Δημοκριτειο Πανεπιστημιο Θρακης
Τμημα ηλεκτρολογων μηχανικων & μηχανικων υπολογιστων

Ασφαλεια Συστηματων Υπολογιστων

Κρυπτογραφικός αλγόριθμος **RSA**

Φοιτητές
Εμμανουηλίδης Κωνσταντίνος
57315
Μακρίδης Βασίλειος
57332
Μασλάρης Ιωάννης-Αριστείδης
57348
Χατζηπαπάς Νικόλαος
57370

Καθηγητής
Τσουλουχάς Γεώργιος

Ιανουάριος 2021

Περιεχόμενα

1 Κρυπτογραφία

1.1 Εισαγωγή στην κρυπτογραφία

Η κρυπτογραφία είναι η μελέτη και η εφαρμογή των τεχνικών για ασφαλή επικοινωνία παρουσία τρίτων, ονομαζόμενοι ως αντίπαλοι. Ασχολείται με την ανάπτυξη και ανάλυση πρωτοκόλλων που εμποδίζουν κακόβουλα τρίτα μέρη να ανακτούν πληροφορίες που μοιράζονται μεταξύ δύο οντοτήτων ακολουθώντας έτσι τις διάφορες πτυχές της ασφάλειας των πληροφοριών.

Πριν από τη σύγχρονη εποχή η κρυπτογραφία ήταν ουσιαστικά συνώνυμη με την απλή κρυπτογράφηση, μετατρέποντας τις πληροφορίες από μια αναγνώσιμη κατάσταση σε ακατανόητη μορφή χωρίς άμεση ουσία. Ο αποστολέας ενός κρυπτογραφημένου μηνύματος μοιράζεται την τεχνική αποκωδικοποίησης μόνο με τους προοριζόμενους παραλήπτες για να αποκλείσει την πρόσβαση από τους αντιπάλους. Από την ανάπτυξη των μηχανών κρυπτογράφησης στον Α' Παγκόσμιο Πόλεμο και την έλευση των υπολογιστών στον Β' Παγκόσμιο Πόλεμο, οι μέθοδοι κρυπτογράφησης έγιναν όλο και πιο περίπλοκες με ποικίλες εφαρμογές που τις βλέπουμε καθημερινά παντού γύρω μας. Η σύγχρονη κρυπτογραφία βασίζεται σε μεγάλο βαθμό στη μαθηματική θεωρία και τις πρακτικές της επιστήμης των υπολογιστών.

Οι κρυπτογραφικοί αλγόριθμοι σχεδιάζονται γύρω από υποθέσεις υπολογιστικής πολυπλοκότητας, καθιστώντας αυτούς τους αλγόριθμους δύσκολο να "σπάσουν" σε πραγματικές συνθήκες από οποιονδήποτε αντίπαλο. Αν και θεωρητικά είναι δυνατό να εισέλθουμε σε ένα τέτοιο, καλά σχεδιασμένο σύστημα, στην πράξη αυτό είναι πολύ δύσκολο έως αδύνατο να γίνει. Αυτά τα συστήματα θεωρούνται ασφαλή, αν και οι συνεχείς εξελίξεις στην ταχύτητα των υπολογιστών και στους αλγορίθμους παραγοντοποίησης ακραίων απαιτούν αυτά τα συστήματα να επαναξιολογούνται συνεχώς και, εάν είναι απαραίτητο, να προσαρμόζονται.

Κάθε κρυπτογραφικό σύστημα θα πρέπει να αποσκοπεί στις εξής τέσσερις βασικές λειτουργίες, στην εμπιστευτικότητα, την ακεραιότητα, την μη απάρνηση και την πιστοποίηση.Ο στόχος της εμπιστευτικότητας αναφέρεται στο γεγονός ότι η πληροφορία προς μετάδοση είναι προσβάσιμη μόνο στα εξουσιοδοτημένα μέλη και ακατανόητη από τρίτους. Η ακεραιότητα επιβάλει αλλοίωση της πληροφορίας να γίνεται μόνο από τα εξουσιοδοτημένα μέλη. Η μη απάρνηση εκφράζει πως ο αποστολέας ή ο παραλήπτης της πληροφορίας δεν μπορεί να αρνηθεί την αυθεντικότητα της μετάδοσης ή της δημιουργίας της. Τέλος η πιστοποίηση ορίζει ότι οι αποστολέας και παραλήπτης μπορούν να εξακριβώνουν τις ταυτότητές τους καθώς και την πηγή και τον προορισμό της πληροφορίας με διαβεβαίωση ότι οι ταυτότητές τους δεν είναι πλαστές.

Σημαντικό κομμάτι της κρυπτογραφίας αποτελεί η ορολογία της. Γνωρίζοντας την, κάποιος μπορεί εύκολα να κατανοήσει κρυπτογραφικούς αλγορίθμους και συστήματα. Οι βασικές τις έννοιες εξηγούνται παρακάτω.

- Κρυπτογράφηση (encryption): Η διαδικασία μετασχηματισμού ενός μηνύματος σε μία ακατανόητη μορφή με τη χρήση κάποιου κρυπτογραφικού αλγορίθμου ούτως ώστε να μην μπορεί να διαβαστεί από κανέναν εκτός του νόμιμου παραλήπτη. Η αντίστροφη διαδικασία όπου από το κρυπτογραφημένο κείμενο παράγεται το αρχικό μήνυμα ονομάζεται αποκρυπτογράφηση (decryption).

- Κρυπτογραφικός αλγόριθμος (cipher): Η μέθοδος μετασχηματισμού δεδομένων σε μία μορφή που να μην επιτρέπει την αποκάλυψη των περιεχομένων τους από μη εξουσιοδοτημένα μέρη. Κατά κανόνα ο κρυπτογραφικός αλγόριθμος είναι μία πολύπλοκη μαθηματική συνάρτηση.

- Αρχικό κείμενο (plaintext): Το μήνυμα το οποίο αποτελεί την είσοδο σε μία διεργασία κρυπτογράφησης.

- Κλειδί (key): Ένας αριθμός αρκετών bit που χρησιμοποιείται ως είσοδος στη συνάρτηση κρυπτογράφησης.

- Κρυπτογραφημένο κείμενο (ciphertext): Είναι το αποτέλεσμα της εφαρμογής ενός κρυπτογραφικού αλγόριθμου πάνω στο αρχικό κείμενο.
- Κρυπτανάλυση (cryptanalysis): Είναι μία επιστήμη που ασχολείται με το "σπάσιμο" κάποιας κρυπτογραφικής τεχνικής ούτως ώστε χωρίς να είναι γνωστό το κλειδί της κρυπτογράφησης, το αρχικό κείμενο να μπορεί να αποκωδικοποιηθεί.

1.2 Κρυπτοσυστήματα

Ένα κρυπτογραφικό σύστημα αποτελείται από πέντε βασικά χαρακτηριστικά (P,C,K,E,D).

- P : Το σύνολο των αποδεκτών μη κρυπτογραφημένων συμβολοσειρών.
- C : Το σύνολο των κρυπτογραφημένων συμβολοσειρών.
- K : Το σύνολο των κλειδιών που θεωρούμε.
- E : Το σύνολο των συναρτήσεων κρυπτογράφησης.
- D : Το σύνολο των συναρτήσεων αποκρυπτογράφησης.

Κατά την διαδικασία της κρυπτογράφησης, το αρχικό κείμενο (plain text), που ανήκει στο σύνολο P, μέσω ενός κρυπτογραφικού αλγορίθμου (chiper) μετασχηματίζεται σε μία νέα ακατανόητη μορφή, το κρυπτογραφημένο κείμενο (ciphertext). Στα μοντέρνα κρυπτογραφικά συστήματα ο αλγόριθμος αυτός αποτελεί μία σύνθετη μαθηματική συνάρτηση.

1.2.1 Συμμετρική κρυπτογραφία

Στα συστήματα συμμετρικής κρυπτογραφίας (Symmetric Cryptography ή Secret-key Cryptography) ο αποστολέας και ο παραλήπτης χρησιμοποιούν ένα κλειδί, το λεγόμενο μυστικό κλειδί (secret key). Ο αποστολέας πριν την αποστολή του μηνύματος το κρυπτογραφεί χρησιμοποιώντας το μυστικό κλειδί. Ο παραλήπτης λαμβάνει το κρυπτοκείμενο, δηλαδή το μήνυμα σε κρυπτογραφημένη μορφή, και χρησιμοποιώντας το ίδιο μυστικό κλειδί το αποκρυπτογραφεί. Αλγόριθμοι συμμετρικής κρυπτογραφίας χρησιμοποιούνται για κρυπτογράφηση ενώ επίσης και για πιστοποίηση ταυτότητας (MAC). Ένα προφανές μειονέκτημα των αλγορίθμων συμμετρικής κρυπτογραφίας είναι ότι ο παραλήπτης και ο αποστολέας πρέπει να συνεννοούνται για το μυστικό κλειδί. Φυσικά αυτή η συνεννόηση θα πρέπει να γίνεται χωρίς κάποιον άλλο να λάβει γνώση αυτού. Εκεί όμως που έχουν πλεονέκτημα αυτοί οι αλγόριθμοι είναι στην ταχύτητα. Γνωστοί συμμετρικοί κρυπτογραφικοί αλγόριθμοι είναι ο AES, ο BlowFish και ο DES.

Στα συμμετρικά συστήματα κρυπτογραφίας είναι απαραίτητη η ύπαρξη ενός ισχυρού αλγορίθμου κρυπτογράφησης. Ελάχιστη απαίτηση θεωρείται η ύπαρξη αλγορίθμου τέτοιος ώστε ακόμη και αν είναι γνωστός σε κάποιον επιτιθέμενο, ο οποίος έχει πρόσβαση και σε ένα σύνολο από κρυπτογραφήματα, να μην είναι σε θέση ούτε να υπολογίσει το μυστικό κλειδί αλλά ούτε να συμπεράνει το αρχικό κείμενο, δηλαδή να κρυπτοαναλύσει το κρυπτογράφημα. Επίσης ο πομπός και ο δέκτης κάθε καναλιού επικοινωνίας θα πρέπει να παραλαμβάνουν από ένα αντίγραφο του μυστικού κλειδιού με ασφαλή τρόπο, ενώ ταυτόχρονα το κλειδί θα πρέπει να αποθηκεύεται σε ασφαλές μέρος. Εάν κάποιος τρίτος που γνωρίζει τον αλγόριθμο αποκτήσει γνώση και για το μυστικό κλειδί τότε κάθε μήνυμα που κωδικοποιείται με αυτό το κλειδί είναι αναγνώσιμο, άρα παραβιάζεται η εμπιστευτικότητα.

1.2.2 Ασύμμετρα κρυπτογραφικά συστήματα

Εξίσου σημαντική σημασία με την συμμετρική κρυπτογραφία παρουσιάζει και η ασύμμετρη κρυπτογραφία. Η ασύμμετρη κρυπτογραφία ή αλλιώς κρυπτογραφία δημόσιου κλειδιού (Public key cryptography) προτάθηκε το 1976 από του W. Diffie και M. Hellman. Οι αλγόριθμοι κρυπτογραφίας δημόσιου κλειδιού βασίζονται σε μαθηματικές συναρτήσεις και όχι σε απλές πράξεις με bits. Σε αυτό το είδος κρυπτογραφίας χρησιμοποιείται ένα ζεύγος κλειδιών για την κρυπτογράφηση και αποκρυπτογράφηση, σε αντίθεση με τη συμμετρική που χρησιμοποιεί μόνο ένα κλειδί. Το ζεύγος κλειδιών που έχει στην κατοχή του κάθε χρήστης αποτελείται από το δημόσιο κλειδί (public key) και το ιδιωτικό κλειδί (private key). Το δημόσιο κλειδί κάθε χρήστη είναι προσβάσιμο για κάθε άλλο χρήστη του συστήματος, ενώ το ιδιωτικό κλειδί παραμένει γνωστό μόνο για τον εκάστοτε χρήστη για τον οποίο παράχθηκε. Όλες οι επικοινωνίες βασίζονται στο δημόσιο κλειδί και το ιδιωτικό δεν μεταδίδεται ποτέ. Έτσι εξαφανίζεται η ανάγκη αποστολέας και παραλήπτης να μοιράζονται το ίδιο κλειδί. Τα κλειδιά βρίσκονται σε απόλυτη αντιστοιχία μεταξύ τους, αυτό σημαίνει ότι κάθε φορά που ένας χρήστης επιθυμεί να τα ανανεώσει πρέπει να το κάνει τόσο για το ιδιωτικό όσο και για το δημόσιο κλειδί ταυτόχρονα. Αυτό αποτελεί και την μόνη απαίτηση της ασύμμετρης κρυπτογραφίας, η επιβεβαιωμένη συσχέτιση του ιδιωτικού και του δημόσιου κλειδιού κάθε χρήστη. Έτσι καθίσταται αδύνατη η σκόπιμη ή μη πλαστοπροσωπία. Η συσχέτιση των δύο κλειδιών είναι μαθηματική, άρα τεχνικά είναι δυνατόν ένα τέτοιο κρυπτοσύστημα να "σπάσει" ανακτώντας το ιδιωτικό κλειδί από το δημόσιο. Ωστόσο για καλά σχεδιασμένα συστήματα η διαδικασία ανάκτησης κλειδιών είναι αρκετά χρονοβόρα ώστε να καθίσταται μη πρακτική.

Εδώ ας δούμε μερικές παρεξηγήσεις σχετικά με την κρυπτογράφησης δημόσιου κλειδιού. Πρώτη και κυριότερη εσφαλμένη αντίληψη αφορά στο ότι τα κρυπτογραφικά συστήματα δημοσίου κλειδιού είναι πιο ασφαλή σχετικά με τα συμμετρικά συστήματα κρυπτογράφησης. Η χρήση ενός ζεύγους κλειδιών για κάθε επικοινωνία κάνει πολλούς να πιστεύουν ότι αυτά είναι πιο ανθεκτικά σε επιθέσεις κρυπτοανάλυσης. Όμως η ασφάλεια κάθε κρυπτογραφικού συστήματος, συμμετρικού ή ασύμμετρου, είναι ανάλογη του μεγέθους του κλειδιού που χρησιμοποιείται και της υπολογιστικής ισχύς που χρησιμοποιείται από τον αντίπαλο. Μία δεύτερη λανθασμένη αντίληψη έρχεται σαν συνέχεια με την πρώτη και αφορά την επικράτηση των ασύμμετρων κρυπτογραφικών συστημάτων. Και τα δύο είδη κρυπτογραφίας χρησιμοποιούνται εξίσου στις διάφορες εφαρμογές και δεν φαίνεται μελλοντικά η εγκατάλειψη ενός εκ των δύο. Αυτό είναι λογικό αν κανείς δει την χρονική επιβάρυνση που απαιτείτε για την εκτέλεση των λειτουργιών ενός ασύμμετρου κρυπτοσυστήματος. Ακόμη υπάρχει η άποψη ότι η διανομή κλειδιών είναι πιο εύκολη στα συστήματα ενός κλειδιού απ' ότι στα συστήματα ζεύγους κλειδιών. Στην πραγματικότητα όμως στα ασύμμετρα συστήματα απαιτείται εκτέλεση κάποιων πρωτοκόλλων, ύπαρξη ενός έμπιστου ενδιάμεσου αντιπροσώπου ενώ οι διαδικασίες που περιλαμβάνονται δεν είναι απλούστερες ή πιο αποδοτικές από αυτές που εκτελούνται για την συμμετρική κρυπτογράφηση.

