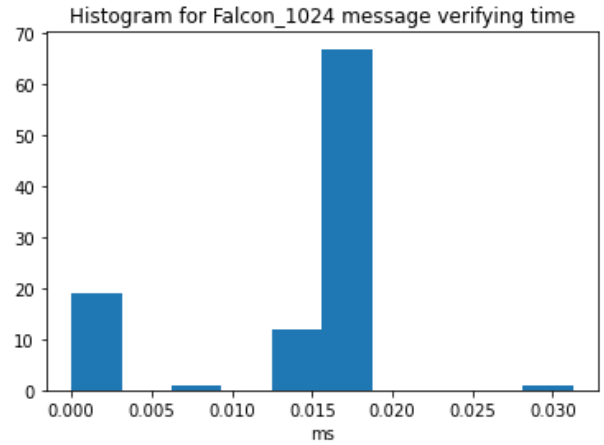
Εικόνα 4-15: Falcon 512 , μέσοι χρόνοι διεργασιών



Σχήμα 4-12: Falcon 1024 key generation time histogram



Σχήμα 4-13: Falcon 1024 message signing time histogram

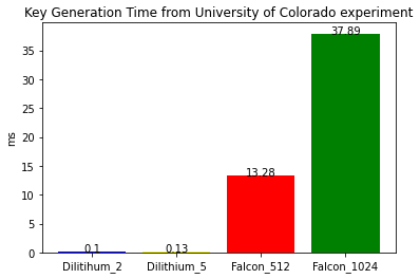


Σχήμα 4-14: Falcon 1024 message verifying time histogram

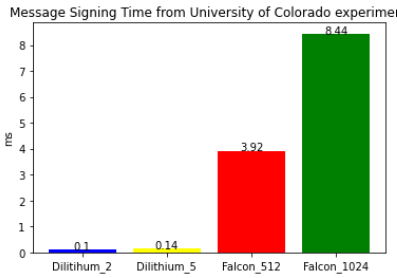
```
/n ==================== Caclulate the averange runing times ======================

Averange key generation time for  Falcon_1024 =  19.53306696653366
Averange message siging time for  Falcon_1024 =  0.0537467098236084
Averange message verifying time for Falcon_1024 =  0.012718238830566407


Experiment time=  1959.9531915187836
/n ==================== end ======================
```

Εικόνα 4-16: Falcon 1024 , μέσοι χρόνοι διεργασιών

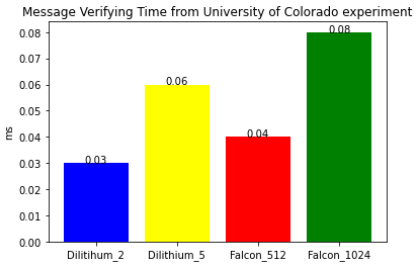
Συγκεντρώνοντας τις πληροφορίες από τα Σχήματα 4-13,4-14,4-15 και 4-16 καθώς επίσης και από τα δεδομένα από το paper της ερευνητικής ομάδας του πανεπιστημίου του Colorado, προκύπτουν τα παρακάτω διαγράμματα:



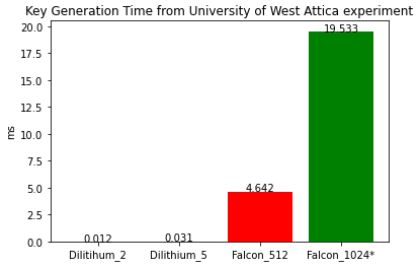
Σχήμα 4-15: Key generation time, Col



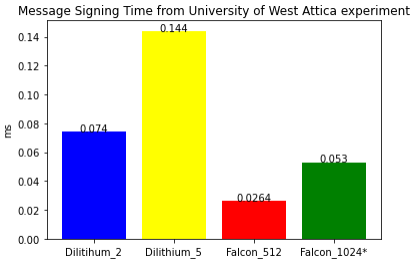
Σχήμα 4-16: Message signing time, Col



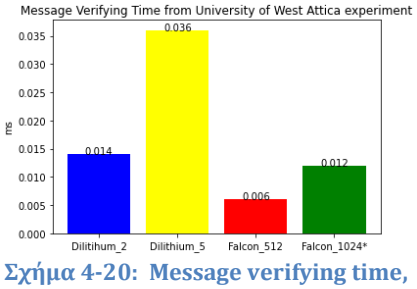
Σχήμα 4-17: Message verifying time, Col



Σχήμα 4-18: Key generation time, Att



Σχήμα 4-19: Message signing time, Att



Σχήμα 4-20: Message verifying time, Att

Αναλύοντας τα Σχήματα 4-16,4-17,4-19 και 4-20 παρατηρούμε πως στα δεδομένα από το Πανεπιστήμιο του Colorado ο Dilithium αλγόριθμος στην υπογραφή όσο και στην εξακρίβωση ενός μηνύματος είναι πιο γρήγορος από τον Falcon. Ωστόσο στο πείραμα που πραγματοποιήθηκε για τις ανάγκες τις παρακάτω διπλωματικής ο Falcon και στις παρακάτω δύο κατηγόριες είναι πιο γρήγορος από τον Dilithium. Η διάφορα αυτή είναι λογική διότι ανάλογα με τον τρόπο που έχει αναπτυχθεί μια συνάρτηση για να υπολογίσει διαφόρους εσωτερικές μεταβλητές επηρεάζεται και η γενική επίδοση του αλγορίθμου. Παρόλα αυτά στην κατηγόρια της παραγωγής των κλειδιών και τα δύο σετ πειραμάτων(Σχήμα 3-15 και 3-18) συμφωνούν πως ο Dilithium είναι αρκετά πιο γρήγορος από τον Falcon. Η διάφορα στον χρόνο είναι τόσο σημαντική που για στον Falcon 1024 το πλήθος από τις 1000 δοκιμές μειώθηκε στις 100 διότι ο χρόνος για να ολοκληρωθεί ένας συνολικός κύκλος δημιουργίας κλειδιών -υπογράφεις μηνύματος - επιβεβαίωση μηνύματος ήταν πολύ μεγάλος και απαιτούσε πολλούς πόρους που οδηγούσαν το σύστημα να σταματήσει την εκτέλεση του python κώδικα διότι ξέμενε από τους αναγκαίους πόρους.

Συμπέρασμα

Στο ερώτημα ποίο είναι το πιο αποδοτικό LBC για την δημιουργία ηλεκτρονικών υπογραφών , βάση των δεδομένων που αναλύθηκαν στο παρακάτω υποκεφάλαιο η απάντηση εξαρτάται την εφαρμογή. Διότι ο Dilithium μπορεί να οδήγηση σε προβλήματα στο δίκτυο λόγου του μεγαλύτερου μεγέθους του δημόσιου κλειδιού και του υπογεγραμμένου μηνύματος, ενώ ο Falcon μπορεί να οδήγηση σε delays λόγου του τεράστιου χρόνου που απαιτείτε για να παραχθούν τα ζεύγη δημόσιου-ιδιωτικού κλειδιού. Ωστόσο δεν υπάρχει ακόμα κάποια πρακτική εφαρμογή που να αξιοποιεί κάποιον από τους δύο αλγορίθμους η εταιρία KeyFactory προσφέρει το ανοικτού λογισμικού προϊών EJBC (EJBCA in Practice: Use cases for Open-Source CA) το οποίο προσφέρει διάφορες δυνατότητες όπως για παράδειγμα δημιουργία ηλεκτρονικού πιστοποιητικού με κάποιον από τους PQC που έχει προτείνει ο NIST, άλλα όπως υπογραμμίζει η εταιρία το προϊόν αυτό προορίζεται μόνο για πειραματικούς σκοπούς διότι, οι αλγόριθμοι που υλοποιούνται έως και το 2024 που θα πιστοποιηθούν επίσημα από τον NIST δεν θεωρούνται ακόμα έτοιμη για κάποια πρακτική εφαρμογή.

4.7 SPHINCS⁺

Ο SPHINCS⁺ αποτελεί έναν PQC για ψηφιακές υπογραφές που βασίζεται σε συναρτήσεις κατακερματισμού(URL: SPHINCS+ – Resources). Το κρυπτοσύστημα αυτό αποτελεί τροποποίηση του προηγουμένως προτεινόμενου σχήματος SPHINCS χρησιμοποιώντας Merkle υπέρ δέντρο. Ο SPHINCS⁺ αξιοποιεί ένα υπέρ δέντρο Merkle το οποίο απαρτίζεται από ένα μεγάλο πλήθος δυαδικών δέντρων κατακερματισμού, στο οποίο κάθε κόμβος στο δυαδικό δέντρο κατακερματισμού σχετίζεται με μια συμβολοσειρά κατακερματισμού και έχει ακριβώς δύο θυγατρικούς κόμβους τέλος οι συμβολοσειρές κατακερματισμού υπολογίζονται ως συνάρτηση των χορδών των παιδιών. Για τη κατασκευή κάθε υπό δέντρου, παράγονται αρκετά ζεύγη μιας χρήσης δημόσιου-ιδιωτικού κλειδιού τύπου Winternitz (WOTS⁺),τα συμπιεσμένα δημόσια κλειδιά χρησιμοποιούνται στη συνέχεια ως φύλλα υπό δέντρου, ενώ τα αντίστοιχα ιδιωτικά κλειδιά χρησιμοποιούνται για την υπογραφή ριζών υπό δέντρων χαμηλότερου επιπέδου. Στο χαμηλότερο επίπεδο, τα ζεύγη κλειδιών Winternitz υπογράφουν τα δημόσια κλειδιά των σχημάτων FORS , αντίστοιχα με τη σειρά τους τα στοιχεία FORS υπογράφουν στο τέλος τα πραγματικά μηνύματα. Για ένα δεδομένο FORS στοιχείο η ασφάλεια της υπογραφής μειώνεται με κάθε υπογεγραμμένο μήνυμα με το στοιχείο αυτό, ωστόσο, τα FORS στοιχεία στο υπέρ δέντρο επιλέγεται ψευδοτυχαία για αυτό και η πιθανότητα επιλογής παρόμοιου FORS στοιχείου πολλές φορές είναι μικρή, για αυτόν τον λόγο ο SPHINCS⁺ θεωρείτε ένα stateless σχήμα κατακερματισμού, διότι δεν απαιτεί να διατηρείτε ο αριθμός των FORS κλειδιών που έχουν ειδή χρησιμοποιηθεί. Για την δημιουργία του υπέρ δέντρο εφαρμόζεται το XMSS σχήμα, στο οποίο υπάρχουν από 10 έως 20 στρώματα υπό δέντρων οπού το ύψος του καθενός κυμαίνεται περίπου από 3 έως 8 επίπεδα ανάλογα με το μέγεθος της ασφάλειας που παρέχει το σχήμα και άλλες διάφορες ιδιότητες. Το πλήρες υπέρ δέντρο δεν διατηρείται στη μνήμη διότι τα απαραίτητα μέρη δημιουργούνται δυναμικά όταν χρειάζεται να πραγματοποιηθεί η διαδικασία της υπογραφής.