Οι τεχνικές κρυπτογραφίας δημοσίου κλειδιού βρίσκουν εφαρμογή στις εξής τρεις υπηρεσίες. Στην κρυπτογράφηση/αποκρυπτογράφηση, στην ψηφιακή υπογραφή και την ανταλλαγή κλειδιών. Η διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης, όπως αναφέρθηκε και προηγουμένως, αφορούν την κρυπτογράφηση ενός μηνύματος στην πλευρά του αποστολέα με το δημόσιο κλειδί του παραλήπτη και την αποκρυπτογράφηση του μηνύματος αυτού στην μεριά του παραλήπτη με το ιδιωτικό κλειδί του. Κατά την διαδικασία της ψηφιακής υπογραφής ο αποστολέας υπογράφει ένα μήνυμα χρησιμοποιώντας το ιδιωτικό κλειδί του. Η υπογραφή δημιουργείται με την χρήση ενός αλγορίθμου κρυπτογράφησης στο μήνυμα ή συνηθέστερα στη σύνοψη (hash) του μηνύματος. Τέλος γίνεται αυθεντικοποίηση του αποστολέα από τον παραλήπτη με χρήση του δημόσιου κλειδιού του. Μερικοί από τους πιο διαδεδομένους αλγορίθμους κρυπτογράφησης δημοσίου κλειδιού είναι ο Diffie-Hellman, ο DSS και ο RSA τον οποίο θα αναλύσουμε εκτενώς στην συνέχεια.

2 Αλγόριθμος RSA

2.1 Εισαγωγή

Αναπτύχθηκε από τους Rivest, Shamir, Adleman, απ' όπου πήρε και το όνομά του. Δημοσιεύτηκε για πρώτη φορά το 1978 και από εκείνη την στιγμή αποτελεί χρυσή σταθερά στα ασύμμετρα κρυπτοσυστήματα. Βασίζεται στις αρχές της θεωρίας των αριθμών και η ασφάλειά του έγκειται στη δυσκολία της παραγοντοποίησης ενός σύνθετου ακεραίου, εφόσον δεν έχει αναπτυχθεί ακόμα ένας αλγόριθμος που μπορεί αποδοτικά σε πολυωνυμικό χρόνο να παραγοντοποιεί έναν ακέραιο. Οι εφαρμογές του παραμένουν πολλές ακόμη και σήμερα. Χρησιμοποιείται ευρύτατα σε εφαρμογές που απαιτούν μεγάλη ασφάλεια όπως συναλλαγές μέσω ίντερνετ. Είναι ικανός αλγόριθμος για ασφαλή κρυπτογράφηση, μεταβίβαση κλειδιών μικρού μεγέθους και δημιουργία ψηφιακών υπογραφών.

Ο αλγόριθμος RSA είναι ένας μη συμμετρικός αλγόριθμος, πράγμα που σημαίνει ότι δύο κλειδιά (δημόσιο κλειδί και ιδιωτικό κλειδί) εμπλέκονται στις διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης. Το δημόσιο κλειδί μπορεί να είναι γνωστό σε όλους τους χρήστες και χρησιμοποιείται για την κρυπτογράφηση του αρχικού κειμένου σε κρυπτογραφημένο κείμενο πριν την αποστολή του. Το μήνυμα που κρυπτογραφείται με ένα δημόσιο κλειδί μπορεί να αποκρυπτογραφηθεί μόνο χρησιμοποιώντας το ιδιωτικό κλειδί που αντιστοιχεί στο δημόσιο κλειδί με το οποίο έγινε η κρυπτογράφηση. Η διαδικασία με την οποία παράγονται τα κλειδιά είναι που κάνει τον αλγόριθμο τόσο ασφαλή και αξιόπιστο μέχρι σήμερα.

2.2 Λειτουργία του αλγορίθμου

Ας δούμε τον γενικό μηχανισμό του αλγορίθμου με το εξής παράδειγμα. Έστω ότι ο Bob θέλει να στείλει ένα μήνυμα στην Alice. Ο Bob και η Alice, για να διασφαλίσουν ασφάλεια και μυστικότητα κατά την επικοινωνία τους, χρησιμοποιούν ένα κρυπτογραφικό σύστημα RSA. Σαν χρήστες του συστήματος έχουν ο καθένας από ένα δημόσιο και ένα ιδιωτικό κλειδί, τα οποία παράγονται από τον αλγόριθμο του συστήματος και τους αναθέτονται. Και οι δύο κρατάνε τα ιδιωτικά τους κλειδιά στις συσκευές τους (υπολογιστής, κινητό τηλέφωνο), ενώ τα δημόσια κλειδιά τους βρίσκονται στη βάση δεδομένων του κρυπτογραφικού συστήματος. Τα ιδιωτικά τους κλειδιά τα γνωρίζουν μόνο οι ίδιοι ενώ καθένας μπορεί να δει το δημόσιο κλειδί οποιουδήποτε χρήστη. Καθώς ο Bob έχει ετοιμάσει στη συσκευή το μήνυμά πατάει αποστολή. Προτού το μήνυμα εισέλθει στο δίκτυο κρυπτογραφείται σύμφωνα με το δημόσιο κλειδί της Alice. Η κρυπτογραφημένη έκδοση του μηνύματος εισέρχεται λοιπόν στο δίκτυο με προορισμό την Alice. Η συσκευή της Alice λαμβάνει το μήνυμα του Bob και το αποκρυπτογραφεί χρησιμοποιώντας το ιδιωτικό κλειδί της. Η Alice το διαβάζει και απαντάει στον Bob με ένα δεύτερο μήνυμα. Πριν την αποστολή του, το μήνυμα κρυπτογραφείται σύμφωνα με το δημόσιο κλειδί του Bob. Το κρυπτογραφημένο μήνυμα αποστέλλεται στον Bob. Κατά τον ίδιο τρόπο η συσκευή του Bob λαμβάνει το μήνυμα της Alice και το αποκρυπτογραφεί χρησιμοποιώντας το ιδιωτικό κλειδί του Bob. Ο Bob διαβάζει το μήνυμα και η κουβέντα μεταξύ του Bob και της Alice συνεχίζεται. Έστω τώρα ότι κάποιος τρίτος χρήστης καταφέρνει και υποκλέπτει ένα μήνυμα από την κουβέντα. Το μήνυμα φθάνει στον υπολογιστεί του σε κρυπτογραφημένη μορφή. Ο τρίτος αυτός χρήστης έχει γνώση του δικού του ιδιωτικού κλειδιού καθώς και των δημοσίων κλειδιών όλων των χρηστών του συστήματος, συμπεριλαμβανομένων του Bob και της Alice. Αυτή η γνώση όμως δεν είναι αρκετή για να αποκρυπτογραφήσει το μήνυμα. Κάθε χρήστης μπορεί να αποκρυπτογραφεί μηνύματα που απευθύνονται στον ίδιο χρησιμοποιώντας το ιδιωτικό κλειδί του. Θεωρητικά ο τρίτος χρήστης που έχει υποκλέψει το μήνυμα θα μπορούσε να το αποκρυπτογραφίσει. Όμως δεδομένου ότι το μέγεθος των κλειδιών είναι αρκετά μεγάλο, η διαδικασία αποκρυπτογράφησης του μηνύματος είναι μη πρακτική αφού στην καλύτερη περίπτωση θα χρειαζόντουσαν εκατοντάδες χρόνια για να επιτευχθεί με τις σημερινές μεθόδους.

Η λογική και η λειτουργία του αλγορίθμου RSA βασίζεται στην αριθμητική υπολοίπων. Αναφέρουμε μερικά χαρακτηριστικά στης παρακάτω.

Όταν διαιρούμε ένα αριθμό θα έχουμε μία εξίσωση της παρακάτω μορφής:

$$A/B \ = \ Q \ με \ υπολοιπο \ R$$

Πολλές φορές μας ενδιαφέρει μόνο το υπόλοιπο της διαίρεσης. Σε αυτήν την περίπτωση ο τελεστής καλείται modulo (εν συντομία mod). Άρα, όταν απλά μας νοιάζει το υπόλοιπο η παραπάνω εξίσωση γίνεται

$$A \, mod \, B \ = \ R$$

Η αριθμητική υπολοίπων μπορεί να αντιμετωπιστεί με την εισαγωγή μίας σχέσης ισοτιμίας στους ακέραιους αριθμούς. Για έναν θετικό ακέραιο n, δύο ακέραιο a και b είναι ισοδύναμοι Modulo n και γράφουμε:

$$a \ = \ b \, (mod \, R)$$

Επίσης λέμε ότι οι αριθμοί A και B είναι στην ίδια κλάση ισοτιμίας. Ας δούμε το εξής παράδειγμα. Έχουμε τα εξής:

$$26 \ = \ 11 \, (mod \, 5)$$

Αυτό σημαίνει ότι:

$$26 \, mod \, 5 \ = \ 1, \ ρα \ το \ 26 \ εναι \ στην \ κλση \ ισοτιμας \ 1$$

$$11 \, mod \, 5 \ = \ 1, \ ρα \ το \ 11 \ εναι \ επσης \ στην \ κλση \ ισοτιμας \ 1$$

Πράξη πρόσθεσης/αφαίρεσης υπολοίπου: Αυτή ορίζεται ως εξής:

$$(A + B)modC = (AmodC + BmodC)modC)$$

Πράξη πολλαπλασιασμού υπολοίπου: Αυτή ορίζεται ως εξής:

$$(AcdotB)modC = (AmodCcdotBmodC)modC$$

Πράξη εκθετικού υπολοίπου: Αυτή ορίζεται ως εξής:

$$A^B modC = ((AmodC)^B)modC$$

Όπως προκύπτει και σε εφαρμογές του RSA πολλές φορές θέλουμε να υπολογίσουμε το $A^B modc$ για μεγάλες τιμές του B. Αυτό αποδυκνείεται να να είναι πολύ δύσκολο

2.3 Δημιουργία κλειδιών

Αυτό που κάνει τον RSA να ξεχωρίζει από τους περισσότερους άλλους κρυπτογραφικούς αλγορίθμους είναι ότι τα κλειδιά που χρησιμοποιούνται από τους χρήστες υπολογίζονται σύμφωνα με μαθηματικές ιδιότητες των πρώτων αριθμών και δεν παράγονται τυχαία. Σε αυτό επίσης έγκειται και η μεγάλη αξιοπιστία του. Η φάση της παραγωγής των κλειδιών είναι σημαντική και βρίσκεται στο επίκεντρο της υλοποίησης του αλγορίθμου.

Η δημιουργία των κλειδιών του RSA αποτελείται από τα παρακάτω πέντε βήματα:

- Επέλεξε δύο μεγάλους πρώτους αριθμούς τους οποίους ονομάζουμε p και q.
- Υπολόγισε το γινόμενό τους:
$$n = p \times q$$
- Χρησιμοποίησε την συνάρτηση φ(n), γνωστή ως συνάρτηση του Euler, για τον υπολογισμό του:
$$\varphi(n) = (p-1) \times (q-1)$$
- Επέλεξε έναν ακέραιο e τέτοιον ώστε να ισχύει 1 ¡ e ¡ φ(n) και εξασφάλισε ότι οι e και φ(n) είναι σχετικά πρώτη ελέγχοντας εάν ισχύει το εξής:
$$gcd(n, \varphi(n)) = 1$$
- Υπολόγισε έναν ακέραιο αριθμό d για τον οποίο ισχύει:
$$d = e^{-1} mod \varphi(n)$$

Αφού ολοκληρωθούν τα πέντε βήματα που αναφέρθηκαν, έχουμε πλέον παράγει δύο ασύμμετρα κλειδιά τα οποία μπορούν να χρησιμοποιηθούν για τις διαδικασίες της κρυπτογράφησης και αποκρυπτογράφησης που θα δούμε αμέσως μετά. Το δημόσιο κλειδί αποτελείται από τα e, n ενώ το ιδιωτικό κλειδί από τα d, n.

$$KU = (e, n)$$
$$KR = (p, n)$$

Το μέγεθος των κλειδιών επιλέγεται να είναι αρκετά μεγάλο ώστε να αυξάνεται η ασφάλεια του κρυπτοσυστήματος. Μία συνηθισμένη τιμή για το μέγεθος των κλειδιών είναι τα 512 bits, έτσι το γινόμενό τους n προκύπτει να έχει μέγεθος 1024 bits. Αυτή η πρακτική αυξάνει σημαντικά την ανοχή του αλγορίθμου σε επιθέσεις τύπου bruteforce κατά τις οποίες ο επιτιθέμενος δοκιμάζει ένα μεγάλο πλήθος κλειδιών κατά συρροή, ελπίζοντας να πετύχει το σωστό. Όπως είναι λογικό, όσο μεγαλύτερο το πλήθος των bits των κλειδιών τόσο περισσότεροι και οι συνδυασμοί τους, άρα τόσο πιο χρονοβόρα η διαδικασία της bruteforce εύρεσης τους.

2.3.1 GCD

Η εύρεση του ΜΚΔ (GCD) δύο φυσικών αριθμών a, b είναι απλή διαδικασία καθώς μπορεί να χρησιμοποιηθεί ο αλγόριθμος του Ευκλείδη, που δίνει αποτέλεσμα με O(log a) διαιρέσεις(O(log3 a)) bit operations). Για οποιαδήποτε a, b λοιπόν με διαδοχικές διαιρέσεις έχουμε:

$$\begin{array}{rll} a & = bq_1 + r_1 & 0 < r_1 < b \\ b & = r_1 q_2 + r_2 & 0 < r_2 < r_1 \\ r_1 & = r_2 q_3 + r_3 & 0 < r_3 < r_2 \\ & \vdots & \vdots \\ r_{j-2} & = r_{j-1} q_j + r_j & 0 < r_j < r_{j-1} \\ r_{j-1} & = r_j q_{j+1} & \end{array}$$

Εικόνα 1: GCD.

$$
\begin{aligned}
1742 &= 3 \cdot 494 + 260 \\
494 &= 1 \cdot 260 + 234 \\
260 &= 1 \cdot 234 + 26 \\
234 &= 9 \cdot 26
\end{aligned}
\qquad
\begin{aligned}
132 &= 3 \cdot 35 + 27 \\
35 &= 1 \cdot 27 + 8 \\
27 &= 3 \cdot 8 + 3 \\
8 &= 2 \cdot 3 + 2 \\
3 &= 1 \cdot 2 + 1 \\
2 &= 2 \cdot 1
\end{aligned}
$$

Εικόνα 2: GCD παράδειγμα.

Μέχρι να βρούμε ένα rj που να διαιρεί ακριβώς το προηγούμενο υπόλοιπο rj1. Τότε το rj είναι ο ΜΚΔ. Για παράδειγμα:

Δηλαδή ο ΜΚΔ μπορεί να γραφεί ως γραμμικός συνδυασμός των a, b. Μάλιστα αυτό μπορεί να γίνει σε πολυωνυμικό χρόνο (O(log3(a) bit operations). Ο αλγόριθμος που υπολογίζει τα x, y στηρίζεται στον αλγόριθμο του Ευκλείδη κάνοντας την αντίστροφή διαδικασία π.χ. στο προηγούμενο παράδειγμα για να υπολογίσουμε το 26 (το 1) ως γραμμικό συνδυασμό των 1742 και 494 (των 132 και 35) κάνουμε τα εξής:

$$
\begin{aligned}
26 &= 260 - 234 \\
&= 260 - (494 - 260) \\
&= 2 \cdot 260 - 494 \\
&= 2(1742 - 3 \cdot 494) - 494 \\
&= 2 \cdot 1742 - 7 \cdot 494
\end{aligned}
\qquad
\begin{aligned}
1 &= 3 - 2 \\
&= 3 - (8 - 2 \cdot 3) \\
&= 3 \cdot 3 - 8 \\
&= 3(27 - 3 \cdot 8) - 8 \\
&= 3 \cdot 27 - 10 \cdot 8 \\
&= 3 \cdot 27 - 10(35 - 27) \\
&= 13 \cdot 27 - 10 \cdot 35 \\
&= 13(132 - 3 \cdot 35) - 10 \cdot 35 \\
&= 13 \cdot 132 - 49 \cdot 35
\end{aligned}
$$

Εικόνα 3: GCD extended παράδειγμα.

2.3.2 Primality testing

Το Primality testing αφορά την εύρεση για το αν ένας αριθμός είναι πρώτος. Μία από τις παλιότερες λύσεις του προβλήματος (primality tests) είναι το Κόσκινο του Ερατοσθένη. Αφού ορίσουμε ένα ανώτατο όριο (αριθμό) k γράφουμε όλους τους αριθμούς 2, 3, 4, 5, ... ,k. Στη συνέχεια, παίρνουμε τον πρώτο αριθμό (εδώ το 2), τον μαρκάρουμε σαν πρώτο και διαγράφουμε όλα τα πολλαπλάσιά του που υπάρχουν στη λίστα (4, 6, 8, ...). Στη συνέχεια παίρνουμε τον επόμενο αριθμό που δεν έχει διαγραφεί, τον μαρκάρουμε ως πρώτο και διαγράφουμε όλα τα πολλαπλάσιά του που υπάρχουν στη λίστα κ.ο.κ. Τελικά θα παραμείνουν μόνο οι πρώτοι που είναι μικρότεροι ίσοι του k.

Έλεγχος Miller-Rabin:

- Έστω ότι το n είναι μεγάλος, θετικός, περιττός αριθμός.
- Διαλέγουμε τυχαία b, όπου 0 ¡ b ¡ n. Αν gcd(b, n) = 1 b $b^{n-1} = 1$ (mod n) τότε το n είναι σύνθετος (με βεβαιότητα) και η εκτέλεση σταματά. Αλλιώς:
- Γράφουμε n-1 = $2^s t$, με t περιττό και υπολογίζουμε το b^t mod n. Αν είναι 1 ή -1, τότε το n περνά το τεστ.

- Αλλιώς, υψώνουμε στο τετράγωνο το $b^t mod n$, έπειτα το ξαναϋψώνουμε στο τετράγωνο mod n κ.ο.κ. για s - 1 φορές το πολύ, έως ότου πάρουμε 1 ή -1.
- Αν πήραμε -1 τότε το n περνάει το τεστ (πιθανόν πρώτος).
- Αν πήραμε +1, τότε το n δεν περνά το τεστ (σύνθετος, με βεβαιότητα) και η εκτέλεση σταματά.
- Επαναλαμβάνουμε τα παραπάνω βήματα 2–6 k φορές, με διαφορετικό b κάθε φορά. Αν το n περάσει το τεστ και τις k φορές ανακοινώνεται ότι ο n περνάει το τεστ (δηλαδή είναι πρώτος με πολύ μεγάλη πιθανότατα).

Αποδεικνύεται ότι το πολύ τα μισά b περνούν το τεστ για n σύνθετο. Συνεπώς, αν το n περάσει το τεστ για k τυχαίες επιλογές του b, τότε η πιθανότητα να μην είναι πρώτος είναι $1/2k$.

Έλεγχος Fermat:

Έστω ένας δοσμένος ακέραιος n, ελέγχουμε αν είναι πρώτος ως εξής:

- Επιλέγουμε τυχαία έναν αριθμό $a \in Z$
- Αν $a^{n-1} \not\equiv 1 \, (mod\, n)$ τότε ο n είναι σύνθετος και άρα με βεβαιότητα δεν είναι πρώτος
- Αν όμως το προηγούμενο ισχύει τότε ο n ίσως είναι πρώτος και επαναλαμβάνουμε την διαδικασία από το πρώτο βήμα

2.4 Κρυπτογράφηση και αποκρυπτογράφηση

Αξίζει κανείς να δει αυτές τις διαδικασίες μαζί καθώς η μία αποτελεί συμπλήρωμα της άλλης. Αφού παραχθούν τα κλειδιά περνάμε στην φάση κρυπτογράφησης και αποκρυπτογράφησης των μηνυμάτων. Όπως αναφέρθηκε και στα εισαγωγικά του αλγορίθμου κάθε χρήστης έχει δύο κλειδιά, το δημόσιο και το ιδιωτικό. Τα δημόσια κλειδιά όλων των χρηστών βρίσκονται σε μία βάση δεδομένων την οποία χρησιμοποιεί το κρυπτοσύστημα. Ενώ τα αντίστοιχα ιδιωτικά κλειδιά των χρηστών βρίσκονται τοπικά στη συσκευή του καθενός και είναι γνωστά μόνο στον εκάστοτε χρήστη.

Όποιος επιθυμεί να κρυπτογραφήσει ένα μήνυμα πρέπει να χρησιμοποιήσει το δημόσιο κλειδί του παραλήπτη. Έτσι εξασφαλίζεται ότι το μήνυμα μπορεί να αποκρυπτογραφηθεί μόνον από τον παραλήπτη, ακόμη και στην περίπτωση που το μήνυμα υποκλαπεί. Οι διαδικασίες κρυπτογράφησης και αποκρυπτογράφησης ορίζονται με τις παρακάτω εξισώσεις:

κρυπτογράφηση: $c = m^e mod n$, όπου m είναι το αρχικό κείμενο

αποκρυπτογράφηση: $m = c^d mod n$, όπου c είναι το κρυπτοκείμενο

Ας θυμηθούμε ότι το δημόσιο κλειδί του παραλήπτη αποτελείται από τους αριθμούς e, n. Άρα ο αποστολέας διαβάζει το δημόσιο κλειδί του παραλήπτη από την βάση του κρυπτοσυστήματος και χρησιμοποιεί τους αριθμούς e, n καθώς και τον τύπο κρυπτογράφησης που δείξαμε προηγούμενος για να κρυπτογραφήσει το μήνυμα. Κάθε χαρακτήρας του μηνύματος αντιστοιχίζεται σε έναν ακέραιο αριθμό. Κάθε αριθμός χαρακτήρα αντικαθίσταται στο σύμβολο m του τύπου κρυπτογράφησης μαζί με τα e, n. Σταδιακά παράγονται οι ακέραιοι αριθμοί c οι οποίοι δομούν το κρυπτοκείμενο. Το κρυπτοκείμενο αποστέλλεται. Ο παραλήπτης λαμβάνει το κρυπτοκείμενο το οποίο αποκρυπτογραφεί χρησιμοποιώντας τον τύπο της αποκρυπτογράφησης που είδαμε προηγουμένως. Ας θυμηθούμε επίσης ότι το ιδιωτικό κλειδί του παραλήπτη αποτελείται από τους αριθμούς d, n. Άρα γίνεται αντικατάσταση των αριθμών αυτών στον τύπο αποκρυπτογράφησης, ενώ επίσης γίνεται αντικατάσταση των ακεραίων από τους οποίους αποτελείται το κρυπτοκείμενο διαδοχικά στην θέση c του τύπου. Λαμβάνονται σταδιακά όλοι οι ακέραιοι που αποτελούν το αρχικό κείμενο. Αντιστοιχίζονται οι ακέραιοι με τα γράμματα και πλέον έχουμε το κείμενο σε αναγνώσιμη μορφή.

2.5 Πρακτικές βελτιστοποίησης αλγορίθμου

2.5.1 Μέθοδος Padding

Στην κρυπτογραφία με τον όρο padding εννοούμε ένα σύνολο από πρακτικές οι οποίες προσθέτουν επιπλέον πληροφορία στην αρχή, τη μέση ή το τέλος του αρχικού κείμενου. Η πληροφορία που προστίθεται είναι συνήθως φράσεις χωρίς κανένα λογικό νόημα ή απλά τυχαίες ακολουθίες χαρακτήρων, με σκοπό να αντιμετωπιστεί το γεγονός ότι πολλά μηνύματα καταλήγουν να κρυπτογραφούνται με προβλέψιμο αποτέλεσμα. Οι πρακτικές Padding έρχονται να αντιμετωπίσουν προσπάθειες κρυπτοάναλυσης χρησιμοποιώντας την προβλεψιμότητα κάποιον γνωστών αρχικών κειμένων.

Ας δούμε το εξής παράδειγμα από τον Β' παγκόσμιο πόλεμο για να γίνει πιο αντιληπτή η έννοια του padding. Κατά την διάρκεια της μάχης του Samar ο ναύαρχος Chester Nimitz έστειλε κρυπτογραφημένο μήνυμα στον ναύαρχο Bull Halsey για να μάθει την θέση του στόλου του ώστε να μπορέσουν να συντονιστούν. Το αρχικό μήνυμα ήταν το εξής:

Where is, repeat, where is Task Force Thirty Four?

Μετά την κρυπτογράφηση και την πρόσθεση μεταδεδομένων έγινε:

GG FROM CINCPAC ACTION COM THIRD FLEET INFO COMINCH CTF SEVENTY-SEVEN X WHERE IS RPT WHERE IS TASK FORCE THIRTY FOUR RR

Τέλος με την προσθήκη padding πήρε την εξής τελική μορφή:

TURKEY TROTS TO WATER GG FROM CINCPAC ACTION COM THIRD FLEET INFO COMINCH CTF SEVENTY-SEVEN X WHERE IS RPT WHERE IS TASK FORCE THIRTY FOUR RR THE WORLD WONDERS

Με την χρήση των μεταδεδομένων θα έπρεπε ο παραλήπτης να το αποκρυπτογραφήση επιτυχώς. Ας δούμε τώρα μερικές από τις πρακτικές padding Που χρησιμοποιούνται σήμερα.

- Bit padding.

  Μπορεί να εφαρμοστεί σε μηνύματα οποιουδήποτε μεγέθους. Ένα bit με τιμή 1 εισάγεται στο μήνυμα και στην συνέχεια εισάγεται ένας πλήθος από bits με τιμή 0. Ο αριθμός των μηδενικών bits εξαρτάται από το όριο του block στο οποίο χρειάζεται να επεκταθεί το μήνυμα. Η συγκεκριμένα πρακτική padding είναι το πρώτο βήμα μίας πρακτικής padding δύο βημάτων η οποία χρησιμοποιείται σε πολλές συναρτήσεις hash συμπεριλαμβανομένων των MD5 και SHA.

- ANSI X9.23

  Σε αυτή την πρακτική προστίθενται μεταξύ ενός και οχτώ bytes σαν padding. Η τιμή των bytes ορίζεται τυχαία ενώ το τελευταίο byte ορίζεται ως το πλήθος των bytes που προστέθηκαν.

- ISO 10126

  Ορίζει ότι το padding πρέπει να προστεθεί στο τέλος του τελευταίου block το οποίο θα αποτελείται από τυχαία ορισμένα bytes ενώ το όριο του padding θα ορίζεται από το τελευταίο byte.

- Zero padding

  Αυτή η πρακτική ορίζει ότι όλα τα byte που θα τοποθετηθούν θα είναι μηδενικά. Είναι μία πρακτική που χρησιμοποιείται για hashes και MACs. Παρουσιάζει την εξής αδυναμία, εάν το αρχικό κείμενο τελειώνει με ένα ή περισσότερα μηδενικά bytes τότε είναι αδύνατον να ξεχωρίσουν τα μηδενικά bytes του αρχικού κειμένου από αυτά του padding.

- OAEP

  Αναπτύχθηκε από τους Mihir Bellare και Phillip Rogaway και είναι μία πρακτική padding Που χρησιμοποιείται συνήθως με τον αλγόριθμο RSA. Ο αλγόριθμος OAEP είναι μία μορφή Freistel δίκτυο το οποίο χρησιμοποιεί ένα ζευγάρι από τυχαία oracles G Και H για να επεξεργαστεί το αρχικό κείμενο πριν την διαδικασία της κρυπτογράφησης. Ας δούμε παρακάτω τα βήματα του αλγορίθμου.

  Για την κρυπτογράφηση του μηνύματος:

  - Προστίθεται padding στα μηνύματα με K1 μηδενικά ώστε να έχουν μήκος n - K0 bits.
  - Το r δημιουργείται τυχαία ως μία ακολουθεί από K0 bits.
  - Το G επεκτείνει τα K0 Bits του r σε n - K0 bits.
  - Το X υπολογίζεται ως X = m000..0 XOR G(r)
  - Το H μειώνει τα n - K0 bits του X σε K0 bits
  - Το Y υπολογίζεται ως Y = r XOR H(X).
  - Η έξοδος είναι X OR Y.

  Το κρυπτογραφημένο μήνυμα που παράχθηκε με τον προηγούμενο αλγόριθμο μπορεί στην συνέχεια να κρυπτογραφήσει περαιτέρω με τον RSA.

  Για την αποκρυπτογράφηση του μηνύματος:

  - Ανέκτησε το r ως r = Y XOR H(X).
  - Ανέκτησε το μήνυμα ως m00...0 = X XOR G(r).

2.5.2 Υπολογισμός modulo πολλαπλασιασμών με τον αλγόριθμο Montgomery

Η πράξη πολλαπλασιασμός modulo προκύπτει κατά την αποκρυπτογράφηση. Ο αλγόριθμος αυτός Χρησιμοποιείται για υλοποιήσεις του αλγορίθμου σε Hardware. Η πράξη του modulo πολλαπλασιασμού ορίζεται ως:

$$A \times B \,(mod\, n)$$

Αυτή προκύπτει ειδικά όταν υλοποιούνται και άλλες τεχνικές βελτιστοποίησης του αλγορίθμου, όπως ο αλγόριθμος square and Multiply που θα δούμε αμέσως μετά. Υπάρχουν και άλλες μέθοδοι βελτιστοποίησης της πράξης modulo πολλαπλασιασμού όπως η Multiply then divide, η Interleaving multiplication and reduction και η μέθοδος Brickell. Εκεί που κερδίζει ο αλγόριθμος Montgomery σε σχέση με τις άλλες είναι ότι αποφεύγει την πράξη της διαίρεσης κατά την υλοποίησή του και εναλλακτικά χρησιμοποιεί τελεστές μετατόπισης και πρόσθεσης.Βλέπουμε παρακάτω τα βήματα του αλγορίθμου.