Στον παρακάτω πίνακα παρουσιάζονται η δύο διαφορετικές εκδοχές του αλγόριθμου :

	Επίπεδο Ασφαλείας NIST	Δημόσιο Κλειδί (bytes)	Ιδιωτικό Κλειδί (bytes)	Μέγεθος Υπογραφή (bytes)
SPHINCS$^+_{128s}$	1	32	64	8080
SPHINCS$^+_{128f}$	1	32	64	16976
SPHINCS$^+_{192s}$	3	48	96	17064
SPHINCS$^+_{192f}$	3	48	96	35664
SPHINCS$^+_{256s}$	5	64	128	29792
SPHINCS$^+_{256f}$	5	64	128	49216

4.7.1 W-OTS+

Το WOTS+ είναι βελτιωμένη έκδοση του σχήματος WOTS το οποίο προσθέτει τυχαιότητα. Το WOTS+ μειώνει το μέγεθος της υπογραφής διότι χρησιμοποιείται μια συνάρτηση κατακερματισμού με μικρότερες εξόδους για το ίδιο επίπεδο ασφάλειας από ότι στο WOTS. Εφαρμόζοντας την ίδια συνάρτηση κατακερματισμού με το ίδιο μήκος εξόδου και την ίδια παράμετρο Winternitz w, το WOTS+ επιτυγχάνει υψηλότερη ασφάλεια.

Αρχικά σε συγκρίσει με το WOTS το σχήμα WOTS+ προσθέτει ένα νέο δημόσιο κλειδί pk_N στο ειδή υπάρχον σύνολο δημόσιων κλειδιών $pk_0, pk_1 \dots, pk_{N-1}$, το νέο δημόσιο κλειδί είναι και αυτό με την σειρά του ένα σύνολο τυχαίων αριθμών $(r_1, \dots, r_{2^l-1})$ μεγέθους 2^l -1 , οπού l το πλήθος των bits του κλειδιού. Στην συνέχεια το σχήμα WOTS+ εισάγει μια νέα συνάρτηση c η οποία ορίζεται μαθηματικά:

$$c^i(x, PK_N) = \begin{cases} x, i = 0 \\ H\left(c^{i-1}(x, pk_N) \oplus pk_N^i\right), i > 0 \end{cases}$$

οπού το x αναπαριστά το ιδιωτικό κλειδί και το H είναι μια συνάρτηση κατακερματισμού. Για την υπογράφει ενός μηνύματος το σχήμα χωρίζει ένα μήνυμα m σε N κομμάτια $(m_0, \dots, m_{N-1})$ στην συνέχεια υπολογίζει για έλεγχο το άθροισμα $C = \sum_{i=1}^{N-1}(2^l - 1 - m)$, καθώς επίσης και την μεταβλητή $b = m||h$. Η υπογράφει s υπολογίζεται ως:

$s = (s_0, \dots, s_{N-1}) = (H^{b_0}(sk_0, pk_N), H^{b_1}(sk_1, pk_N), \dots, H^{b_{N-1}}(sk_{N-1}, pk_N),$ στην συνέχεια αποστέλλεται στον verifier το ζευγάρι (pk_N, s). Για να γίνει επαλήθευση της υπογραφής ο verifier πρέπει επαναλαμβανόμενα να καταμερίζει τα στοιχεία της υπογραφής s έως ότου επαληθευτούν από το pk.

$$pk_v = (pk_N, \dots, H^{2^l-1-b_0}(sk_0, pk_N), H^{2^l-1-b_1}(sk_1, pk_N), \dots, H^{2^l-1-b_{N-1}}(sk_{N-1}, pk_N)$$

Αν τα στοιχειά του συνόλου pk_V ταιριάζουν με αυτά του συνόλου pk, τότε η υπογραφή γίνεται αποδέκτη.

4.7.2 FORS

Το FORS είναι ένα σχήμα δημιουργίας υπογραφών σύντομου χρονικού διαστήματος βασιζόμενων σε κατακερματισμό το όποιο χρησιμοποιείται ως βάση του SPHINCS⁺. Το FORS θωρακίζετε από επιθέσεις διότι αυξάνει το μέγεθος των κλειδιών κατά έναν συντελεστή k, όπου k ο αριθμός των τυχαίων υποσυνόλων. Ωστόσο το FORS μπορεί να χαρακτηριστικά και ως ένα γενικευμένο στιγμιότυπο του σχήματος HORS από το οποίο κληρονομεί και τα περισσότερα χαρακτηριστικά του. Το FORS είναι παρόμοιο με το HORS ωστόσο στο FORS έχουμε kt σετ ιδιωτικών κλειδιών και k δυαδικά δέντρα κατακερματισμού, το FORS επιλέγει όλες τις τιμές υπογραφής του από το ίδιο σύνολο κλειδιών t, επίσης το FORS παράγει kt μυστικά κλειδιά οπού t από αυτά είναι μυστικά κλειδιά για κάθε ευρετήριο. Το FORS ορίζεται από έναν ακέραιο k και έναν ακέραιο $t=2^a$, τα οποία ορίζουν και το μέγεθος των μηνυμάτων που μπορεί να υπογράψει αφού το string που υπογράφεται από τον αλγόριθμο μπορεί να είναι μεγέθους έως $(k \times a)$ bits.

Δημιουργία FORS κλειδιών:

- Τα FORS ιδιωτικά κλειδιά αποτελούνται από kt τυχαίες τιμές μεγέθους n-bit τα οποία έχουν ομαδοποιηθεί σε k σετ τα οποία περιλαμβάνουν t τιμές, στο SPHINCS⁺ οι τιμές αυτές επηρεάζονται από το SK_{seed}.
- Το FORS δημόσιο κλειδί δημιουργεί k δυαδικά δέντρα hash, μεγέθους a πάνω στο k σετ των t ιδιωτικών κλειδιών. Στο SPHINCS⁺ οι ρίζες των κόμβων συμπιέζονται η τεχνική αυτή έχει ως αποτέλεσμα τα FORS δημόσια κλειδιά για το SPHINCS⁺ framework να έχουν μέγεθος n bit.

Υπογραφεί μηνύματος:

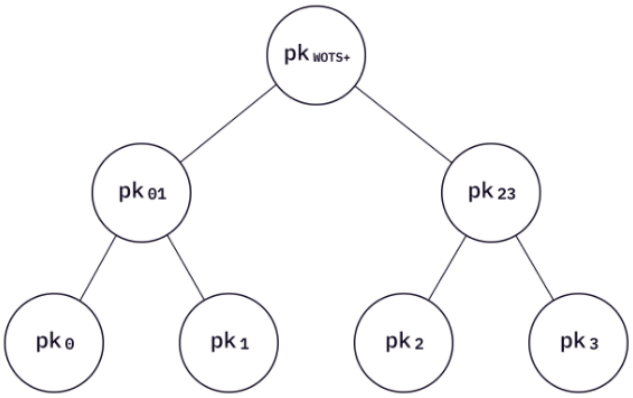
- Δοθέντος ένα μήνυμα μεγέθους ka bits, το FORS διαμελίζει το μήνυμα σε k μέρη το καθένα μεγέθους a bits. Το καθένα από αυτά τα k μέρη αναπαριστάται ως ένας μεμονωμένος κόμβος (0,1,…, t-1) σε καθένα από τα k FORS δέντρα.

Επαλήθευση μηνύματος

- Ο verifier δημιουργεί εκ νέου τη διαδρομή ελέγχου ταυτότητας για κάθε s_i και εξετάζει ένα η υπολογισμένη ρίζα ισούται με το δημοσιευμένο δημόσιο κλειδί.

4.7.3 XMSS

Το WOTS⁺ είναι σχήμα που μπορεί να υπογράψει μόνο μια φόρα, για αυτό και χρειάζεται ένα σύστημα το οποίο θα διαχειριστή τα κλειδιά σε περίπτωση που απαιτείται να υπογραφούν πολλαπλά μηνύματα. Την διαχείριση αυτή την κάνει το XMSS framework , το οποίο χρησιμοποιεί δέντρα Markel για να διαχειριστή τα κλειδιά του σχήματος WOTS⁺. Στο XMSS το δημόσιο κλειδί είναι μια ρίζα ενός XMSS Markel δέντρου ενώ κάθε φύλο του δέντρου είναι ένα σετ από WOTS⁺ δημόσια κλειδιά το οποίο έχει δημιουργηθεί χρησιμοποιώντας μια αλυσίδα από κερματισμένες τιμές από ένα WOTS⁺ ιδιωτικό κλειδί όπως είδαμε στο υποκεφάλαιο 4.7.2, στο Σχήμα 4-21 απεικονίζεται οπτικά η παρακάτω δομή οπού σχηματίζεται ένα δέντρο τύπου L.