Για την αποκρυπτογράφηση του μηνύματος:
Έστω ότι θέλουμε να υπολογίσουμε το εξής $M = A \times B \,(mod\, n)$, όπου οι αριθμοί A, B είναι άκαιροι μήκους κ bit. Συμβολίζουμε με Ai, Bi το i-οστο bit των A και B αντίστοιχα.

- Ορίζουμε M = 0.
- Για κάθε bit των αριθμών ξεκινώντας από το i = 0 κάνε το εξής.
  - Υπολόγισε $M = M + (A_i)$
  - Εάν ισχύει $M_0 = 1$, τότε
  - Θέσε $M = M/2$
  - Αλλιώς θέσε $M = (M + n)/2$
- Τελικά επέστρεψε το M.

Παρακάτω βλέπουμε τον αλγόριθμο και σε μορφή ψευδοκώδικα

```
      Input: A, B, n
      Output: M = A×B mod n
M = 0;
For i = 0 to k
      M=M+(A×B_i)
        if M_0 = 1
           M = M/2;
        else
        M=(M+n)/2;
      return M;
```

Εικόνα 4: Αλγόριθμος Montgomery.

2.5.3 Υπολογισμός εκθετικού modulo με τον αλγόριθμο Fast Powering

Ο αλγόριθμος αυτός μπορεί να βρεθεί στην βιβλιογραφία και με τα ονόματα Square and Multiply, exponentiation by squaring και binary exponentiation. Κατά την διαδικασία της αποκρυπτογράφησης χρειάζεται να γίνει ο εξής υπολογισμός: $m = c^d \,mod\, n$. Αυτός αποδεικνύεται να είναι εξαιρετικά δύσκολος για μεγάλες τιμές του d. Αυτό όμως είναι το ιδιωτικό κλειδί το οποίο, όπως και το δημόσιο κλειδί, θέλουμε να είναι όσο το δυνατόν μεγαλύτερα για να διασφαλίζουμε την αξιοπιστία του αλγορίθμου. Άρα αν θέλουμε η αποκρυπτογράφηση να γίνεται σε λογικό χρόνο θα πρέπει να περιοριζόμαστε σε μικρά μήκη ιδιωτικών κλειδιών και άρα να συμβιβαζόμαστε σε λιγότερη ασφάλεια κατά την επικοινωνία.Το ίδιο πρόβλημα προκύπτει και κατα΄την διαδικασία της κρυπτογράφησης. Η μέθοδος Square and Multiply είναι μία πρακτική για γρήγορο και εύκολο υπολογισμό της πράξης εκθετικό modulo, ακόμη και όταν το μέγεθος του ιδιωτικού κλειδιού ή του δημόσιου κλειδιού είναι πολύ μεγάλο. Άρα η συγκεκριμένη μέθοδος δεν προσφέρει μόνο μεγαλύτερη ταχύτητα υπολογισμών αλλά μας δίνει και περιθώριο να αυξήσουμε την αξιοπιστία του συστήματος RSA. Τα βήματα του αλγορίθμου είναι τα εξής:

- Έστω ότι θέλουμε να υπολογίσουμε την εξής έκφραση:

$$a^m \, mod \, N$$

- Υπολόγισε τα $a^2 \, mod \, N, a^4 \, mod \, N, a^8 \, mod \, N$ για όλα τα $2^k \leq m$
- Αν το αποτέλεσμα a^i για κάποιο i βγαίνει μεγαλύτερο του N τότε μείωσε το ως εξής:

$$a^i \, mod \, N$$

- Όρισε το a^m σαν γινόμενο των παραγόντων $a, a^2, a^4, ..., s^{2^k}$
- Υπολόγισε το γινόμενο Q
- Υπολόγισε το $Q \, mod \, N$

Ας δούμε το εξής παράδειγμα για την λειτουργία του αλγορίθμου.Έστω ότι θέλουμε να υπολογίσουμε το $3^{35} \, mod \, 37$. Ξεκινάμε με τον υπολογισμό των $3^1, 3^2, 3^4, 3^8, 3^16, 3^32$.

$$3^1 \equiv 3$$

$$3^2 \equiv 9$$

$$3^4 \equiv 81$$

τώρα θα πρέπει να μειώσουμε το 81 ως εξής:

$$81 \, mod \, 37 = 7$$

Άρα έχουμε πλέον:

$$3^4 \equiv 7$$

Και συνεχίζουμε στα υπόλοιπα εκθετικά:

$$3^8 \equiv 7^2 \equiv 49 \ \text{, το οποιο μειωνουμε σε } 12$$

$$3^16 \equiv 12^2 \equiv 144 \ \text{, το οποιο μειωνουμε σε } 33$$

$$3^32 \equiv 33^2 \equiv 1089 \ \text{, το οποιο μειωνουμε σε } 16$$

Άρα τελικά έχουμε:

$$33^35 \equiv 3^32 \times 3^2 = 432$$

Το οποίο μειώνουμε τελικά σε

$$432 \, mod \, 37 \equiv 25$$

2.5.4 Κινέζικο θεώρημα Υπολοίπων (CRT)

Το κινέζικο θεώρημα υπολοίπων αναφέρει ότι εάν γνωρίζουμε τα υπόλοιπα $l_1, l_2, ..., l_k$ ενός αριθμού N με διάφορους ακεραίους $d_1, d_2, ..., d_k$, τότε μπορούμε να υπολογίσουμε το υπόλοιπο Υ του N με το γινόμενο $D = d_1 d_2 ... d_k$ αρκεί οι ακέραιοι να είναι μεταξύ τους πρώτοι.

$$\text{Οπότε έχουμε: } Y = \upsilon 1 D1 \chi 1 + \upsilon 2 D2 \chi 2 + ... + \upsilon k Dk \chi k$$

όπου $D1 = D/d1$, χ1 η λύση της $D1\chi 1 = 1(modd1)$, κ.ο.κ. και $N = \Upsilon(modD)$.

Στην περίπτωση του αλγορίθμου RSA το CRT εφαρμόζεται ως εξής: Έστω οι ακέραιοι p, q είναι πρώτη μεταξύ τους. Τότε η συσχέτιση $X \equiv a \, mod \, pq$ ισχύει αν και μόνο αν

$$X \equiv a \, mod \, p$$

$$X \equiv a \, mod \, q$$

Ας θυμηθούμε ότι κατά την φάση της αποκρυπτογράφησης ο παραλήπτης πρέπει να υπολογίζει την εξής έκφραση:

$$m = c^d \, mod \, n$$

Η οποία για μεγάλες τιμές του n μπορεί να γίνει υπολογιστικά πολύ δύσκολη. Ο παραλήπτης γνωρίζει τους παράγοντες του n και άρα μπορεί να σπάσει τον υπολογισμό σύμφωνα με την προηγούμενη μεθοδολογία. το ίδιο συμβαίνει και για την φάση της κρυπτογράφησης όπου καλούμαστε να λύσουμε το:

$$c = m^e \, mod \, n$$

Ας δούμε το εξής παράδειγμα για να καταλάβουμε ακριβώς την λειτουργία. Έστω ότι στον Bob έχουν ανατεθεί από το RSA σύστημα το ιδιωτικό κλειδί $e = 65537$ και το δημόσιο κλειδί $d = 5731241$. Επίσης γνωρίζει ότι $N = p = 17086049$, με $p = 3863 \, \kappa\alpha\iota \, q = 4423$. Υποθέτουμε τώρα ότι ο bob λαμβάνει το μήνυμα $c = 4831984$ από την Alice. Ο Bob χρησιμοποιεί την πρακτική CRT για να το αποκρυπτογραφήσει και να ανακτήσει το αρχικό κείμενο m. Για να το κάνει αυτό θα πρέπει να λύσει ως γνωστών το εξής $m \equiv c^d \, mod \, N$, άρα στη συγκεκριμένη περίπτωση η εξίσωση γίνεται $m \equiv 4831984^{5731241} \, mod \, 17086049$.Άρα σύμφωνα με το CRT το πρόβλημα μετατρέπεται στο εξής:

$$m \equiv 4831984^{5731241} \, mod 3863$$

$$m \equiv 4831984^{5731241} \, mod 4423$$

Τώρα βλέπουμε ότι ισχύει $33 \equiv 5731241 mod 3862$ και $392 \equiv 5731241 mod 4423$. Άρα οι προηγούμενες δύο εξισώσεις απλοποιούνται στις εξής.

$$m \equiv 4831984^{33} \equiv 3084 \, mod 3863$$

$$m \equiv 4831984^{329} \equiv 1436 \, mod 4423$$

Άρα τελικά το αρχικό κείμενο ανακτάται και είναι το:

$$m = 3084 + 3863 \times 1319 \times (1436 - 3084) \times (mod 3863 \times 4423) = 9289736 (mod N)$$

Η απάντηση είναι:

$$4831984^{5731241} \equiv 9289736 (mod N)$$

2.5.5 Αλγόριθμος RSA-CRT

Η συγκεκριμένη υλοποίηση του αλγορίθμου RSA χρησιμοποιεί τις βελτιστοποιήσεις που προσφέρουν οι μέθοδοι Square and Multiply και CRT ώστε η ταχύτητα εκτέλεσής του να βελτιστοποιείται. Συγκεκριμένα το CRT εφαρμόζεται κατά την φάση της αποκρυπτογράφησης όπως είδαμε και προηγούμενος ενώ ο αλγόριθμος Square and Multiply χρησιμοποιείται για την ταχύτερη εκτέλεση της πράξεις εκθετικού modulo κατά την φάση της αποκρυπτογράφησης ως επιπλέον βελτιστοποίηση στην διαδικασία του CRT.

Η διαδικασία αποκρυπτογράφησης σε αυτή την περίπτωση είναι η εξής:

- Ο παραλήπτης λαμβάνει το μήνυμα c.
- Αποκρυπτογράφησε σύμφωνα με την: $m = c^d \, mod \, n$
- Εφάρμοσε το CRT και πλέον το πρόβλημα είναι $m = c^d \, mod \, p$ και $m = c^d \, mod \, q$

- Απλοποίησε το c^d και για τις δύο περιπτώσεις. Το πρόβλημα πλέον είναι:

$$m = a \, mod \, p$$

$$m = b \, mod \, q$$

- Λύσε τη σχέση:

$$max[p,q]X + [a \; or \; b \; of \; max[p,q]] = [a \; or \; b \; of \; min[p,q] \; mod \; min[p,q]$$

- Το μήνυμα τελικά δύνεται από την εξίσωση:

$$m = max[p,q]X + min[p,q]$$

2.5.6 Βελτιστοποίηση σε Hardware

Ο αλγόριθμος RSA βασίζεται στην αριθμητική υπολοίπων και σε πράξεις modulo. Αν αυτές οι πράξεις modulo πραγματοποιούνται σε κομμάτια υλικού τα οποία βελτιστοποιούνται σύμφωνα με αυτές τότε και ο αλγόριθμος θα τρέχει σημαντικά πιο γρήγορα. Οι Sushanta και Manoranjan [3] προτείνουν μία αρχιτεκτονική για επιτάχυνση του αλγορίθμου RSA. Στοχεύουν στην επιτάχυνση των πράξεων εκθετικό modulo , πολλαπλασιασμός modulo και πρόσθεση/αφαίρεση modulo. Συγκεκριμένα για την επιτάχυνση της πράξης εκθετικού modulo χρησιμοποιούν των αλγόριθμο Square and Multiply. Τον συγκεκριμένο αλγόριθμο τον αναλύσαμε και στο μέρος 2.5.3, ωστόσο στην συγκεκριμένη υλοποίηση παίρνει μία άλλη μορφή ώστε να εκμεταλλευτεί τις ιδιότητες της δυαδικής αναπαράστασης η οποία είναι διαθέσιμη σε επίπεδο hardware. Ο ψευδοκώδικας δίνεται παρακάτω.

Input: M, e, n
Output: $C = M^e \bmod n$
Let e contain k bits)

If e_{k-1}=1 then C=M else C=1

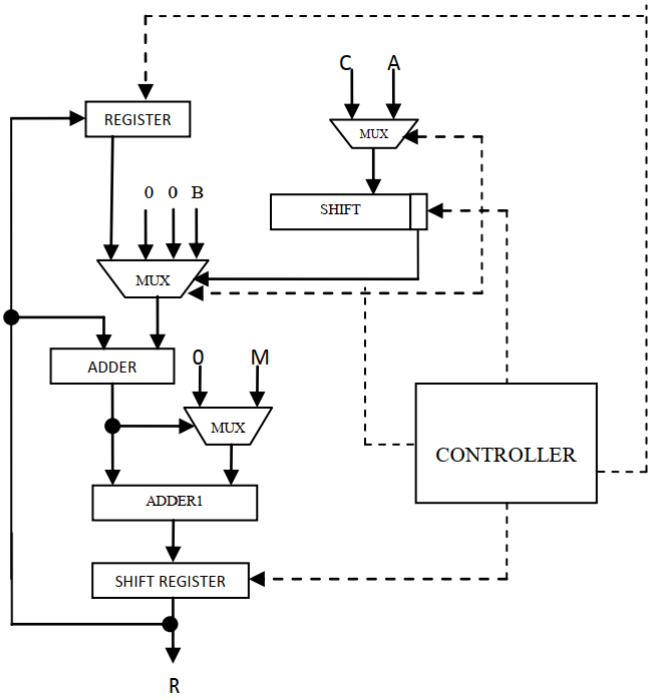
For i=k-2 down to 0

C=C×C

If e_i=1 then C=C×M

Εικόνα 5: Αλγόριθμος Square and multiply.

Όπως είδαμε και πριν με τη χρήση της μεθόδου προκύπτει ένα πλήθος από πράξεις πολλαπλασιασμών modulo.Για την επιτάχυνσή τους σε αυτή την εφαρμογή προτείνεται ο αλγόριθμος του Montgomery όπως τον είδαμε και στο μέρος 2.5.2. Παρακάτω βλέπουμε μία προτεινόμενη αρχιτεκτονική για την υλοποίηση του αλγορίθμου σε hardware.



Εικόνα 6: Αρχιτεκτονική αλγορίθμου Montgomery.

Για την επιτάχυνση των πράξεων πρόσθεση και αφαίρεση Modulo προτείνουν την χρήση απλών carry save adders.

2.6 Προγραμματιστική Υλοποίηση

2.6.1 Κλάση RSA

Για την υλοποίηση του RSA αποφασίσαμε να υλοποιήσουμε μία κλάση ως wrapper των μεταβλητών και των συναρτήσεων που απαιτούνται στον αλγόριθμο. Κάθε αντικείμενο της κλάσης αποθηκεύει τρεις μεταβλητές publicKey, privateKey και modulo που αντιστοιχούν στα e, d και n. Η κλάση περιλαμβάνει τις ακόλουθες συναρτήσεις, οι οποίες και αναλύονται στις επόμενες παραγράφους:

- GenerateNum()
- GeneratePrime()
- PrimalityTest()
- ModularExponentation()
- GCD()
- ExtendedGCD()
- Encryption()
- EncryptionWithForeignKey()
- Decrytpion()

Αξίζει να σημειωθεί ότι για την αποθήκευση των αριθμών πολλών bits χρησιμοποιήσαμε την βιβλιοθήκη big_int που αποτελεί μέρος της boost. Η βιβλιοθήκη big_int παρέχει τη δυνατότητα πραγματοποίησης πράξεων με αριθμούς πολλών bits, όπως με 2048 bits που χρησιμοποιήθηκε στην υλοποίηση για ασφαλή λειτουργία του αλγορίθμου RSA.

2.6.2 Constructor

Κάθε αντικείμενο της κλάσης RSA θα πρέπει να έχει τα δικά του κλειδιά, για αυτό η διαδικασία παραγωγής τους γίνεται στον constructor. Ειδικότερα, κατά την αρχικοποίηση του αντικειμένου παράγονται τυχαία δύο αριθμοί p και q, μέσω της συνάρτησης GenerateNum(), επανειλημμένα, και ελέγχονται μέσω της συνάρτησης PrimalityCheck() μέχρι να αποδειχθεί πως είναι πρώτοι. Για την επιτάχυνση της διαδικασίας οι δύο αριθμοί παράγονται ταυτόχρονα, με την χρήση threads. Στη συνέχεια, υπολογίζεται το δημόσιο κλειδί με την διαδικασία που προσδιορίζει το πρότυπο του RSA. Για το ιδιωτικό κλειδί χρησιμοποιήσαμε την μέθοδο του ExtendedGCD().

```
RSA::RSA(){
  big_int p, q;
  big_int phi_n;
  //create threads for faster computation
  // thread 1
  std::promise<big_int> promisedP;
  std::future<big_int> futureP = promisedP.get_future();
  std::thread th1(&RSA::GeneratePrime, this, &promisedP);

  do {
    do
      q = GenerateNum(1024);
    while (q % 2 == 0);
  } while (!PrimalityTest(20, q));
  p = futureP.get();
  // Waiting for thread to finish
  th1.join();
  modulo = p * q;
  phi_n = (p - 1) * (q - 1);
  do
    publicKey = GenerateNum(2048) % (phi_n - 2) + 2; // 1 < publicKey < phi_n
  while (GCD(publicKey, phi_n) != 1);

  privateKey = ExtendedGCD(phi_n, publicKey);
}
```

2.6.3 GenerateNum()

Η συνάρτηση GenerateNum() παίρνει σαν όρισμα τον αριθμό των bit που θέλουμε να έχει ο αριθμός που θα παραχθεί, τον μετατρέπει σε αριθμό δεκαδικών ψηφίων σύμφωνα με τον τύπο:

$$NumberOfDigits = floor(NumberOfBits/\log 2(10) + 1)$$

και έπειτα παράγει έναν big_int με τόσα ψηφία.

```
big_int RSA::GenerateNum(int NumOfBits){
  big_int Num;
  int NumOfDigits = NumOfBits/3.321928095 + 1;
  int temp;
  std::stringstream ss;
  temp = rand()%9+1; //Make sure the first number is not 0
  ss << temp;
  for (int i = 0; i < NumOfDigits-1; i++)
  {
    temp = rand()%10;
    ss << temp;
  }
  ss >> Num;
  return Num;
}
```

2.6.4 GeneratePrime()

Η συνάρτηση GeneratePrime() είναι μια βοηθητική συνάρτηση που χρησιμοποιείται μόνο από το thread που παράγει το p. Πιο συγκεκριμένα, η συνάρτηση αποτελείται από έναν βρόγχο, ο οποίος καλεί την συνάρτηση GenerateNum() για την παραγωγή ενός αριθμού με 1024 bits και στη συνέχεια ελέγχει αν πρόκειται για πρώτο αριθμό, καλώντας τη συνάρτηση PrimalityTest(). Αν ο αριθμός δεν είναι πρώτος τότε ο βρόγχος επαναλαμβάνει την παραπάνω διαδικασία, διαφορετικά η συνάρτηση επιστρέφει τον αριθμό που έχει βρει ότι είναι πρώτος.
Επισημαίνουμε ότι στην παραπάνω διαδικασία καλούμε τη συνάρτηση PrimalityTest() με όρισμα IterNum=20 διότι το Miller-Rabin τεστ δεν μπορεί με μια βάση πάντα να αποφανθεί αν πρόκειται για πρώτο ή σύνθετο.

2.6.5 PrimalityTest()

Για την συνάρτηση PrimalityTest() χρησιμοποιήσαμε έναν πίνακα με τους μικρότερους 99 πρώτους και έπειτα εφαρμόζουμε την μέθοδο Miller-Rabin. Η συνάρτηση παίρνει σαν όρισμα των αριθμό των ελέγχων που θα κάνει.

```
bool RSA::PrimalityTest(int IterNum, big_int n){
  int primes[] = {3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,
  79,83,89,97,101,103,107,109,113,131,137,139,149,151,157,163,167,173,179,
  181,191,193,197,199,211,223,227,229,233,239,241,251,257,263,269,271,277,
  281,283,293,307,311,313,317,331,337,347,349,353,359,367,373,379,383,389,
  397,401,409,419,421,431,433,439,443,449,457,461,463,467,479,487,491,499,503,509,521,523,541};
  for (int i = 0; i < 98; i++)
  {
    if (n % primes[i]==0)
      return false;
  }

  for (int j = 0; j < IterNum; ++j){
    big_int d = n - 1;
    big_int k = 0;
    big_int T;
    // select a random base
    big_int a = 2 + rand() % (n - 4);

    // Find d,k such that n = (2^k)*d + 1
    while (d % 2 == 0) {
      k++;
      d = d / 2;
    }

    // Miller-Rabin test
    T = ModularExponentiation(a, d, n);

    if (T == 1 || T == n - 1)
      return true;

    for (int i = 0; i < k; i++) {
      T = ModularExponentiation(T, 2, n);
      if (T == 1)
        return false;
      if (T == n - 1)
        return true;
    }
  }
  return false;
}
```

2.6.6 ModularExponentiation()

Για τον υπολογισμό του υπολοίπου υλοποιήσαμε την μέθοδο Exponentiation by squaring που περιγράφεται στην παράγραφο 2.5.3 και πιο συγκεκριμένα την right-to-left binary μέθοδο.

```
big_int RSA::ModularExponentiation(big_int base, big_int exponent, big_int modulus){
  big_int r;
  big_int y = 1;

//Right-to-left binary method
  while (exponent > 0) {
    r = exponent % 2;
    if (r)
      y = y * base % modulus;
    base = base * base % modulus;
    exponent = exponent>>1;
  }
  return y;
}
```

2.6.7 GCD() και ExtendedGCD()

Η συνάρτηση GCD() αποτελεί μια απλή υλοποίηση του Ευκλείδειου ΜΚΔ, ενώ η ExtendedGCD() είναι ελαφρώς αλλαγμένη ώστε να επιστρέφει μόνο το συντελεστή του α (στην περίπτωσή μας του totient coefficient Φ(n)) και χρησιμοποιείται μόνο για την παραγωγή του κρυφού κλειδιού.

```
big_int RSA::GCD(big_int a, big_int b)
{
  big_int temp;

  if (a < b) {
    temp = a;
    a = b;
    b = temp;
  }

  while (b > 0) {
    temp = a % b;
    a = b;
    b = temp;
  }

  return a;
}
```

```
big_int RSA::ExtendedGCD(big_int a, big_int b){
  //ax + by = gcd(a,b)
  big_int inv, x = 0, y = 1;
  big_int q, temp, tempA = a;

  while (b > 0) {
    q = a / b; //Do not remove q
    temp = a % b;
    a = b;
    b = temp;

    temp = x - q * y;
    x = y;
    y = temp;
  }

  if (a == 1)
    inv = x;

  if (inv < 0)
    inv = inv + tempA;

  return inv;
}
```

2.6.8 Encryption()

Η συνάρτηση Encryption() παίρνει σαν ορίσματα δύο file streams, ένα input και ένα output, καθώς και δύο string με τα ονόματα των αρχείων που θα χρησιμοποιήσει. Διαβάζει έναν-έναν τους χαρακτήρες του plaintext, τους κρυπτογραφεί χρησιμοποιώντας την συνάρτηση ModularExponentiation() και έπειτα γράφει το ciphertext().

```
void RSA::Encryption(std::ifstream& inp, std::string plainFile,
        std::ofstream& out, std::string cipherFile){
  inp.open(plainFile);
  // destroy contents of these files (from previous runs, if any)
  out.open(cipherFile);
  out.close();
  // Check if files can be opened.
  if (!inp) {
    printf("Error opening Source File.\n");
    exit(1);
  }

  out.open(cipherFile);
  if (!out) {
    printf("Error opening Destination File.\n");
    exit(1);
  }

  while (true) {
    char ch;
    inp.read(&ch,1);
    if (inp.eof())
      break;
    int value = toascii(ch);
    big_int cipher;
    cipher = ModularExponentiation(value,publicKey , modulo);
    out << cipher << " ";
  }

  inp.close();
  out.close();
}
```

2.6.9 EncryptionWithForeignKey()

Για την κρυπτογράφηση ενός μηνύματος με ξένο κλειδί χρησιμοποιούμε την συνάρτηση EncryptionWithForeignKey(), με δύο επιπλέον ορίσματα, τα e και n, που αποτελούν το δημόσιο κλειδί του συνομιλητή μας.

```
void RSA::EncryptionWithForeignKey(std::ifstream& inp, std::string plainFile, std::ofstream& out,
                std::string cipherFile, big_int ForeignModulo, big_int ForeignPublicKey) {
  inp.open(plainFile);
  // destroy contents of these files (from previous runs, if any)
  out.open(cipherFile);
  out.close();
  // Check if files can be opened.
  if (!inp) {
    printf("Error opening Source File.\n");
    exit(1);
  }

  out.open(cipherFile);
  if (!out) {
    printf("Error opening Destination File.\n");
    exit(1);
  }

  while (true) {
    char ch;
    inp.read(&ch, 1);
    if (inp.eof())
      break;
    int value = toascii(ch);
    big_int cipher;
    cipher = ModularExponentiation(value, ForeignPublicKey, ForeignModulo);
    out << cipher << " ";
  }

  inp.close();
  out.close();
}
```

2.6.10 Decryption()

Η Decryption() παίρνει τα ίδια ορίσματα με την Encryption() και, όπως ορίζει το πρότυπο, ακολουθεί την ίδια διαδικασία χρησιμοποιώντας το κρυφό κλειδί αντί του φανερού για να αναπαράξει το αρχικό κείμενο.

```
void RSA::Decryption(std::ifstream& inp, std::string cipherFile,
        std::ofstream& out, std::string decipherFile){
  inp.open(cipherFile);
  // destroy contents of these files (from previous runs, if any)
  out.open(decipherFile);
  out.close();

  // Check if files can be opened.
  if (!inp) {
    std::cout << "Error opening Cipher Text.\n";
    exit(1);
  }

  out.open(decipherFile);
  if (!out) {
    std::cout << "Error opening File.\n";
    exit(1);
  }

  while (1) {
    big_int cipherNum;
    std::stringstream ss;
    big_int decipher;
    int temp;

    if (!(inp >> cipherNum))
      break;

    decipher = ModularExponentiation(cipherNum, privateKey, modulo);
    ss << decipher;
    ss >> temp;
    out << (char)temp;
  }
  out.close();
  inp.close();
}
```

2.6.11 GetPublicKey()

Η συνάρτηση GetPublicKey() αποτελεί μια κλασική συνάρτηση getter που χρησιμοποιείται στις κλάσεις. Σκοπός της είναι όταν την καλούμε να μπορούμε να έχουμε πρόσβαση στα ιδιωτικά μέλη της συνάρτησης και στη συγκεκριμένη περίπτωση στη μεταβλητή που περιέχει την πληροφορία για το ιδιωτικό κλειδί.

2.6.12 GetModulo()

Η συνάρτηση GetModulo() επιστρέφει το modulo που χρησιμοποιήθηκε για την υλοποίηση του αλγορίθμου.

2.6.13 Testing του Κώδικα

Για τον έλεγχο της ορθής λειτουργίας της υλοποίησης μας δημιουργήσαμε έναν κώδικα για testing. Πιο συγκεκριμένα, χρησιμοποιήσαμε ένα δεύτερο αρχείο κώδικα το οποίο αποτελείται από τις εξής συναρτήσεις:

- testing()
- range_equal()
- compare_files()
- gen_random()

Συνάρτηση testing() Με τον κώδικα αυτό καλούμε τη συνάρτηση testing() η οποία παράγει ένα νέο αντικείμενο RSA με διαφορετικά κλειδιά κάθε φορά και τρέχει τον κώδικα κρυπτογράφησης και αποκρυπτογράφησης για τυχαία plain texts. Ειδικότερα, η συνάρτηση testing() αρχικά δημιουργεί ένα τυχαίο αλφαριθμητικό με την χρήση της gen_random(), το οποίο και θα αποτελέσει το plain text. Στη συνέχεια, πραγματοποιεί κρυπτογράφηση και αποκρυπτογράφηση καλώντας τις αντίστοιχες συναρτήσεις του κώδικα που τίθεται προς έλεγχο(testing) και ελέγχει αν το αποκρυπτογραφημένο μήνυμα είναι ίδιο με το αρχικό μήνυμα χρησιμοποιώντας την συνάρτηση compare_files(). Σημειώνουμε ότι η συνάρτηση testing() επαναλαμβάνει την παραπάνω διαδικασία για όσες επαναλήψεις πάρει ως όρισμα στο max_iter.

```
int testing(int max_iter){
  int correct_results = 0;
  std::ifstream inp;
  std::ofstream out;

  for(int i=0; i<max_iter; i++){
    //create rsa object
    RSA rsa;
    //create random plaintext
    std::ofstream outfile ("plain.txt");
    outfile << gen_random(15) << std::endl;
    outfile.close();

    // encryption stage
    rsa.Encryption(inp,"plain.txt", out, "cipher.txt");

    // decryption stage
    rsa.Decryption(inp, "cipher.txt", out, "decipher.txt");

    //compare plain.txt to decipher.txt
    if(compare_files("plain.txt", "decipher.txt")) correct_results++;

    //print wrong results every 100 tests
    if(i % 10 == 0) std::cout << "Number of correct results so far: "
      << correct_results << " out of " << i+1 << "\n";
  }
  return correct_results;
}
```