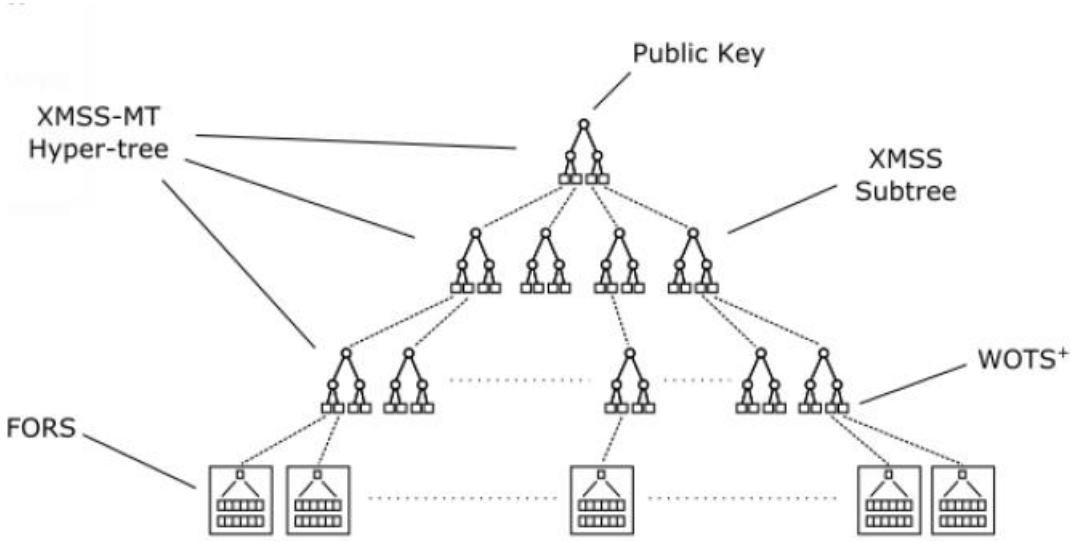


Σχήμα 4-21: Διαχείριση σετ WOTS⁺ κλειδιών με το XMSS

Πηγή: How do hash-based post-quantum digital signatures work? (Part 1) · Eric Zhang, VegeBun · Dora Research Blog (dorahacks.io)

Για να υπογραφεί ένα μήνυμα m, επιλέγεται αρχικά από XMSS δέντρο ένα φύλο αναφοράς i το οποίο ελέγχεται για το αν έχει ξανά χρησιμοποιηθεί , διότι αν δεν έχει ξανά χρησιμοποιηθεί σημαίνει ότι το φύλο αυτό είναι ένα σετ από WOTS⁺ δημόσια κλειδιά. Στην συνέχεια με ένα WOTS⁺ ιδιωτικό κλειδί υπογράφετε το μήνυμα m και παράγεται μια WOTS⁺ υπογραφεί s_i. Η XMSS υπογράφει αποτελείτε από το παρακάτω σύνολο στοιχείων $s_m = (s_i, auth_i, i, leaf_i)$, όπου i είναι η αναφορά στο WOTS⁺ ζεύγος κλειδιών και $leaf_i$, η Markel ρίζα ενός σετ από WOTS⁺ δημόσια κλειδιά. Τέλος για να επαληθευτεί η υπογραφή, αρχικά επαληθεύεται η υπογραφή s_i μέσω του i στην συνέχεια επαληθεύεται η ακεραιότητα του $leaf_i$ χρησιμοποιώντας το $auth_i$ και ένα XMSS δημόσιο κλειδί.

4.7.4 Αλγόριθμος/Framework



Σχήμα 4-22: Δομή του σχήματος ηλεκτρονικών υπογραφών SPHINCS⁺

Πηγή: E.O. Kiktenko, A.A. Bulychev, P.A. Karagodin, N.O. Pozhar, M.N. Anufriev and A.K. Fedorov, SPHINCS⁺ post-quantum digital signature scheme with Streebog hash function, URL: 1904.06525.pdf (arxiv.org) , Σελίδα 2

Αναλύοντας ακόμα πιο πολύ τον αλγόριθμο η πιο σωστά το framework SPHINCS⁺ όπως απεικονίζεται στο Σχήμα 4-22, για να εφαρμοστεί σωστά πρέπει να εκτελεστούν οι παρακάτω λειτουργίες:

- Το FORS υπογράφει ένα μήνυμα με ιδιωτικό κλειδί ενώ τα αντίστοιχα δημόσια κλειδιά FORS αποτελούν τα φύλλα σε k δυαδικά δέντρα, στην συνέχεια οι ρίζες των δέντρων αυτών συνενώνονται για να δημιουργήσουν μια FORS ρίζα. Οι ρίζες του FORS δέντρου υπογράφονται από ένα WOTS⁺ ιδιωτικό κλειδί υπογραφής μίας χρήσης, παράλληλα τα αντίστοιχα WOTS⁺ δημόσια κλειδιά σχηματίζουν τα φύλλα από d-στρώματα τύπου Merkle υπό δέντρων σε SPHINCS⁺ υπέρ δέντρο στη συνέχεια η βάση αυτού του υπέρ δέντρου υπογράφει τις ρίζες FORS με WOTS. Ενώ οι ρίζες της βάση του Merkle υπό δέντρου υπογράφεται από το WOTS⁺ δημόσιο κλειδί του φύλου, του υπό δέντρου που βρίσκεται στο επόμενο επίπεδο.

- Οι ρίζες των υπό δέντρων κατά συνέπεια υπογράφονται από τα στρώματα του αντιστοίχου υπό δέντρου που υπάγονται, η διαδικασία αυτή επαναλαμβάνεται μέχρι το κορυφαίο υπό δέντρο. Το υπό δέντρο του ανώτερου στρώματος σχηματίζει τη ρίζα υπέρ δέντρου που χρησιμοποιείται από τον verifier για τον έλεγχο της αξιοπιστίας της υπογραφής.
- Μια SPHINCS⁺ υπογραφή απαρτίζεται από την FORS υπογραφή, WOTS⁺ υπογραφή σε κάθε στρώμα και τη διαδρομή προς τη ρίζα κάθε υπό δέντρου μέχρι να φτάσουμε στη ρίζα του.
- Μια SPHINCS⁺ υπογραφή επαληθεύεται, αρχικά επαληθεύοντας την FORS υπογραφή, τις υπογραφές WOTS+ καθώς επίσης και τη διαδρομή προς τη ρίζα κάθε υπό δέντρου. Φτάνοντας στην ρίζα του υπέρ δέντρου, η υπογραφή επαληθεύεται εφόσον μπορεί να κατακερματιστεί στην προ-έμπιστη ρίζα του SPHINCS⁺ υπέρ δέντρου.
- Το SPHINCS⁺ δημόσιο κλειδί αποτελείται από δύο τιμές μεγέθους n byte, η πρώτη τιμή είναι ο ριζικός κόμβος του δέντρου κορυφής στο υπέρ δέντρο (PK_{root}) και η δεύτερη τιμή είναι ένα τυχαίος δημόσιος σπόρος μεγέθους n byte (PK_{seed}). Το SPHINCS⁺ ιδιωτικό κλειδί αποτελείται από το δημόσιο κλειδί μαζί με ακόμα δύο τυχαίους σπόρους μεγέθους n byte. Από τους οποίους ο ένας χρησιμοποιείται για την δημιουργία των μυστικών κλειδιών WOTS⁺ και FORS (SK_{seed}) ενώ ο δεύτερος μυστικός σπόρος χρησιμοποιείται για την τυχαιοποιημένη σύνοψη μηνυμάτων (SK_{prf}) .

4.7.5 Δοκιμές αλγορίθμου

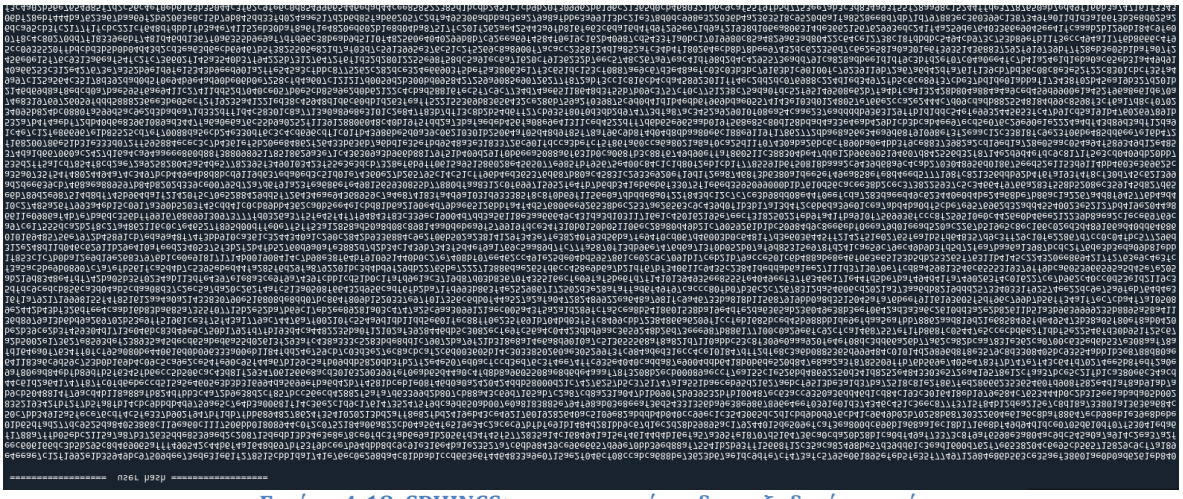
Τρέχοντας το υποθετικό παράδειγμα που χρησιμοποιήθηκε για τις δοκιμές του Dilithium (υποκεφάλαιο 4.5.3) και του Falcon υποκεφάλαιο (4.6.3), για να εξετάσουμε τον SPHINCS⁺ θα χρησιμοποιήσουμε την βιβλιοθήκη pyspx (URL: PySPX · PyPI) που προσφέρει η python. Επαναλαμβάνοντας το ίδιο πείραμα μπορούμε να παρατηρήσουμε τα έξεις:

```
================== user public key ==================
63785df97fb86e0ce84f41b10d3e4097efebc0d0f505cfe40197400ff7fb433f
==================  user secret key ==================
1cccee545b6283c53a59cbb155bb58c941403c5cb17f03136f66d6bd8b3f3a5663785df97fb86e0ce84f41b10d3e4097efebc0d0f505cfe40197400ff7fb433f
```

Εικόνα 4-17: Δημόσιο κλειδί και ιδιωτικό για τον $SPHINCS^{+}_{128}$ σε δεκαεξαδική μορφή

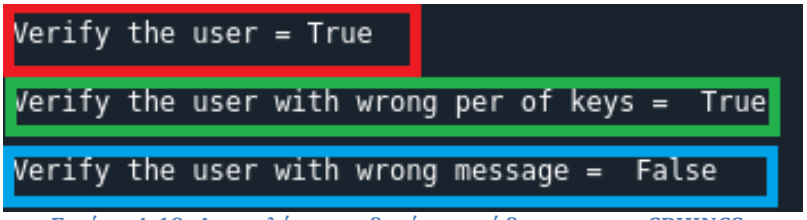
Σε σύγκριση με τα δημόσια και τα ιδιωτικά κλειδιά του Dilithium και του Falcon, το SPHINCS⁺ όπως βλέπουμε και στην Εικόνα 4-17 έχει μικρούς μεγέθους δημόσιο αλλά και ιδιωτικό κλειδί.

Ωστόσο στην Εικόνα 4-18 παρατηρούμε το μεγάλο μέγεθος που έχει το τελικό υπογεγραμμένο μήνυμα:



Εικόνα 4-18: SPHINCS⁺$_{128}$ τπογραφή σε δεκαεξαδική μορφή

Το οποίο παρατηρούμε οπτικά έχει μεγαλύτερο μέγεθος και από την αντίστοιχη Dilithium υπογραφή (Εικόνα 4-3) και αυτό οφείλεται όπως είδαμε και στο υποκεφάλαιο 4.7.4 από τα δεδομένα που φέρει η υπογραφή μαζί της, γενικά το SPHINCS⁺ έχει υπογραφές μεγάλου μεγέθους.

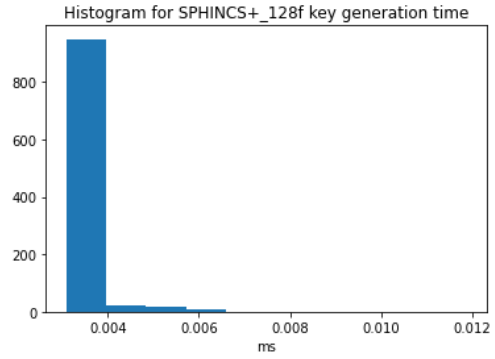


Εικόνα 4-19: Αποτελέσματα δοκίμων κώδικα για τον SPHINCS⁺

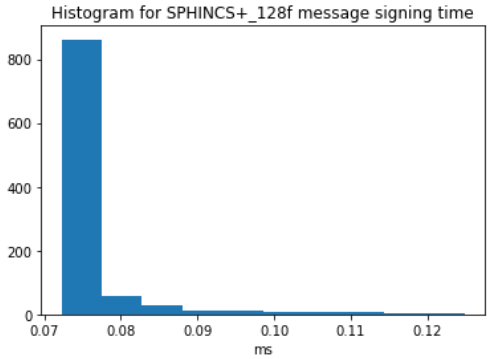
Στις Εικόνες 4-7 (Dilithium) , 4-11 (Falcon) είδαμε πως το τεστ στο πράσινο ορθογώνιο επέστρεψε false, δηλαδή ή υπογραφή δεν αναγνωριζόταν από ένα ξένο δημόσιο κλειδί, ωστόσο στο SPHINCS⁺ στον μηχανισμό επαληθεύσεις αν δύο κλειδιά έχουν πρόκυψη από το ίδιο seed όπως είδαμε και στο υποκεφάλαιο 4.7.4 μπορούν να θεωρηθούν ισοδύναμα και ας μην είναι τα ιδία. Για το λόγο αυτό χρειάζεται προσοχή στον τρόπο που εφαρμόζεται το SPHINCS⁺ .

Στην συνέχεια θα υλοποιήσουμε το πείραμα που πραγματοποιήθηκε στο υποκεφάλαιο 4.6.5, για τις διάφορες παράλεγες του SPHINCS⁺ . Για την υλοποίηση του πειράματος κάθε ζευγάρι δημόσιου ιδιωτικού κλειδιού δημιουργείτε κάθε φόρα από ένα νέο seed. Τα αποτελέσματα του πειράματος παρουσιάζονται παρακάτω:

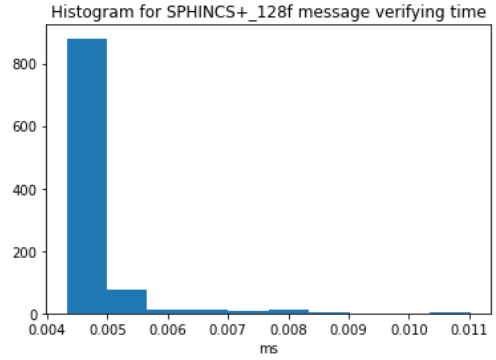
Αποτελέσματα



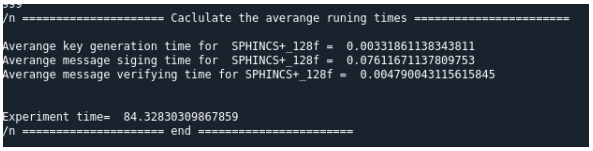
Σχήμα 4-23: SPHINCS⁺$_{128f}$ key generation time histogram



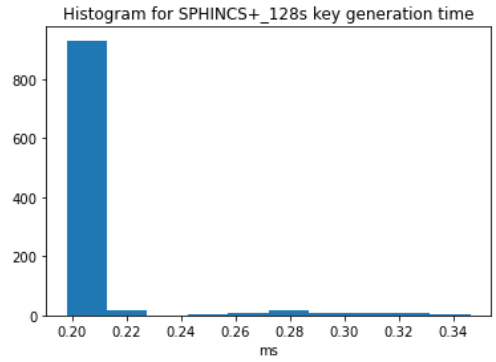
Σχήμα 4-24: SPHINCS⁺$_{128f}$ message signing time histogram



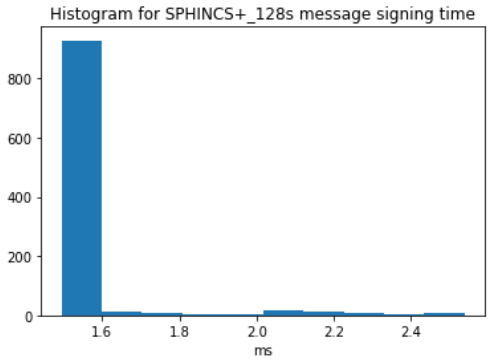
Σχήμα 4-25: SPHINCS⁺$_{128f}$ message verifying time histogram



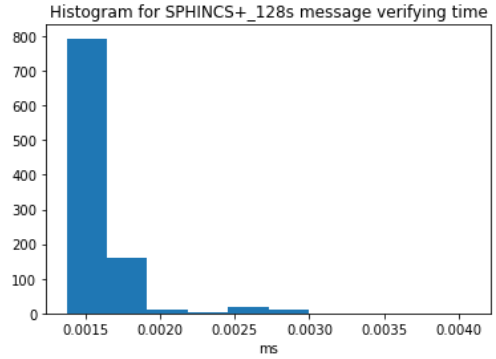
Εικόνα 4-20: SPHINCS⁺$_{128f}$, μέσοι χρόνοι διεργασιών



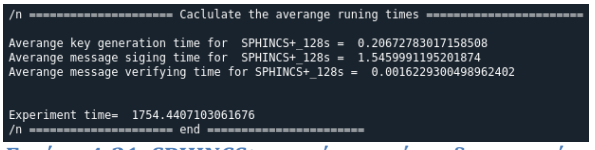
Σχήμα 4-26: SPHINCS⁺$_{128s}$ key generation time histogram



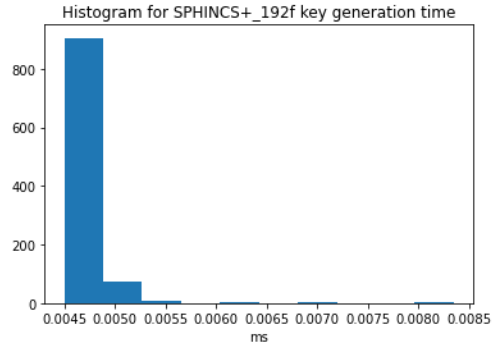
Σχήμα 4-27: SPHINCS⁺$_{128s}$ message signing time histogram



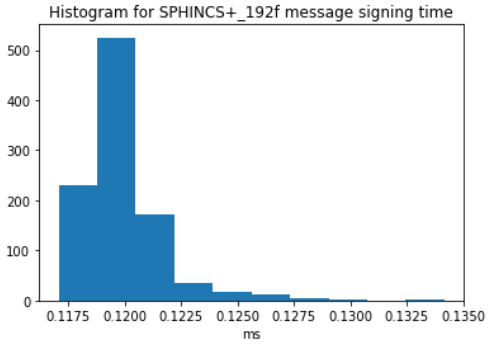
Σχήμα 4-28: SPHINCS⁺$_{128s}$ message verifying time histogram



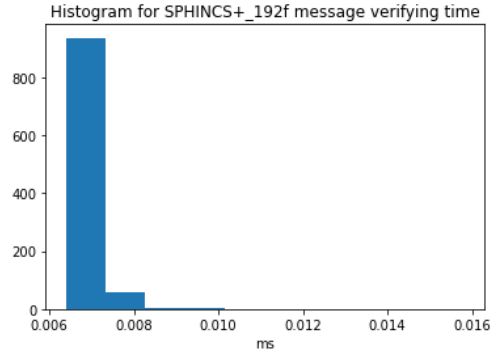
Εικόνα 4-21: SPHINCS⁺$_{128s}$, μέσοι χρόνοι διεργασιών



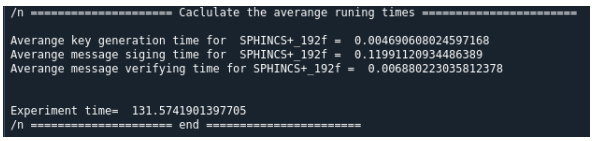
Σχήμα 4-29: SPHINCS⁺$_{192f}$ key generation time histogram



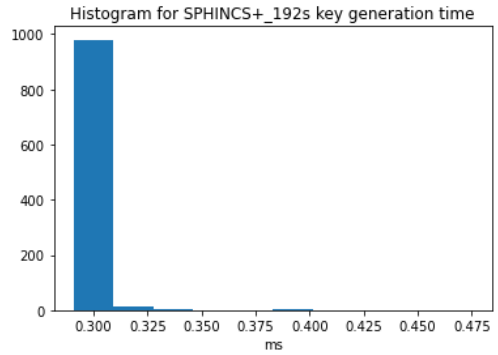
Σχήμα 4-30: SPHINCS⁺$_{192f}$ message signing time histogram



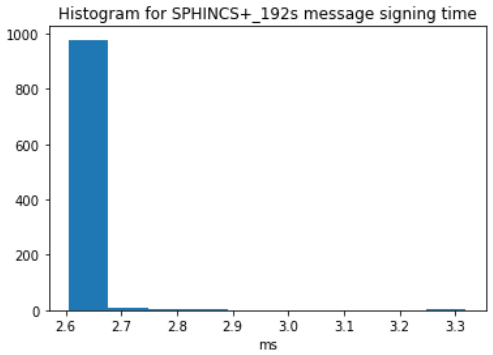
Σχήμα 4-31: SPHINCS⁺$_{192f}$ message verifying time histogram



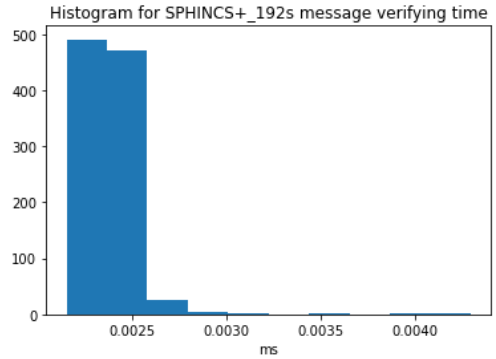
Εικόνα 4-22: SPHINCS⁺$_{192f}$, μέσοι χρόνοι διεργασιών