Συνάρτηση gen_random()
Η συνάρτηση gen_random() παίρνει ως όρισμα έναν ακέραιο και επιστρέφει μια τυχαία αλφαριθμητική συμβολοσειρά μήκους ίσο με το όρισμα της. Ειδικότερα, η συνάρτηση περιέχει έναν πίνακα με αλφαριθμητικά στοιχεία και με χρήση της εντολής rand() παράγει έναν τυχαίο αριθμό. Ο αριθμός αυτός αντιστοιχίζεται σε κάποιο αλφαριθμητικό σύμβολο που είναι αποθηκευμένο στον

πίνακα και έτσι επιλέγεται ο πρώτος χαρακτήρας. Κάνοντας ξανά την παραπάνω διαδικασία τόσες φορές όσο το μήκος της ζητούμενης συμβολοσειράς η συνάρτηση επιστρέφει την παραχθείσα συμβολοσειρά.

```
std::string gen_random(const int len) {
  std::string tmp_s;
  static const char alphanum[] =
      "0123456789"
      "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
      "abcdefghijklmnopqrstuvwxyz";

  tmp_s.reserve(len);

  for (int i = 0; i < len; ++i)
    tmp_s += alphanum[rand() % (sizeof(alphanum) - 1)];

  return tmp_s;
}
```

Συνάρτηση compare_files()

Η συνάρτηση compare_files() ελέγχει αν δύο αρχεία περιέχουν περιέχουν τα ίδια στοιχεία και επομένως αν είναι ίδια. Πιο συγκεκριμένα, η συνάρτηση παίρνει ως ορίσματα τα ονόματα των αρχείων που θέλουμε να ελέγξουμε και επιστρέφει True αν είναι ίδια, διαφορετικά επιστρέφει False. Η συνάρτηση δύο ifstream files και στη συνέχεια καλεί τη συνάρτηση range_equal για να ελέγξει αν τα αρχεία είναι ίδια.

```
bool compare_files(const std::string& filename1, const std::string& filename2){
  std::ifstream file1(filename1);
  std::ifstream file2(filename2);

  std::istreambuf_iterator<char> begin1(file1);
  std::istreambuf_iterator<char> begin2(file2);

  std::istreambuf_iterator<char> end;

  return range_equal(begin1, end, begin2, end);
}
```

Συνάρτηση range_equal()

Η συνάρτηση range_equal() παίρνει ως ορίσματα τους pointers που "δείχνουν" στην αρχή και στο τέλος των δύο αρχείων και συγκρίνει τα περιεχόμενα όλων των θέσεων από την αρχή έως το τέλος των αρχείων. Σε περίπτωση που κάποιο στοιχείο του ενός αρχείου δεν είναι ίσο με το στοιχείο του άλλου αρχείου ο έλεγχος διακόπτεται και η συνάρτηση επιστρέφει False υποδεικνύοντας ότι τα αρχεία δεν είναι ίδια. Διαφορετικά, η συνάρτηση συγκρίνει όλο το περιεχόμενο των αρχείων και επιστρέφει True, υποδεικνύοντας ότι τα αρχεία έχουν το ίδιο περιεχόμενο.

```
template<typename InputIterator1, typename InputIterator2>
bool range_equal(InputIterator1 first1, InputIterator1 last1,
                 InputIterator2 first2, InputIterator2 last2){
  while(first1 != last1 && first2 != last2){
      if(*first1 != *first2) return false;
      ++first1;
      ++first2;
  }
  return (first1 == last1) && (first2 == last2);
}
```

Στην παρούσα δοκιμή του κώδικα καλούμε τη συνάρτηση testing() με όρισμα $max\_iter = 1000$ και παίρνουμε το ακόλουθο μήνυμα επιτυχούς λειτουργίας για όλες τις επαναλήψεις:

```
Number of correct results so far: 651 out of 651
Number of correct results so far: 661 out of 661
Number of correct results so far: 671 out of 671
Number of correct results so far: 681 out of 681
Number of correct results so far: 691 out of 691
Number of correct results so far: 701 out of 701
Number of correct results so far: 711 out of 711
Number of correct results so far: 721 out of 721
Number of correct results so far: 731 out of 731
Number of correct results so far: 741 out of 741
Number of correct results so far: 751 out of 751
Number of correct results so far: 761 out of 761
Number of correct results so far: 771 out of 771
Number of correct results so far: 781 out of 781
Number of correct results so far: 791 out of 791
Number of correct results so far: 801 out of 801
Number of correct results so far: 811 out of 811
Number of correct results so far: 821 out of 821
Number of correct results so far: 831 out of 831
Number of correct results so far: 841 out of 841
Number of correct results so far: 851 out of 851
Number of correct results so far: 861 out of 861
Number of correct results so far: 871 out of 871
Number of correct results so far: 881 out of 881
Number of correct results so far: 891 out of 891
Number of correct results so far: 901 out of 901
Number of correct results so far: 911 out of 911
Number of correct results so far: 921 out of 921
Number of correct results so far: 931 out of 931
Number of correct results so far: 941 out of 941
Number of correct results so far: 951 out of 951
Number of correct results so far: 961 out of 961
Number of correct results so far: 971 out of 971
Number of correct results so far: 981 out of 981
Number of correct results so far: 991 out of 991
1000 out of 1000
```

2.7 Εφαρμογές

Ο κρυπτογραφικός αλγόριθμος RSA έχει εφαρμογή σε κάθε τύπο επικοινωνιών που απαιτεί ασφαλή αποστολή δεδομένων. Χρησιμοποιείται σε:

- Κρυπτογράφηση/Αποκρυπτογράφηση
- Ψηφιακή υπογραφή
- Ανταλλαγή κλειδιών

Η κρυπτογράφηση (encryption) είναι η διαδικασία κατά την οποία ο αποστολέας μετατρέπει την αρχική πληροφορία σε άλλη μορφή και μεταδίδει το προκύπτον ακατανόητο μήνυμα μέσω ενός ανοικτού δικτύου.Ο αποστολέας χρησιμοποιεί έναν αλγόριθμο κρυπτογράφησης και ένα κλειδί για τη μετατροπή του απλού κειμένου (αρχικού μηνύματος) σε κρυπτοκείμενο (κρυπτογραφημένο μήνυμα).Το απλό κείμενο (plaintext) είναι τα δεδομένα που πρέπει να προστατευθούν κατά τη διάρκεια της μετάδοσης.Το κρυπτοκείμενο (cipher text) είναι το κωδικοποιημένο κείμενο που παράγεται ως αποτέλεσμα του αλγόριθμου κρυπτογράφησης για τον οποίο χρησιμοποιείται ένα συγκεκριμένο κλειδί.Ο αλγόριθμος κρυπτογράφησης είναι ένας κρυπτογραφικός αλγόριθμος στον οποίο εισάγεται ένα απλό κείμενο και ένα κλειδί κρυπτογράφησης και παράγει ένα κρυπτογραφημένο κείμενο.

Η αποκρυπτογράφηση (decryption) αναστρέφει τη διαδικασία της κρυπτογράφησης για να μετατρέψει το μήνυμα στην αρχική του μορφή. Ο δέκτης χρησιμοποιεί έναν αλγόριθμο αποκρυπτογράφησης και ένα κλειδί για να μετατρέψει το κρυπτοκείμενο στο αρχικό, απλό κείμενο. Ο αλγόριθμος αποκρυπτογράφησης είναι μια μαθηματική διαδικασία που χρησιμοποιείται για την αποκρυπτογράφηση και παράγει το αρχικό απλό κείμενο ως αποτέλεσμα οποιουδήποτε δεδομένου κρυπτογραφημένου κειμένου και του κλειδιού αποκρυπτογράφησης. Είναι η αντίστροφη διαδικασία του αλγόριθμου κρυπτογράφησης.

Η ψηφιακή υπογραφή είναι ένα μαθηματικό σύστημα που χρησιμοποιείται για την απόδειξη της γνησιότητας ενός ψηφιακού μηνύματος ή εγγράφου. Μια έγκυρη ψηφιακή υπογραφή δίνει στον παραλήπτη την πιστοποίηση ότι το μήνυμα που δημιουργήθηκε ανήκει στον αποστολέα που το υπέγραψε ψηφιακά και ότι δεν αλλοιώθηκε-παραποιήθηκε κατά την μεταφορά. Οι ψηφιακές υπογραφές χρησιμοποιούν συνδυασμό μιας κρυπτογραφικής συνάρτησης κατατεμαχισμού (hash function) για δημιουργία της σύνοψης (hash) σε συνδυασμό με ασύμμετρη κρυπτογραφία για κρυπτογράφηση/αποκρυπτογράφηση σύνοψης (ο συνδυασμός σύνοψης και κρυπτογράφησης με ασύμμετρη κρυπτογραφία αποδεικνύει την ακεραιότητας του εγγράφου αλλά και την απόδειξη ταυτότητας του αποστολέα). Η ψηφιακή υπογραφή αποτελείται από 3 αλγορίθμους:

- Ο αλγόριθμος δημιουργίας δημόσιου και ιδιωτικού κλειδιού: Ο αλγόριθμος αυτός χρησιμοποιεί μια γεννήτρια τυχαίων αριθμών και με βάση αυτόν τον τυχαίο αριθμό δημιουργεί το δημόσιο και ιδιωτικό κλειδί (με το ιδιωτικό κλειδί δημιουργείται η ψηφιακή υπογραφή και με το δημόσιο κλειδί ελέγχεται η ψηφιακή υπογραφή).
- Ο αλγόριθμος προσθήκης ψηφιακής υπογραφής σε μηνύματα ή έγγραφα: Χρησιμοποιώντας το μήνυμα/έγγραφο και το ιδιωτικό κλειδί (το οποίο ανήκει μόνο σε αυτόν που υπογράφει το έγγραφο), δημιουργεί την ψηφιακή υπογραφή.
- Ο αλγόριθμος ελέγχου ψηφιακής υπογραφής μηνύματος ή εγγράφου: Χρησιμοποιώντας το μήνυμα/έγγραφο και το δημόσιο κλειδί (το δημόσιο κλειδί είναι διαθέσιμο σε όλους, και συσχετίζεται με το ιδιωτικό κλειδί και ανήκει σε αυτόν που υπέγραψε ψηφιακά το μήνυμα/έγγραφο), ελέγχει την αυθεντικότητα (ποιος το υπέγραψε) αλλά και ακεραιότητα (ότι το μήνυμα δεν παραποιήθηκε) του μηνύματος/εγγράφου.

Ένα πρόβλημα με τις ψηφιακές υπογραφές είναι ότι δεν γνωρίζουμε αν το δημόσιο κλειδί (κατά την διάρκεια ελέγχου της υπογραφής) που έχουμε ανήκει σε αυτόν που ισχυρίζεται ότι είναι. Για αυτό ακριβώς τον λόγο υπάρχει ο Πάροχος Υπηρεσιών Πιστοποίησης ο οποίος είναι ένας οργανισμός-οντότητα ο οποίος πιστοποιεί την σχέση ενός ανθρώπου με το δημόσιο κλειδί του. Ο Πάροχος Υπηρεσιών Πιστοποίησης θα πρέπει να εμπνέει εμπιστοσύνη γιατί είναι η αρχή η οποία εκδίδει ψηφιακά πιστοποιητικά. Τα ψηφιακά πιστοποιητικά ταυτοποιούν ένα δημόσιο κλειδί με τον δικαιούχο του. Πολλές φορές αυτός που υπογράφει ψηφιακά ένα ηλεκτρονικό έγγραφο, ενδέχεται να επισυνάψει στο έγγραφο μαζί με την ψηφιακή υπογραφή και το ψηφιακό πιστοποιητικό του δημόσιου κλειδιού.

Οι μηχανισμοί διαχείρισης κλειδιών (key management) και ανταλλαγής κλειδιών (key exchange), ασχολούνται με την ασφαλή παραγωγή, διανομή και αποθήκευση των κλειδιών κρυπτογράφησης. Η εύρεση απρόσβλητων μεθόδων διαχείρισης και ανταλλαγή κλειδιών είναι πολύ σημαντική στη διατήρηση της ασφάλειας της επικοινωνίας. Η έννοια της διαχείρισης κλειδιών αναφέρεται στα ασύμμετρα κρυπτοσυστήματα. Τα χαρακτηριστικά που πρέπει να έχει ένας μηχανισμός διαχείρισης κλειδιών είναι τα ακόλουθα. Οι χρήστες πρέπει να είναι σε θέση να μπορούν να αποκτήσουν με ασφάλεια ένα ζεύγος δημόσιου – ιδιωτικού κλειδιού που θα ικανοποιεί τις ανάγκες τους για προστατευμένη επικοινωνία.

Πρέπει να υπάρχει τρόπος αποθήκευσης και δημοσιοποίησης των δημόσιων κλειδιών, ενώ παράλληλα θα είναι δυνατή η ανάκτησή τους όποτε χρειάζεται. Επίσης τα δημόσια κλειδιά θα πρέπει να συσχετίζονται με σίγουρο τρόπο με την ταυτότητα του νόμιμου κατόχου. Έτσι,δεν θα μπορεί κάποιος να παρουσιάζεται σαν κάποιος άλλος, επιδεικνύοντας ένα ψεύτικο δημόσιο κλειδί. Τέλος οι χρήστες πρέπει να έχουν τη δυνατότητα να φυλάσσουν τις ιδιωτικές τους κλείδες με ασφάλεια, οι οποίες θα είναι έγκυρες μόνο για συγκεκριμένο χρονικό διάστημα. Η ανταλλαγή κλειδιών εφαρμόζεται στα συμμετρικά κρυπτοσυστήματα όπου οι δύο επικοινωνούντες χρήστες πρέπει να αποφασίσουν για το κοινό μυστικό κλειδί και έπειτα να αποκτήσουν από ένα αντίγραφο αυτού, χωρίς κανένας άλλος να μάθει για αυτό.

Μερικές ακόμα εφαρμογές του RSA είναι οι εξής:

- browsers
- εφαρμογές συνομιλιών
- emails
- VPN
- κάθε σύνδεση της μορφής https: URL
- για την επαλήθευση των software updates ότι γίνονται από τον γνήσιο developer
- τραπεζικά συστήματα
- ηλεκτρονικό εμπόριο
- αλγορίθμους ηλεκτρονικών υπογραφών
- κυβερνητικά έγγραφα

3 Cracking του αλγορίθμου

Στη βιβλιογραφία αναφέρεται πως η ασφάλεια του RSA βασίζεται εξ' ολοκλήρου στο πρόβλημα παραγοντοποίησης μεγάλων πρώτων αριθμών αλλά αυτό είναι απλώς μια εικασία. Δεν έχει ποτέ αποδειχτεί μαθηματικά ότι πρέπει να παραγοντοποιηθεί το N ώστε να υπολογιστεί το C από τα M και e. Αν και θεωρείται πιθανό να ανακαλυφθεί ένας εντελώς νέος τρόπος κρυπτανάλυσης του RSA υπολογίζοντας το d, η μέθοδος αυτή μπορεί να χρησιμοποιηθεί εξίσου για να παραγοντοποιηθούν μεγάλοι πρώτοι αριθμοί δίνοντας ένα νέο εργαλείο κρυπτογράφησης.Είναι επίσης δυνατή η επίθεση στον RSA μαντεύοντας το (p-1)(q-1) αλλά αυτού του είδους η επίθεση δεν είναι πιο εύκολη από τη παραγοντοποίηση του N. Κάποιες παραλλαγές του RSA έχουν αποδειχτεί τόσο δύσκολες στη κρυπτανάλυση όσο και το πρόβλημα της παραγοντοποίησης. Αποδεικνύεται ότι η ανάκτηση μερικών μόνο bits πληροφορίας είναι τόσο δύσκολη όσο και η αποκρυπτογράφηση ολόκληρου του μηνύματος.