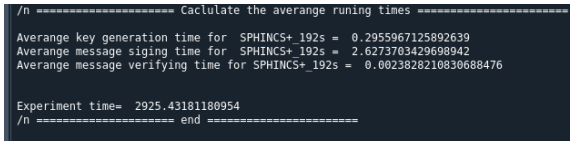
Σχήμα 4-32: SPHINCS⁺$_{192s}$ key generation time histogram



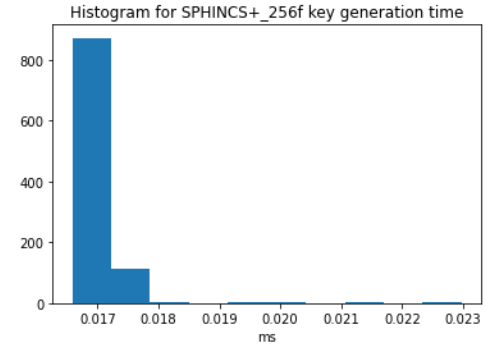
Σχήμα 4-33: SPHINCS⁺$_{192s}$ message signing time histogram



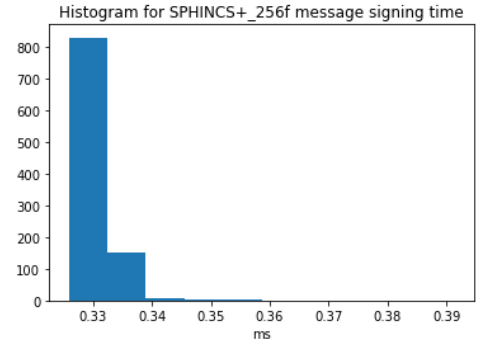
Σχήμα 4-34: SPHINCS⁺$_{192s}$ message verifying time



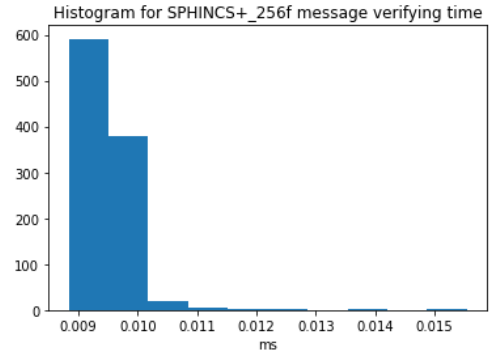
Εικόνα 4-23: SPHINCS⁺$_{192s}$, μέσοι χρόνοι διεργασιών



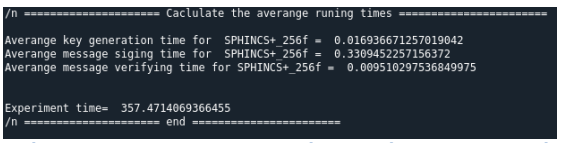
Σχήμα 4-35: SPHINCS⁺$_{256f}$ key generation time histogram



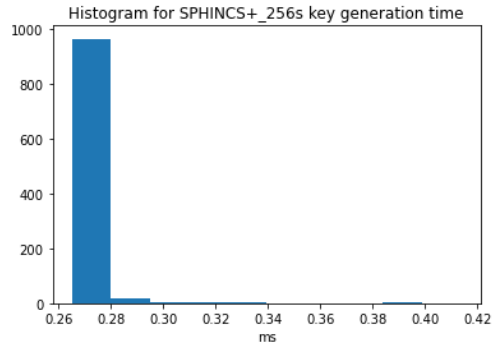
Σχήμα 4-36: SPHINCS⁺$_{256f}$ message signing time histogram



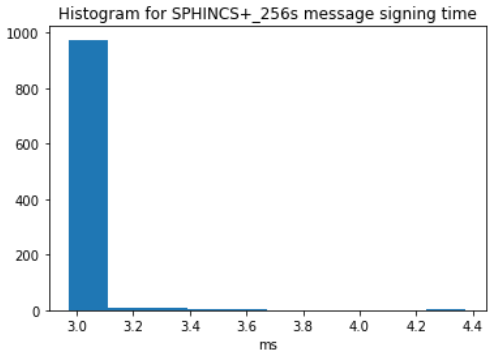
Σχήμα 4-37: SPHINCS⁺$_{256f}$ message verifying time histogram



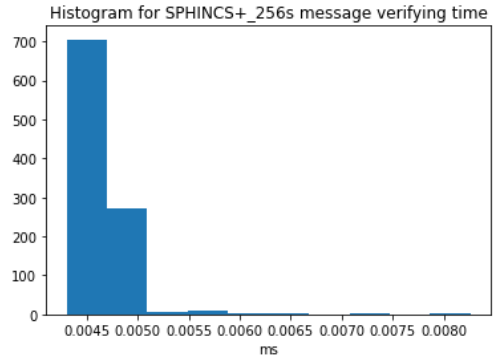
Εικόνα 4-24: SPHINCS⁺$_{256f}$, μέσοι χρόνοι διεργασιών



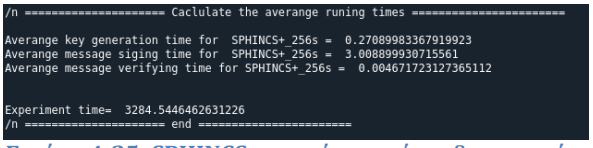
Σχήμα 4-38: SPHINCS⁺256s key generation time histogram



Σχήμα 4-39: SPHINCS⁺256s message signing time histogram



Σχήμα 4-40: SPHINCS⁺256s message verifying time histogram



Εικόνα 4-25: SPHINCS⁺256s, μέσοι χρόνοι διεργασιών

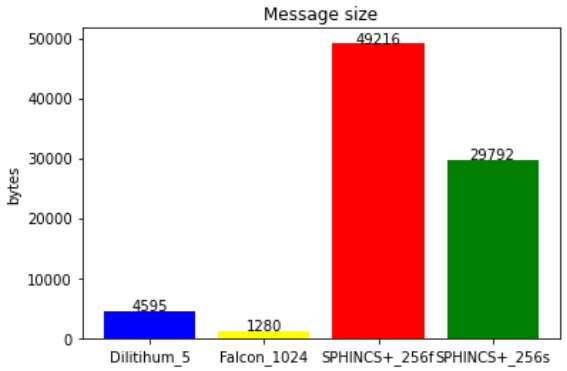
Ανάλυση δεδομένων

Στο παρακάτω πίνακα απεικονίζονται οι μέσοι χρόνοι που θέλει κάθε παραλλαγή του SPHINCS⁺ για να εκτέλεση κάθε μια από τις τρεις βασικές λειτουργίες που εκτελεί ένα κρυπτογραφικό σχήμα υπογραφών.

	key generation(ms)	Message signing(ms)	Message verifying(ms)
128_f	0,003	0,076	0,004
128_s	0,206	1,545	0,006
192_f	0,004	0,119	0,009
192_s	0,25	2,627	0,012
256_f	0,016	0,33	0,023
256_s	0,27	3	0,03

Παρατηρώντας τα δεδομένα του πίνακα αυτού, αρχικά φαίνεται η διάφορα που επιφέρουν στον χρόνο εκτέλεσης, η δύο διαφορετικές παράλεγες που προσφέρει κάθε μια από τις τρείς διαφορετικές εκδοχές του SPHINCS⁺. Το s και f στο τέλος κάθε SPHINCS⁺ εκδοχής, είναι τα αρχικά από το fast SPHINCS⁺ και αντιστατικά slow SPHINCS⁺. Στην fast παραλλαγή το SPHINCS⁺ προσφέρει (ταχύτερος υπολογισμός, μεγαλύτερο μέγεθος υπογραφής) αντίστοιχα στην slow παραλλαγή προσφέρει (πιο αργούς υπολογισμούς, μικρότερο μέγεθος υπογραφής).

Οι χρόνοι του SPHINCS⁺ σε όλες τις λειτουργίες είναι πιο γρήγοροι από τους αντίστοιχους χρόνους που αναλύθηκαν στο υποκεφάλαιο 4.6.5 για τους Dilithium και Falcon, ωστόσο αυτό δεν σημαίνει πως το SPHINCS⁺ παραμένει το ιδανικό framework ηλεκτρονικών υπογραφών διότι όπως φαίνεται και στο Σχήμα 3-41:



Σχήμα 4-41: Μέγεθος υπογραφής για τα PQC σχήματα ηλεκτρονικής υπογραφής

στο οποίο απεικονίζονται το μέγεθος της υπογραφής για κάθε εκδοχή των PQC σχημάτων ηλεκτρονικής υπογραφής που προσφέρουν επίπεδο ασφαλείας 5 σύμφωνα με τα πρότυπα του NIST, οι υπογραφές του SPHINCS⁺ είναι αρκετά πιο μεγάλες από τις αντιστατικές των Dilithium και Falcon, ανάλογα με την εφαρμογή το μέγεθος αυτό μπορεί να προκαλέσει προβλήματα στο δίκτυο.

4.8 Ανάλυση αποτελεσμάτων

Τα δεδομένα για τις στήλες (Επίπεδο Ασφαλείας NIST Δημόσιο Κλειδί(bytes) Ιδιωτικό Κλειδί(bytes) Μέγεθος Υπογραφή(bytes)) μπορούμε να τα βρούμε στις επίσημες ιστοσελίδες οι οποίες έχουν δημιουργηθεί από τους οργανισμούς ανάπτυξης των αλγορίθμων με σκοπό την προώθησης και μελέτη τους.

- Falcon: Falcon (falcon-sign.info)
- Dilithium: Dilithium (pq-crystals.org)
- SPHINCS⁺: SPHINCS+

Στους παρακάτω συνδέσμου μπορούμε να βρούμε επίσης υλοποιημένους τους αλγόριθμους για την διεξαγωγή πειραμάτων σε γλώσσα python. Στα Υποκεφάλαια 4.6.5 και 4.7.5 οι παρακάτω αλγόριθμοι χρησιμοποιήθηκαν για την διεξαγωγή κάποιων πειραμάτων και τα αποτελέσματά τους μπορούμε να δούμε στις στήλες (key generation(ms), Athens Message signing(ms), Athens Message verifying(ms) Athens). Στις στήλες (key generation(ms) Colorado,Message signing(ms) Colorado,Message verifying(ms) Colorado), υπάρχουν τα αποτελέσματα από ένα παρόμοιο πείραμα που έχει πραγματοποιήσει το πανεπιστήμιο του Colorado (URL: 1 Security Comparisons and Performance Analyses of Post-Quantum Signature Algorithms.pdf (uccs.edu)). Τέλος τα δεδομένα στην στήλη (Επίπεδο Ασφαλείας NIST) παρουσιάζουν το επίπεδο ασφάλειας του κάθε αλγόριθμου με βάση των προτύπων του οργανισμού (URL: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths (nist.gov)), συνοπτικά τα επίπεδα ασφάλειας ορίζονται βάσει της ευαισθησίας των δεδομένων και το μεγέθους του κλειδιού του αλγόριθμου, αφού όσο πιο ευαίσθητα τα δεδομένα τόσο μεγαλύτερο το επίπεδο ασφάλειας που πρέπει να προσφέρει ο αλγόριθμος.

Αλγόριθμος	Επίπεδο Ασφαλείας NIST	Δημόσιο Κλειδί(bytes)	Ιδιωτικό Κλειδί(bytes)	Μέγεθος Υπογραφή(bytes)		key generation(ms) Colorado	Message signing(ms) Colorado	Message verifying(ms) Colorado		key generation(ms) Athens	Message signing(ms) Athens	Message verifying(ms) Athens
Dilithium$_2$	2	1.312	2528	2.420		0,1	0,13	0,3		0,012	0,074	0,014
Dilithium$_5$	5	2.592	4864	4.595		0,13	0,14	0,06		0,031	0,144	0,036
Falcon$_{512}$	1	897	1281	666		13,28	3,92	0,04		4,642	0,0264	0,006
Falcon$_{1024}$	5	1.793	2305	1.280		37,89	0,44	0,06		19,533	0,053	0,012
SPHINCS$^+{}_{128f}$	1	32	64	16976		NaN	NaN	NaN		0,003	0,076	0,004
SPHINCS$^+{}_{128s}$	1	32	64	8080		NaN	NaN	NaN		0,206	1,545	0,006
SPHINCS$^+{}_{192f}$	3	48	96	35664		NaN	NaN	NaN		0,004	0,119	0,009
SPHINCS$^+{}_{192s}$	3	48	96	17064		NaN	NaN	NaN		0,25	2,627	0,012
SPHINCS$^+{}_{256f}$	5	64	128	49216		NaN	NaN	NaN		0,016	0,33	0,023
SPHINCS$^+{}_{256s}$	5	64	128	29792		NaN	NaN	NaN		0,27	3	0,03

Αρχικά παρατηρώντας τον πίνακα βλέπουμε πως απουσιάζουν δεδομένα από τον SPHINCS$^+$, αυτό συμβαίνει διότι το paper με το οποίο συγκρίνουμε τα αποτελέσματα μας διεξήγαγε το πείραμα μόνο στον Dilithium και στον Falcon. Στην συνέχεια παρατηρούμε διάφορες στις τιμές των πειραμάτων, αυτό οφείλεται:

- Στο διαφορετικό hardware που τρέξανε οι κώδικες.
- Στην έκδοση της python. Το paper δημοσιεύτηκε το 2021 ενώ η παρακάτω εργασία συγγράφεται το 2023, μέσα σε αυτά τα χρόνια διάφορες ρουτίνες και βιβλιοθήκες της python έχουν αναβαθμιστεί.

Τέλος για να μπορέσουμε να συγκρίνουμε καλύτερα τους διάφορους PQCs που προορίζονται για υπογραφή ψηφιακών εγγράφων και ηλεκτρονική πιστοποίηση/ταυτοποίηση μπορούμε να κατασκευάσουμε τον παρακάτω πίνακα, στον οποίο συμπεριλαμβάνονται τα δεδομένα για το επίπεδο ασφάλειας 5.

Αλγόριθμος	Επίπεδο Ασφαλείας NIST	Δημόσιο Κλειδί(bytes)	Ιδιωτικό Κλειδί(bytes)	Μέγεθος Υπογραφή(bytes)		key generation(ms) Athens	Message signing(ms) Athens	Message verifying(ms) Athens
Dilithium$_5$	5	2.592	4864	4.595		0,031	0,144	0,036
Falcon$_{1024}$	5	1.793	2305	1.280		19,533	0,053	0,012
SPHINCS$^+{}_{256f}$	5	64	128	49216		0,016	0,33	0,023
SPHINCS$^+{}_{256s}$	5	64	128	29792		0,27	3	0,03

Στα υποκεφάλαια 4.5.3, 4.6.3, 4.6.5 και 4.7.5 αναλύοντας τα ευρήματα των πειραμάτων έχουν αναφερθεί παραδείγματα εφαρμογών που ο κάθε αλγόριθμος θα μπορούσε να χρησιμοποιηθεί έναντι κάποιου αλλού, ωστόσο επειδή διάφορες εφαρμογές έχουν και διαφορετικές ανάγκες μπορούμε σε συνδυασμό και με τα δεδομένα του πίνακα να δημιουργήσουμε τους παρακάτω πινάκες, οπού παρουσιάζονται τα προτερήματα και μειονεκτήματα κάθε αλγόριθμου συγκριτικά με τους άλλους .

Dilithium		
Θετικά	**Αρνητικά**	**Ουδέτερα**
+ Γρήγορη υπογράφει εγγράφων.	-Μεγάλο μέγεθος ιδιωτικού κλειδιού.	* Τρία επίπεδα ασφάλειας.
+ Γρήγορη δημιουργία κλειδιών.	-Μεγάλο μέγεθος υπογραφής.	* Σχετικά μεγάλο μέγεθος ιδιωτικού κλειδιού.
+ Γρήγορη επαλήθευση υπογραφών.		

Falcon		
Θετικά	**Αρνητικά**	**Ουδέτερα**
++ Γρήγορη υπογράφει εγγράφων.	- Δύο επίπεδα ασφάλειας.	* Σχετικά μεγάλο μέγεθος ιδιωτικού κλειδιού.
++ Γρήγορη επαλήθευση υπογραφών.	--- Απαιτεί αρκετός χρόνος για την δημιουργία των κλειδιών.	* Σχετικά μεγάλο μέγεθος δημόσιου κλειδιού.
++ Μικρό μέγεθος υπογραφής.		

SPHINCS$^+$		
Θετικά	**Αρνητικά**	**Ουδέτερα**
++ Μεγάλη δυνατότητα προσαρμοστικότητα (SPHINCS$^+_{slow,}$ SPHINCS$^+_{fast}$) για κάθε επίπεδο ασφάλειας .	---Παρά πολύ μεγάλο μέγεθος υπογραφής.	
+Γρήγορη επαλήθευση υπογραφών.	-Αργός χρόνος υπογραφής εγγράφων (SPHINCS$^+_{\mathbf{slow}}$).	
+ Γρήγορη υπογραφή εγγράφων.		
+ Γρήγορη δημιουργία κλειδιών.		
+++ Πάρα πολύ μικρό μέγεθος δημοσίου και ιδιωτικού κλειδιού.		

Όπως παρατηρούμε και στους πίνακες δεν υπάρχει ένας αλγόριθμος για όλες τις ανάγκες, η χρήση του καθενός εξαρτάται από τις ανάγκες της εκάστου εφαρμογής ασφάλειας.

Μετρώντας τον συνολικό χρόνο που ήθελε ο κάθε αλγόριθμος για να ολοκληρώσει το πείραμα μπορούμε να πάρουμε κάποια δεδομένα για την συνολική απόδοση του κάθε αλγόριθμου. Οι τιμές στον πίνακα παρουσιάζουν τον χρόνο που ήθελε κάθε αλγόριθμος για να παράγει το ζεύγος δημόσιου ιδιωτικού κλειδιού να υπογράψει ένα μήνυμα μεγέθους 100 bytes και να γίνει επαλήθευση της υπογραφής. Ο κύκλος αυτός επαναλαμβάνεται για 1000 νέα ζευγάρια δημόσιου-ιδιωτικού κλειδιού.

Αλγόριθμος	sec
Dilithium$_5$	212
Falcon$_{1024}$	19590
SPHINCS$^+{}_{256f}$	357
SPHINCS$^+{}_{256s}$	3284

Όπως παρατηρούμε ο πιο αποδοτικός αλγόριθμος είναι ο Dilithium. Ωστόσο ο πιο σημαντικός αργός είναι Falcon και οφείλεται στο χρόνο που απαιτείται για την παραγωγή των κλειδιών.

Επίλογος

Η αξία της ασύμμετρης κρυπτογράφησης ή κρυπτογράφησης δημοσίου κλειδιού είναι τεράστια. Το σύγχρονο Διαδίκτυο με την πληθώρα υπηρεσιών που προσφέρει δεν θα υπήρχε χωρίς τους αλγορίθμους που χρησιμοποιούνται για την κρυπτογράφηση ή την επιβεβαίωση των πληροφοριών. Η επιλογή του σωστού αλγόριθμου αποτελεί ζωτικής σημασίας για την εύρυθμη λειτουργία του δικτύου διότι πρωτόκολλα και υπηρεσίες πάνω στα οποία στηρίζονται όλες οι εφαρμογές που ανταλλάσσουν δεδομένα στο δίκτυο, δεν μπορούν να αλλάξουν εύκολα. Ένα κενό ασφάλειας ή κάποια λάθος υλοποίηση σε κάποιο από αυτά τα θεμελιώδη πρωτόκολλα συνεπάγεται πρόβλημα σε όλες τις άλλες υπηρεσίες, ενώ οι επιπτώσεις της επιλογής ακατάλληλου αλγόριθμου είναι σοβαρές. Η σοβαρότητα των επιλογών αυτών αποτελεί και τον λόγο που η αλλαγή των αλγορίθμων ασύμμετρης κρυπτογράφησης αποτελεί μια τόσο δύσκολη, χρονοβόρα και περιπλοκη διαδικασία.

Αναλύοντας την υπάρχουσα βιβλιογραφία μπορούμε να υποθέσουμε με σιγουριά την ανάγκη κατάργησης του RSA και του DH, αφού οι πλέον διαδεδομένοι αλγόριθμοι κρυπτογράφησης δημόσιου κλειδιού, δεν θα θεωρούνται ασφαλείς και μετά από τόσα χρόνια πρέπει να αντικαθιστούν. Η κοινότητα της κυβερνοασφάλειας καλείται πλέον να επιλέξει τους διαδόχους αυτών των αλγορίθμων. Υπάρχουν δύο ομάδες αλγορίθμων κρυπτογράφησης δημόσιου κλειδιού που προορίζονται για αυτή την θέση οι ECC που είδαμε στο Υποκεφάλαιο 2.5 και η PQCs που αναλύθηκαν στο τέταρτο κεφάλαιο. Πάρα τα πλεονεκτήματα και τα μειονεκτήματα της κάθε ομάδας η ειδοποιός διάφορα έγκειτι στην αντοχή των αλγορίθμων αυτών σε επιθέσεις από έναν κβαντικό υπολογιστή. Οι αλγόριθμοι βασιζόμενοι στα ECC δεν είναι πλήρως θωρακισμένοι από τέτοιες επιθέσεις ενώ οι PQCs αλγόριθμοι είναι πλήρως κατασκευασμένοι για να παρέχουν πλήρη θωράκιση. Οι απόψεις για το πραγματικό αντίκτυπο ενός QC στην κυβερνοασφάλεια διίστανται, κανένας δεν γνωρίζει επακριβώς πότε θα κατασκευαστεί ένας λειτουργικός QC που θα μπορούσε να αποτελέσει πραγματική απειλή.