Η παραγοντοποίηση του N είναι η πιο προφανής μέθοδος επίθεσης. Ο κάθε επιτιθέμενος θα έχει στη διάθεση του το δημόσιο κλειδί e και το modulus N, οπότε για να βρει το κλειδί αποκρυπτογράφησης d, θα πρέπει να παραγοντοποιήσει το N. Επειδή η τεχνολογία μπορεί ήδη να υπολογίσει ένα 129 δεκαδικών ψηφίων modulus, το N θα πρέπει να είναι μεγαλύτερο από αυτό. Είναι φυσικά πιθανό για ένα κρυπταναλυτή να δοκιμάσει κάθε πιθανό d ώσπου να βρει το σωστό αλλά αυτή η εξαντλητική (brute-force) επίθεση είναι λιγότερο αποδοτική ακόμα και από τη παραγοντοποίηση του N.

Κατά καιρούς έχουν εμφανιστεί επιστήμονες που υποστηρίζουν ότι έχουν βρει εύκολους τρόπους επίθεσης στον RSA αλλά μέχρι στιγμής δεν έχει βρεθεί κάτι αποτελεσματικό αναφορικά με τον χρόνο. Για παράδειγμα, σε εργασία του William Payne προτείνεται μια μέθοδος που βασίζεται στο μικρό θεώρημα του Fermat η οποία δυστυχώς αποδεικνύεται πιο αργή και από τη παραγοντοποίηση του modulus. Ένα άλλο σημείο ανησυχίας είναι πιο κοινοί αλγόριθμοι υπολογισμού πρώτων αριθμών είναι πιθανοτικοί και με τον τρόπο αυτό είναι δυνατό να ελαχιστοποιηθεί η πιθανότητα ένας εκ των p ή q να είναι σύνθετος. Αν αυτό συμβεί, τότε η κρυπτογράφηση ή η αποκρυπτογράφηση δεν θα δουλεύουν σωστά. Υπάρχουν μερικοί αριθμοί που ονομάζονται αριθμοί Carmichael τους οποίους αδυνατούν να υπολογίσουν /εντοπίσουν πιθανοτικές γεννήτριες τυχαίων αριθμών. Αν και αυτοί οι αριθμοί είναι επισφαλείς αν χρησιμοποιηθούν, είναι εξαιρετικά σπάνιοι.

Παράμετροι:	n γινόμενο δύο πρώτων αριθμών p και q (πρέπει να παραμείνουν κρυφοί)
Δημόσιο Κλειδί:	e σχετικά πρώτος με το (p-1)(q-1)
Ιδιωτικό Κλειδί:	$d=e^{-1} \bmod ((p-1)(q-1))$
Κρυπτογράφηση:	$C=M^e \bmod N$
Αποκρυπτογράφηση:	$M=C^d \bmod N$

Εικόνα 7: Κρυπτογράφηση RSA

Η πιο προφανής μέθοδος για την εύρεση των παραγόντων ενός αριθμού n είναι η διαίρεση του αριθμού αυτού με κάθε πρώτο αριθμό ο οποίος είναι μικρότερος από $\sqrt{n}$. Παρόλα αυτά λόγω του μεγάλου αριθμού των πρώτων αριθμών μεταξύ 2 και $\sqrt{n}$ η διαδικασία αυτή μπορεί να πάρει πάρα πολύ χρόνο ακόμα και για αριθμούς με λιγότερα από 10 ψηφία.

3.1 Θεώρημα πρώτων αριθμών

Έστω ότι π(m) είναι το σύνολο των πρώτων αριθμών που είναι μικρότεροι ή ίσοι του αριθμού m. Τότε το όριο:

$$\lim_{m\to\infty} \frac{\pi(m)}{m/ln(m)} = 1$$

Έτσι μπορούμε να πούμε ότι το π(m) είναι ασυμπτωτικά ίσο με το $\frac{m}{ln(m)}$ Με άλλα λόγια αν θεωρήσουμε ότι $m = \sqrt{n}$ τότε μπορούμε να υπολογίσουμε τον αριθμό των πρώτων αριθμών με τους οποίους θα πρέπει να διαιρέσουμε το n.

Ας υποθέσουμε ότι το n=1000000000000, ένας αριθμός με μόλις 13 ψηφία. Τότε η ρίζα του αριθμού είναι ίση με 1000000. Οπότε $\pi(n) = \frac{n}{ln(n)} = 72382$. Αυτό σημαίνει ότι θα πρέπει να διαιρέσουμε το n με τους πρώτους περίπου 72382 αριθμούς ώστε να είμαστε σίγουροι ότι βρήκαμε έναν παράγοντα του n. Αυτή η μέθοδος ονομάζεται trial division. Αυτός σίγουρα δεν είναι ένας αποδοτικός τρόπος για να βρούμε τους παράγοντες ενός αριθμού δεδομένου ότι πιθανόν για να σπάσουμε έναν RSA αλγόριθμο θα έχουμε να κάνουμε με αριθμούς με εκατοντάδες ψηφία.

3.2 Πολυπλοκότητα Big O

Η συνάρτηση $f = O(g)$ είναι ένα μέτρο της πολυπλοκότητας ενός αλγορίθμου. Για την ακρίβεια μας δείχνει με τι ρυθμό αυξάνονται οι υπολογισμοί που απαιτούνται για μια εργασία, όσο αυξάνονται τα δεδομένα που εισάγουμε, ή αντίστοιχα, ο χώρος που απαιτείται στην μνήμη. Ένας αλγόριθμος τρέχει σε πολυωνυμικό χρόνο αν ο αριθμός των διεργασιών που απαιτούνται είναι $O(k^c)$ όπου k είναι ο αριθμός των bits εισόδου. Κανένας από τους αλγόριθμους εύρεσης παραγόντων που έχουν προταθεί μέχρι σήμερα δεν υλοποιείται σε πολυωνυμικό χρόνο. Οι περισσότεροι εκτελούνται σε εκθετικό χρόνο που σημαίνει ότι ο αριθμός των διεργασιών που γίνονται για την εύρεση των παραγόντων ενός αριθμού με k ψηφία έχει πολυπλοκότητα $O(e^{ck})$ για έναν αριθμό c. Αυτό πρακτικά σημαίνει ότι για μια μεγάλη τιμή του k ο αλγόριθμος θα ολοκληρωθεί σε άπειρο χρόνο και απαιτεί τεράστια επεξεργαστική ισχύ. Σήμερα για να είναι ένας RSA αλγόριθμος ασφαλής προτείνεται το λιγότερο να έχει 1024 bits και για εξαιρετική ασφάλεια προτείνεται ένα κλειδί με 2048 bits.

3.3 Αλγόριθμοι γενικοί και ειδικού σκοπού

Οι γενικού σκοπού αλγόριθμοι έχουν χρόνους εκτέλεσης οι οποίοι βασίζονται μόνο στο μέγεθος του n, δηλαδή του αριθμού του οποίου θέλουμε να βρούμε τους παράγοντες. Αλγόριθμοι γενικού σκοπού είναι οι:

- Αλγόριθμος του Dixon
- Quadratic Sieve
- Multipolynomial Quadratic Sieve
- General Number Field Sieve

Από την άλλη, οι αλγόριθμοι ειδικού σκοπού έχουν χρόνους εκτέλεσης που βασίζονται στους παράγοντες p και q ενός αριθμού n. Για παράδειγμα, υπάρχουν αλγόριθμοι στους οποίους το p πρέπει να έχει μια συγκεκριμένη μορφή όπως στον αλγόριθμο "Polland p-1". Σε αυτόν πρέπει ο αριθμός p-1 πρέπει να είναι τέτοιος ώστε να διαιρείται από μικρούς πρώτους αριθμούς αριθμούς. Αλγόριθμοι ειδικού σκοπού είναι οι:

- Trial Devision
- Polland's Rho
- Polland's p-1
- Μέθοδος Fermat's Factorization

- Μέθοδος Lenstra's Elliptic Curve Factorization
- Special Number Field Sieve

3.3.1 Μέθοδος Polland's Rho

Η μέθοδος Rho ή αλλιώς p μέθοδος είναι ένας αλγόριθμος εύρεσης παραγόντων ο οποίος είναι πολύ αποτελεσματικός όταν εφαρμόζεται σε σύνθετους (composite) αριθμούς που έχουν μικρούς παράγοντες. Τα βήματα υλοποίησής του είναι τα εξής:

1. Επιλέγουμε ένα πολυώνυμο το οποίο δεν είναι ένα προς ένα. Συνήθως επιλέγουμε πολυώνυμα της μορφής $f(x) = x^2 + c$.
2. Επιλέγουμε έναν ακέραιο $x_0 = 0$
3. Θεωρούμε $x_i + 1 = f(x_i)$ για i=0,1,2,...
4. Βρίσκουμε τον Μέγιστο Κοινό Διαιρέτη (GCD) $(x_j - x_k, n)$ μέχρι να βρούμε μια λύση τέτοια ώστε $1 < m < n$. Έτσι το m είναι ένας παράγοντας του n και ο αλγόριθμος τερματίζεται. Διαφορετικά επιλέγουμε ένα νέο πολυώνυμο f και ένα νέο σημείο x_0 και επαναλαμβάνουμε τον αλγόριθμο.

Το 11 είναι πρώτος αριθμός οπότε επιλύσαμε το πρόβλημα. Σε αυτή τη μέθοδο υπάρχει μεγάλη πιθανότητα να βρούμε έναν παράγοντα του n σε $O(e^{4\sqrt{n}ln^3(n)})$. Αυτή η μέθοδος είναι πιο αποτελεσματική από την trial division που παρουσιάστηκε αλλά λιγότερο αποτελεσματική από μεθόδους όπως η μέθοδος p-1.

Παράδειγμα: Έστω ότι θέλουμε να βρούμε τους παράγοντες του αριθμού n=187. Επιλέγουμε τυχαία σημείο εκκίνησης $x_0 = 2$ και c=1. Άρα έχουμε το πολυώνυμο $f(x) = x^2 + 1$. Προκύπτει ο πίνακας:

$x_{i+1}=f(x_1)$	$x_{i+1}=f(f(y_1))$	c	d=GCD(|x-y|, n)
5	26	1	1
26	180	1	11

Εικόνα 8: Αλγόριθμος Polland's Rho

Το 11 και το $\frac{187}{11} = 17$ είναι πρώτοι αριθμοί οπότε επιλύσαμε το πρόβλημα.

3.3.2 Μέθοδος Polland's p-1

Έστω B ένας θετικός ακέραιος. Ένας ακέραιος αριθμός n ονομάζεται B-ομαλός (B-smooth) αν όλοι οι πρώτοι παράγοντές του είναι μικρότεροι ή ίσοι του B. Τότε το B ονομάζεται όριο ομαλότητας του n. Παρόλο που η μέθοδος p-1 είναι πιο γρήγορη από την μέθοδο Rho έχει το μειονέκτημα ότι βρίσκει τον παράγοντα p ενός σύνθετοι αριθμού στην μορφή $n = c^k$ αν ο k διαιρείται με p-1 και αν ο p-1 είναι ομαλός. Τα βήματα υλοποίησης είναι τα εξής:

1. Επιλέγουμε τυχαία ένα όριο ομαλότητας B.

2. Υπολογίζουμε το γινόμενο όλων των πρώτων αριθμών που είναι μικρότεροι ίσοι με το B αφού υψώσουμε τον κάθε πρώτο αριθμό στην δύναμη $log_q B$. Το αποτέλεσμα το συμβολίζουμε με το γράμμα M.

3. Επιλέγουμε τυχαία έναν σχετικά πρώτο (coprime) αριθμό του n (οι α, β λέγονται σχετικά πρώτοι αριθμοί αν ισχύει ότι μέγιστος κοινός διαιρέτης τους είναι το 1)

4. Βρίσκουμε τον ΜΚΔ των αριθμών $(a^m - 1, n)$ και το συμβολίζουμε ως g.

5. Αν 1¡g¡n τότε το αποτέλεσμα είναι το g.

6. Αν το g=1 τότε επιλέγουμε ένα μεγαλύτερο B και πάμε πάλι στο βήμα 2.

7. Αν το g=n τότε επιλέγουμε ένα μικρότερο B και πάμε πάλι στο βήμα 2.

Παράδειγμα: Έστω ότι θέλουμε να βρούμε τους παράγοντες του αριθμού n=299.

1. Επιλέγουμε B=5.

2. $M = 2^2 3^1 5^1$.

3. Επιλέγουμε a=2.

4. g=ΜΚΔ$((a^m - 1, n))$=13.

5. Αν 1i̊3ż99 οπότε ο ένας παράγοντας είναι το 13.

6. $\frac{299}{13} = 23$ που είναι πρώτος αριθμός.

Η πολυπλοκότητα του αλγορίθμου είναι $O(Bln(B)(ln(n))^2 + (ln(n))^3)$.

3.3.3 Αλγόριθμος παραγοντοποίησης του Fermat

Ο αλγόριθμος αυτός βασίζεται στην αναπαράσταση ενός περιττού αριθμού ως την διαφορά δύο τετραγώνων. Για έναν ακέραιο n ισχύει ότι: $n = a^2 - b^2 = (a+b)(a-b)$. Οι αριθμοί a και b είναι οι παράγοντες του αριθμού n. Η πρόταση του αλγορίθμου είναι να υπολογίσουμε τους a και b δοκιμάζοντας για διάφορες τιμές $\sqrt{n}, \sqrt{n}+1, \sqrt{n}+2, ...$ μέχρι να βρούμε κάποιον αριθμό τέτοιον ώστε το τετράγωνό του μείον το n να μας δίνουν τέλειο τετράγωνο.