Οι ECC αλγόριθμοι όπως είδαμε ξεκίνησαν να γίνονται ευρέως γνωστοί από το 2006 και

διάφορες εταιρίες που τους έχουν εφαρμόσει στις δικτυακές υπηρεσίες που προσφέρουν. Ωστόσο παραμένουν ακόμα τρωτοί σε επιθέσεις από έναν QC, για αυτό και διάφοροι οργανισμοί ξεκίνησαν project για την δημιουργία αλγορίθμων θωρακισμένων από ένα QC. Η Ευρωπαϊκή Ένωση πραγματοποίησε ένα ερευνητικό έργο για την δημιουργία PQC το οποίο είχε διάρκεια από το 2015 έως το 2018 το project είχε όνομα PQCRYPTO (Πηγή: Post-quantum cryptography for long-term security | PQCRYPTO | Project | Fact sheet | H2020 | CORDIS | European Commission (europa.eu)), ωστόσο δεν παρήγαγε κάποιο πρακτικό αποτέλεσμα. Το πιο σημαντικό project στο θέμα αυτό είναι του NIST. Διότι είχε ως αποτέλεσμα την δημιουργία των αλγορίθμων που αναλύθηκαν στο τέταρτο κεφάλαιο. Επίσης μπορεί οι αλγόριθμοι αυτοί να προορίζονται για υποχρεωτική εφαρμογή στην δημόσιες υπηρεσίες των ΗΠΑ, παρόλα αυτά οι περισσότερες αλλαγές που δρομολογούνται από τις ΗΠΑ στο Διαδίκτυο συνήθως επηρεάζουν με διάφορους τρόπους όλους του χρήστες. Ένας από τους στόχους της εργασίας ήταν η παρουσίαση και η ανάλυση αυτών των αλγορίθμων, ωστόσο είναι απολύτως αναγκαίο να πραγματοποιηθούν περαιτέρω έρευνες. Διότι υπάρχει πάντα η πιθανότητα ένας από τους αλγορίθμους να σπάσει όπως έγινε και με τον SIDH (Πηγή URL: Breaking SIDH in polynomial time (iacr.org)), ο οποίος ήταν ένας ακόμα PQC που πέρασε τον τρίτο γύρω του διαγωνισμού ωστόσο απορρίφθηκε μετά την δημοσίευση του paper. Το θέμα των μετά κβαντικών αλγορίθμων κρυπτογράφησης χρήζει περισσότερης μελέτης για πιθανά κενά ασφάλειας τόσο από κλασικές επιθέσεις όσο και επιθέσεις από κβαντικούς αλγορίθμους.

Αναφορές

[1] Zhiyong Zheng , Modern Cryptography Volume 2: A Classical Introduction 2022, ISBN-10: 9811909229, ISBN-13: 978-9811909221 , URL: Modern Cryptography Volume 2: Post-Quantum Cryptography - Free Computer, Programming, Mathematics, Technical Books, Lecture Notes and Tutorials (freecomputerbooks.com)

[2] Charles Pfleeger, Shari Pfleeger , Jonathan Margulies, Security in Computing 5th Edition 2015, ISBN-10: 9780134085043, ISBN-13: 978-0134085043

[3] Ευστάθιος Ζάχος, Αριστείδης Παγουρτζής, Παναγιώτης Γρόντας, Υπολογιστική Κρυπτογραφία 2015, ISBN: 978-960-603-276-9, URL: Υπολογιστική κρυπτογραφία | Δωρεάν βιβλία - Free ebooks (ebooks4greeks.gr)

[4] Hans Delfs , Helmut Knebl, Introduction to Cryptography Principles and Applications Third Edition 2015 , ISBN: 978-3-662-479 3-5

[5] National Academies of Sciences, Engineering and Medicine, Emily Grumbling , Mark Horowitz, Quantum Computing Progress and Prospects, ISBN: 978-0-309-47969-1, URL: Quantum Computing: Progress and Prospects | The National Academies Press

[6] David Adrian , Karthikeyan Bhargavan , Zakir Durumeric , Pierrick Gaudry , Matthew Green , J. Alex Halderman , Nadia Heninger , Drew Springall , Emmanuel Thomé , Luke Valenta , Benjamin VanderSloot , Eric Wustrow , Santiago Zanella-Béguelin , Paul Zimmermann, Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice 2015, URL: Imperfect Forward Secrecy | Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security

[7] Eric Landquist, The Quadratic Sieve Factoring Algorithm 2001, URL: QFS Simple.pdf (virginia.edu)

[8] An Introduction to the General Number Field Sieve, Matthew E. Briggs 1998, URL: briggs_gnfs_thesis.pdf (vt.edu)

[9] Neal Koblitz and Alfred Menezes, Another look at security definitions 2013, URL: Another look at security definitions (aimsciences.org)

[10] Philip Krantz, Morten Kjaergaard, Fei Yan, Terry P. Orlando, Simon Gustavsson, William D. Oliver, A Quantum Engineer's Guide to Superconducting Qubits 2019, URL: A quantum engineer's guide to superconducting qubits | Applied Physics Reviews | AIP Publishing

[11] Global Risk Institute, Dr. Michele Mosca, Dr. Marco Piani, 2021 Quantum Threat Timeline Report: Global Risk Institute 2022, URL: 2021 Quantum Threat Timeline Report: Global Risk Institute - Global Risk Institute

[12] Elisa Baumer, Jan-Grimo Sobez, Stefan Tessarini, Shor's Algorithm 2015, URL: Shors Algorithm.pdf (ethz.ch)

[13] C. Lavor, L.R.U. Manssur, R. Portugal, Shor's Algorithm for Factoring Large Integers 2003, [quant-ph/0303175] Shor's Algorithm for Factoring Large Integers (arxiv.org)

[14] Vikas Srivastava , Anubhab Baksi, Sumit Kumar Debnath, An overview of hash based signatures 2023, URL: An Overview of Hash Based Signatures (iacr.org)

[15] Mateusz Zych, Quantum Safe Cryptography Based on Hash Functions: A Survey 2018, URL: Master_thesis.pdf (uio.no)

[16] Lingyun Li, Xianhui Lu , Kunpeng Wang, Hash-based signature revisited 2022, URL: Hash-based signature revisited | Cybersecurity | Full Text (springeropen.com)

[17] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, Zooko Wilcox-O'Hearn, SPHINCS: Practical Stateless Hash-Based Signatures 2015, URL: SPHINCS: Practical

Stateless Hash-Based Signatures | SpringerLink
[18] Wenjuan Jia, Guanhao Xue, Baocang Wang, Yupu Hu, Module-LWE-Based Key Exchange Protocol Using Error Reconciliation Mechanism 2022,URL: Module-LWE-Based Key Exchange Protocol Using Error Reconciliation Mechanism (hindawi.com)
[19] Ziying Ni, Ayesha Khalid, Dur-e-Shahwar Kundi, Weiqiang Liu, HPKA: A High-Performance CRYSTALS-Kyber Accelerator Exploring Efficient Pipelining 2022, URL: HPKA: A High-Performance CRYSTALS-Kyber Accelerator Exploring Efficient Pipelining (iacr.org)
[20] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky,John M. Schanckk, Peter Schwabe, Gregor Seiler, Damien Stehlé, CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM 2017, URL: 634.pdf (iacr.org)
[21] Georg Maringer, Sven Puchinger, Antonia Wachter-Zeh, Information- and Coding-Theoretic Analysis of the RLWE/MLWE Channel 2022, URL: 2009.08681.pdf (arxiv.org)
[22] Lipeng Wan, Fangyu Zheng, Guang Fan,Rong Wei, Lili Gao, Yuewu Wang, Jingqiang Lin, Jiankuo Dong, A Novel High-performance Implementation of CRYSTALS-Kyber with AI Accelerator 2022, URL: A Novel High-performance Implementation of CRYSTALS-Kyber with AI Accelerator (iacr.org)
[23] Linus Backlund, Kalle Ngo, Joel Gartner, Elena Dubrova, Secret Key Recovery Attacks on Masked and Shuffled Implementations of CRYSTALS-Kyber and Saber, URL: Secret Key Recovery Attacks on Masked and Shuffled Implementations of CRYSTALS-Kyber and Saber (iacr.org)
[24] Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler and Damien Stehlé, CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation 2020, URL: Dilithium – Resources (pq-crystals.org)
[25] Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle, CRYSTALS - Dilithium: Digital Signatures from Module Lattices 2017, URL: CRYSTALS -- Dilithium: Digital Signatures from Module Lattices (iacr.org)
[26] Saad Islam, Koksal Mus, Richa Singh, Patrick Schaumont, Berk Sunar, Signature Correction Attack on Dilithium Signature Scheme 2022, URL: 2203.00637.pdf (arxiv.org)
[27] Soundes Marzougui, Vincent Ulitzsch, Mehdi Tibouchi, Jean-Pierre Seifert, Profiling Side-Channel Attacks on Dilithium: A Small Bit-Fiddling Leak Breaks It All 2022, URL: Profiling Side-Channel Attacks on Dilithium: A Small Bit-Fiddling Leak Breaks It All (iacr.org)
[28] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU, URL: Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU (falcon-sign.info)
[29] Kristian Gjøsteen, Instantiating the GPV-framework 2022,
URL: no.ntnu:inspera:103848036:48690051.pdf
[30] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte6 Zhenfei Zhang, Falcon 2018, Falcon (nist.gov)
[31] Mihir Bellare, Igors Stepanovs, Security Under Message-Derived Keys: Signcryption in iMessage 2020, URL: Security Under Message-Derived Keys: Signcryption in iMessage | SpringerLink
[32] Morgane Guerreau, Ange Martinelli,Thomas Ricosset, Mélissa Rossi, The Hidden Parallelepiped Is Back Again: Power Analysis Attacks on Falcon 2022, URL: The Hidden Parallelepiped Is Back Again: Power Analysis Attacks on Falcon (d-nb.info)
[33] ALVARO CINTAS CANTO, JASMIN KAUR, MEHRAN MOZAFFARI KERMANI, REZA AZARDERAKHSH, Algorithmic Security is Insufficient: A Comprehensive Survey on Implementation Attacks Haunting Post-Quantum Security 2023, URL: arXiv:2305.13544v1 [cs.CR] 22 May 2023

[34] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, Damien Stehlé, CRYSTALS-Kyber Algorithm Specifications And Supporting Documentation (version 3.0) 2020, URL: Kyber – Resources (pq-crystals.org)

[35] E.O. Kiktenko, A.A. Bulychev, P.A. Karagodin, N.O. Pozhar, M.N. Anufriev, A.K. Fedorov, SPHINCS+ post-quantum digital signature scheme with Streebog hash function 2020, URL: 1904.06525.pdf (arxiv.org)

[36] Mahmoud Yehia, Riham AlTawy, T. Aaron Gulliver, Hash-based Signatures Revisited: A Dynamic FORS with Adaptive Chosen Message Security 2020, URL: Hash-based Signatures Revisited: A Dynamic FORS with Adaptive Chosen Message Security (iacr.org)

[37] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe, The SPHINCS⁺ Signature Framework, URL: The SPHINCS+ Signature Framework (iacr.org)

[38] Yang Li, Kee Siong Ng, Michael Purcell, A TUTORIAL INTRODUCTION TO LATTICE-BASED CRYPTOGRAPHY AND HOMOMORPHIC ENCRYPTION 2022, URL: 2208.08125.pdf (arxiv.org)

[39] RUMAISA NIAZI, Introduction to Digital Signature Algorithm (DSA), Make Us OF, MAR 11 2022, URL: Introduction to Digital Signature Algorithm (DSA) (makeuseof.com)

[40] Nick Sullivan, A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography, GloudFlare, 24/10/2013, URL: A (Relatively Easy To Understand) Primer on Elliptic Curve Cryptography (cloudflare.com)

[41] Michael Cobb, RSA algorithm (Rivest-Shamir-Adleman), TechTarget, November 2021, URL: What is the RSA algorithm? Definition from SearchSecurity (techtarget.com)

[42] Simplilearn, What Is RSA Algorithm and How Does It Work in Cryptography?, Feb 13, 2023, URL: What Is RSA Algorithm In Cryptography? | Simplilearn

[43] David, Fault attacks on RSA's signatures, ZKSECURITY, September 2016, URL: Fault attacks on RSA's signatures (cryptologie.net)

[44] SAVVY SECURITY, What Is the RSA Algorithm? A Look at RSA Encryption, JUNE 24 2022, URL: What Is the RSA Algorithm? A Look at RSA Encryption (cheapsslsecurity.com)

[45] Rebecca Bales, The Complete Guide to Public Key Cryptography, History-Computer, August 1, 2023, URL: Public Key Cryptography: The Complete Guide - History-Computer

[46] Baivab Kumar Jena, Digital Signature Algorithm (DSA) in Cryptography: How It Works & More, Simplilearn, Feb 14, 2023, URL: Digital Signature Algorithm (DSA) in Cryptography: A Complete Guide | Simplilearn

[47] JOSH LAKE, Demystifying Diffie-Hellman key exchange and explaining how it works, comparitech, August 24 2023, URL: What is the Diffie–Hellman key exchange and how does it work? (comparitech.com)

[48] Medium, What is Lattice-Based Cryptography & Why You Should Care, Jun 16 2018,URL: What is Lattice-Based Cryptography & Why You Should Care | by Wickr | Wickr Crypto + Privacy Blog | Medium

[49] Robert Relyea, Post-quantum cryptography: Hash-based signatures, RedHat, October 27 2022, URL: Post-quantum cryptography: Hash-based signatures (redhat.com)

[50] Ruben Gonzalez, Kyber - How does it work?, Approachable Cryptography, 2021-09-14, URL: Kyber - How does it work? | Approachable Cryptography (cryptopedia.dev)

[51] Elena Bakos Lang, Building Intuition for Lattice-Based Signatures – Part 2: Fiat-Shamir with Aborts,nccgroup, August 17, 2023 ,Building Intuition for Lattice-Based Signatures – Part 2: Fiat-Shamir with Aborts | NCC Group Research Blog | Making the world safer and more secure

[52] Goutam Tamvada,Sofía Celi, Deep dive into a post-quantum signature scheme,

CLOUDFLARE, 22/02/2022, URL: Deep dive into a post-quantum signature scheme (cloudflare.com)

[53] Eric Zhang, VegeBun, How do hash-based post-quantum digital signatures work?, DORA RESEARCH BLOG, 26 Oct 2022, URL: How do hash-based post-quantum digital signatures work? (Part 1) · Eric Zhang, VegeBun · Dora Research Blog (dorahacks.io)

[54] David Wong, Hash-Based Signatures Part I: One-Time Signatures (OTS), Cryptography Services NCC Group, Dec 4 2015, URL: Hash-Based Signatures Part I: One-Time Signatures (OTS) (cryptoservices.github.io)

[55] Elaine Barker; Allen Roginsky, Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths, 11/06/15, URL: Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths (nist.gov)

Αποθετήριο κώδικα

1. Κώδικας δοκίμων για τα υποκεφάλαια 4.5.3, 4.6.3 & 4.7.5

```
import falcon
from dilithium import Dilithium3
import  pyspx.sha2_128f as sphinc
import os,binascii


msg=b"geraldo1234"


#FALCON
"""
sk = falcon.SecretKey(1024)
pk = falcon.PublicKey(sk)
sig = sk.sign(msg)

"""

#DILITHIUM
"""
pk, sk = Dilithium3.keygen()
sig = Dilithium3.sign(sk, msg)

"""
#SPHINCS+
"""
seed=os.urandom(sphinc.crypto_sign_SEEDBYTES)
pk,sk=sphinc.generate_keypair(seed)
sig=sphinc.sign(msg,sk)

"""

print("\n ================== user public key ================== \n")
print(pk.hex())

print("\n ==================  user secret key ================== \n")
print(sk.hex())

print("\n ==================  user hash ================== \n")
print(sig.hex())


#FALCON
"""
print("\nVerify the user =",pk.verify(msg, sig),'\n')
sk_2 = falcon.SecretKey(512)
pk_2 = falcon.PublicKey(sk_2)
print("Verify the user with wrong per of keys = ", pk_2.verify(msg, sig),'\n')

"""

#DILITHIUM
"""
print("\n Verify the user =",Dilithium3.verify(pk, msg, sig))
pk_2, sk_2 = Dilithium3.keygen()
print("\n Verify the user with wrong per of keys = ", Dilithium3.verify(pk_2,
```

```
msg, sig))

"""

#SPHINCS+
"""

print("\nVerify the user =",sphinc.verify(msg,sig,pk),'\n')
pk_2,sk_2 = sphinc.generate_keypair(seed)
print("Verify the user with wrong per of keys = ",
sphinc.verify(msg,sig,pk_2),'\n')

"""

msg_2=b"aliaj1234"


#FALCON
"""
print("Verify the user with wrong message = ",pk.verify(msg_2, sig))

"""

#DILITHIUM
"""
print("\n Verify the user with wrong message = ", Dilithium3.verify(pk, msg_2,
sig))

"""

#SPHINCS+
"""
  print("Verify the user with wrong message = ",sphinc.verify(msg_2,sig,pk))

  """
```

2. Κώδικας δοκίμων για το υποκεφάλαιο 4.6.5 & 4.7.5

```
from dilithium import Dilithium5
import falcon
import  pyspx.sha2_256s as sphinc

import os
import time
import matplotlib.pyplot as plt

range_test=1000
algorithm=''
msg = b"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
            aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"

count_generation_time=0
count_siging_time=0
count_verifying_time=0

star_experiment=time.time()

key_generation_time=[]
```

```
message_siging_time=[]
message_verifying_time=[]

#sphincs with the same seed


for i in range(range_test):

 #key generation test

 start = time.time()

 #DILITHIUM
 """
 pk, sk = Dilithium5.keygen()
 end = time.time()

 """

 #FALCON
 """
 sk = falcon.SecretKey(falcon_size)
 pk = falcon.PublicKey(sk)

 """

 #SPHINCS+
 """
 seed=os.urandom(sphinc.crypto_sign_SEEDBYTES)
 pk,sk=sphinc.generate_keypair(seed)

 """

 end=time.time()
 key_generation_time.append(end - start)
 count_generation_time=count_generation_time+(end - start)

 #message siging test

 start = time.time()

 #DILITHIUM
 """
 sig = Dilithium5.sign(sk, msg)

 """
 #FALCON
 """
 sig = sk.sign(msg)

 """

 #SPHINCS+
 """
 sig=sphinc.sign(msg,sk)

 """


 end = time.time()

```

```
 82  message_siging_time.append(end - start)
 83  count_siging_time=count_siging_time+(end - start)
 84
 85
 86  #message verifying test
 87
 88  start = time.time()
 89
 90  #DILITHIUM
 91  """
 92  k=Dilithium5.verify(pk, msg, sig)
 93
 94  """
 95
 96  #FALCON
 97  """
 98  k=pk.verify(msg, sig)
 99
100  """
101  #SPHINCS+
102  """
103  k=sphinc.verify(msg,sig,pk)
104
105  """
106
107  end = time.time()
108
109  message_verifying_time.append(end - start)
110  count_verifying_time=count_verifying_time+(end - start)
111
112  print(i)
113
114 #end for
115
116 end_experiment=time.time()
117
118 print("/n ====================== Caclulate the averange runing times
119 ======================\n")
120 print("Averange key generation time for ",algorithm,"=
121 ",count_generation_time/range_test)
122 print("Averange message siging time for ",algorithm,"=
123 ",count_siging_time/range_test)
124 print("Averange message verifying time for",algorithm,"=
125 ",count_verifying_time/range_test)
126 print("\n")
127 print("Experiment time= ",end_experiment-star_experiment)
128 print("/n ====================== end ======================== \n")
129
130 #Plot histograms
131
132 plt.hist(key_generation_time)
133 plt.xlabel('ms')
134 title="Histogram for "+algorithm+" key generation time"
135 plt.title(title)
136 plt.show()
137
138
139 plt.hist(message_siging_time)
140 plt.xlabel('ms')
141 title="Histogram for "+algorithm+" message signing time"
142 plt.title(title)
```

```
143 plt.show()
144
145
146 plt.hist(message_verifying_time)
    plt.xlabel('ms')
    title="Histogram for "+algorithm+" message verifying time"
    plt.title(title)
    plt.show()
```