Παράδειγμα: Έστω ότι θέλουμε να βρούμε τους παράγοντες του αριθμού n=6557. Στην αρχή δοκιμάζουμε βρίσκοντας την ρίζα του αριθμού n που είναι το 81. Στη συνέχεια βρίσκουμε το b:

$$b^2 = 81^2 - 6557 = 4$$

Είναι τέλειο τετράγωνο. Άρα b=2. Έτσι οι παράγοντες του 6557 είναι:

$$(a-b) = 81 - 2 = 79$$
$$(a+b) = 81 + 2 = 83$$

3.3.4 Παραγοντοποίηση με ελλειπτικές καμπύλες

Ο αλγόριθμος RSA βασίζεται στην δυσκολία να παραγοντοποιήσουμε μεγάλους ακέραιους αριθμούς. Η μέθοδος αυτή έχει κάποιες ομοιότητες με την μέθοδο Polland. Δημιουργός της είναι ο Hendrik Lenstra. Έχει τα ακόλουθα βήματα:

1. Επιλέγουμε ένα όριο καμπυλότητας B.
2. Βρίσκουμε το ελάχιστο κοινό πολλαπλάσιο των αριθμών 1,..,B.
3. Επιλέγουμε μια ελλειπτική τροχιά με εξίσωση $y^2 = x^3 + ax + b$ και ένα σημείο της $P(x_0, y_0)$. Η κλίση της δίνεται από τον τύπο: $s = \frac{3x^2+a}{2y}$. Η κλίση μεταξύ δύο σημείων P και Q είναι $s = \frac{P_y - Q_y}{P_x - Q_x}$.
4. Επιχειρούμε να υπολογίσουμε τις δυνάμεις mP. Αν κάπου αποτύχουμε να υπολογίσουμε μια δύναμη αυτό οφείλεται στο ότι κάποιος παρανομαστής Π στους τύπους πρόσθεσης σημείων είναι διαιρετός με n. Αν ο ΜΚΔ (Π,n)¡n τότε το (Π,n) είναι ένας τετριμμένος διαιρέτης του n και η μέθοδος έχει ολοκληρωθεί.
5. Διαφορετικά δοκιμάζουμε με διαφορετική ελλειπτική καμπύλη.

3.3.5 Αλγόριθμος του Dixon

Η μέθοδος παραγοντοποίησης ενός σύνθετου αριθμού του Dixon βασίζεται σε ένα πολύ γνωστό μαθηματικό θεώρημα:

Εάν $x^2 \equiv y^2 (mod n) \mu \varepsilon x \neq \pm y (mod n)$ τότε είναι πιθανό ο ΜΚΔ(x-y,n) να είναι παράγοντας του n.

Παράδειγμα: Έστω ότι θέλουμε να βρούμε τους παράγοντες του αριθμού n=84923. Ξεκινάμε από το 292 που είναι ο πρώτος κατά ένα μεγαλύτερος αριθμός από την τετραγωνική ρίζα του n. Ισχύει ότι:

$$5052 \bmod 84923 = 256 = 16^2$$
$$\text{Έτσι } (505\text{-}16)(505+16) = 0 \bmod 84923$$

Ο ΜΚΔ((505-16),n) είναι ο 163 που είναι ένας παράγοντας του n.

3.4 Κβαντικοί υπολογιστές

Κβαντικός υπολογιστής ονομάζεται μία υπολογιστική συσκευή που εκμεταλλεύεται χαρακτηριστικές ιδιότητες της κβαντομηχανικής, όπως την αρχή της υπέρθεσης και της διεμπλοκής καταστάσεων, για να φέρει εις πέρας επεξεργασία δεδομένων και εκτέλεση υπολογισμών. Η εξέταση της λειτουργίας των κβαντικών υπολογιστών και η διατύπωση κατάλληλων αλγορίθμων από τη σκοπιά της θεωρητικής πληροφορικής, είναι ένα σύγχρονο ακαδημαϊκό πεδίο με τίτλο κβαντικός υπολογισμός. Οι κβαντομηχανικές ιδιότητες και αρχές λειτουργίας των κβαντικών υπολογιστών μελετώνται και από την επιστήμη της φυσικής. Η σχετική πρακτική τεχνολογία είναι ακόμα στα πολύ πρώιμα στάδια ανάπτυξης.

Σε έναν συμβατικό ψηφιακό υπολογιστή (κατά κανόνα ηλεκτρονικό), στοιχειώδης μονάδα πληροφορίας είναι το bit, ενώ σε έναν κβαντικό υπολογιστή το qubit. Η βασική αρχή της κβαντικής υπολογιστικής επιστήμης είναι το γεγονός ότι οι κβαντομηχανικές ιδιότητες της ύλης μπορούν να χρησιμοποιηθούν για την αναπαράσταση και τη δόμηση δεδομένων, καθώς και το γεγονός ότι

μπορούν να επινοηθούν και να κατασκευαστούν μηχανισμοί στηριγμένοι στην κβαντομηχανική για την επεξεργασία αυτών των δεδομένων. Αν και οι κβαντικοί υπολογιστές βρίσκονται ακόμα σε πειραματικό στάδιο, τα αποτελέσματα των σχετικών πειραμάτων με μικρό πλήθος από qubit) είναι ενθαρρυντικά.

Μεγάλης κλίμακας κβαντικοί υπολογιστές αναμένεται να λύνουν προβλήματα πολύ ταχύτερα από τους κλασικούς υπολογιστές, χρησιμοποιώντας τους καλύτερους μέχρι τώρα γνωστούς αλγόριθμους, όπως η παραγοντοποίηση μεγάλων αριθμών χρησιμοποιώντας τον αλγόριθμο του Shor ή η προσομοίωση μεγάλων συστημάτων. Αν δοθούν αρκετοί υπολογιστικοί πόροι σε έναν κλασικό υπολογιστή, μπορεί να προσομοιώσει οποιοδήποτε κβαντικό αλγόριθμο. Ωστόσο η υπολογιστική ισχύς 500 qubit, για παράδειγμα, θα ήταν ήδη πολύ μεγάλη για να αναπαρασταθεί σε έναν κλασικό υπολογιστή γιατί θα χρειαζόταν να αποθηκευτούν 2500 τιμές (ένα terabyte πληροφορίας μπορεί να αποθηκεύσει 243 διακριτές τιμές).

Η μνήμη ενός κλασικού ψηφιακού υπολογιστή αποτελείται από bit τα οποία μπορούν να αναπαραστήσουν την τιμή 1 ή 0. Ένα qubit μπορεί να αναπαραστήσει την τιμή 1, 0 ή οποιαδήποτε υπέρθεση αυτών των 2. Δύο qubits μπορούν να αναπαραστήσουν οποιαδήποτε υπέρθεση τεσσάρων δυνατών καταστάσεων, 3 qubits οποιαδήποτε υπέρθεση 8 καταστάσεων. Γενικά ένας κβαντικός υπολογιστής με n qubits μπορεί να βρίσκεται σε αυθαίρετη υπέρθεση των έως 2n δυνατών καταστάσεων ταυτόχρονα, ενώ ένας κλασικός υπολογιστής μπορεί να βρίσκεται μόνο σε μια από αυτές τις καταστάσεις κάθε στιγμή. Ο κβαντικός υπολογιστής λειτουργεί θέτοντας τα qubits σε μια ελεγχόμενη αρχική κατάσταση που αναπαριστά το αρχικό πρόβλημα και χειρίζεται τα qubits χρησιμοποιώντας λογικές κβαντικές πύλες. Η αλληλουχία των πυλών που χρησιμοποιούνται ονομάζεται κβαντικός αλγόριθμος.

Ένα παράδειγμα εφαρμογής των qubits σε έναν κβαντικό υπολογιστή θα ξεκινούσε με την χρήση σωματιδίων με δύο καταστάσεις περιστροφής (spin): πάνω και κάτω. Στην πραγματικότητα οποιοδήποτε σύστημα έχει μια ποσότητα A που μπορεί να παρατηρηθεί, η οποία διατηρείται με την εξέλιξη του χρόνου και είναι τέτοια ώστε η A να έχει τουλάχιστον δύο διακριτές και επαρκώς κατανεμημένες διαδοχικές ιδιοτιμές, είναι κατάλληλο για να υλοποιήσει ένα qubit. Αυτό συμβαίνει επειδή ένα τέτοιο σύστημα μπορεί να χαρτογραφηθεί πάνω σε ένα αποτελεσματικό σύστημα με περιστροφή 1/2 (spin-1/2).

Εικόνα 9: Κβαντικός υπολογιστής της IBM

Στην εικόνα βλέπουμε έναν υπερσύγχρονο κβαντικό υπολογιστή της εταιρίας IBM. Η εταιρία δίνει δωρεάν online πρόσβαση στον κβαντικό υπολογιστή της σε όλους με περιορισμένο όμως αριθμό qubits.

Στον τομέα της ψηφιακής ασφάλειας υπάρχουν πολλές προκλήσεις. Ένα σημαντικό πλεονέκτημα των κβαντικών υπολογιστών έναντι των παραδοσιακών υπολογιστών είναι η ικανότητα υπολογισμών με πολύ μεγάλους αριθμούς – μια λειτουργία που είναι εξαιρετικά δύσκολη για τους τωρινούς ηλεκτρονικούς υπολογιστές και γι' αυτό έχει αναδειχθεί σε βάση όλων των μηχανισμών κρυπτογράφησης.Ένας προχωρημένος κβαντικός υπολογιστής, αντιθέτως, θα μπορεί να επιλύει παρεμφερή μαθηματικά προβλήματα με τρομακτική ισχύ. Γι' αυτόν το λόγο και η προοπτική της κβαντικής υπολογιστικής απαιτεί μια ριζική αναδιαμόρφωση της κρυπτογραφίας.

Η σημερινή κρυπτογράφηση μπορεί σύντομα να ανήκει ολοκληρωτικά στο παρελθόν. Επιστήμονες όπως ο Michele Mosca, συνιδρυτής του Ινστιτούτου Κβαντικής Υπολογιστικής στο Οντάριο, εκτιμούν πως η πρόοδος στην κβαντική υπολογιστική θα έχουν καταστήσει άχρηστο το RSA-2048, ένα κοινό πρότυπο κρυπτογράφησης, μέχρι το 2026. Κρίνεται αναγκαίο να αρχίσουμε από τώρα να αναπροσαρμόζουμε τις ψηφιακές μας άμυνες. Όπως γράφει ο Adam Langley, μηχανικός λογισμικού στη Google, «είναι επικίνδυνο» να επαναπαυτούμε περιμένοντας κάποια καθοδήγηση γύρω από τα πρότυπα· δεν υπάρχει καιρός για χάσιμο.

Όπως επισημαίνει και μια πρόσφατη εργασία ερευνητών από τις Εθνικές Ακαδημίες Επιστημών, Μηχανικής και Ιατρικής, οι προετοιμασίες για την αντιμετώπιση μιας επερχόμενης διάλυσης της παραδοσιακής κρυπτογράφησης από τους κβαντικούς υπολογιστές πρέπει να ξεκινήσουν όσο το δυνατόν ταχύτερα. Όπως τονίζεται στο σχετικό δελτίο τύπου, η ανάπτυξη νέων αλγόριθμων είναι κρίσιμη και πρέπει να γίνει από τώρα.

Ένας από τους πιο διαδεδομένους αλγορίθμους για το σπάσιμο του κρυπτογραφικού αλγόριθμου RSA είναι ο κβαντικός αλγόριθμος του Shor.Ο κβαντικός αλγόριθμος του Shor μπορεί να χρησιμοποιηθεί για την εύρεση της περιόδου περιοδικών συναρτήσεων και για την ανάλυση ενός αριθμού σε γινόμενο πρώτων παραγόντων.

3.4.1 Κβαντικός αλγόριθμος του Shor

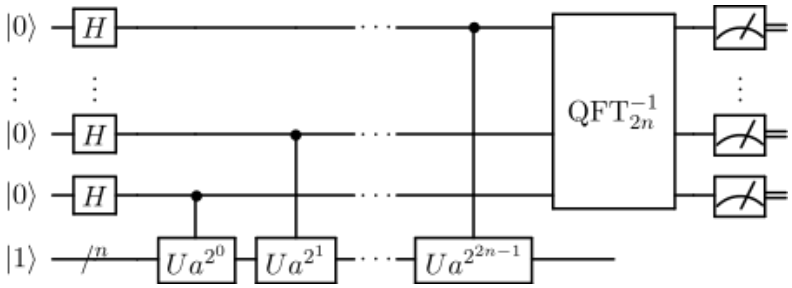
Ο κβαντικός αλγόριθμος του Shor αρχίζει με δύο κβαντικούς καταχωρητές. Ο πρώτος ονομάζεται Reg1 και ο δεύτερος Reg2. Οι δύο κβαντικοί καταχωρητές αποτελούν έναν κβαντικό καταχωρητή που τον ονομάζουμε Reg. Αν η κατάσταση του Reg1 είναι $|\psi_1>$ και η κατάσταση του Reg2 είναι $|\psi_2>$ τότε η κατάσταση του reg, η ψ δίνεται από τον τύπο:

$$|\psi>=|\psi_1>|\psi_2>=|\psi_1\psi_2>$$

Για να αναλύσουμε έναν ακέραιο αριθμό n σε γινόμενο δύο πρώτων αριθμών θα πρέπει να υπολογίσουμε την περίοδο συνάρτησης $f_{n,a}(x) = a^x(modn)$. Τα βήματα του κβαντικού αλγορίθμου του Shor είναι τα εξής:

1. Επιλέγουμε έναν ακέραιο αριθμό N, ελέγχουμε αν ο N είναι σύνθετος. Αν δεν είναι σταματάμε.
2. Επιλέγουμε τυχαία έναν αριθμό α που είναι μεγαλύτερος από 1 και μικρότερος από N.
3. Αν b=gcd(a,N)Î το αποτέλεσμα είναι το b και σταματάμε.
4. Υπολογίζουμε την περίοδο r έτσι ώστε $a^r = 1modN$.
5. Αν ο r είναι περιττός πάμε στο βήμα 2.
6. Υπολογίζουμε $x = (a^{r/2} + 1)modN$ και $y = (a^{r/2} - 1)modN$.
7. Αν x=0 πάμε στο βήμα 2. Αν y=0 κάνουμε $r = r/2$ και πάμε στο βήμα 5.
8. Υπολογίζουμε p=gcd(x,N) και q=gcd(y,N). Τουλάχιστον ένας από αυτούς τους δύο αριθμούς είναι παράγοντας του N.

Η υλοποίησή του με κβαντικές πύλες είναι η ακόλουθη:



Εικόνα 10: Κβαντικός αλγόριθμος του Shor

Ο αλγόριθμος του Shor έχει πολυωνυμικό χρόνο εκτέλεσης. Είναι εκθετικά πιο γρήγορος σχεδόν από όλους τους κλασσικούς αλγορίθμους. Η αποδοτικότητά του ευθύνεται κυρίως στον κβαντικό μετασχηματισμό Fourier. Με την χρήση ενός ιδανικού κβαντικού υπολογιστή με μεγάλο αριθμό από qubits είναι εφικτό να σπάσουμε ένα κλειδί του RSA σε πολυωνυμικό χρόνο.

4 Βιβλιογραφικές αναφορές

1. Aya Furutani, Yese, Kurt, Christopher Towse, "RSA Cryptography Cracking the Code", Scripps College, 2006, doi:10.1.1.113.5632
2. Moretti, Valter, "Spectral Theory and Quantum Mechanics", 2017
3. Sushanta Kumar Sahu, Manoranjan Pradhan "FPGA Implementation of RSA Encryption System"
4. Sarthak R Patel, Prof. Khushbu Shah, Gaurav R Patel "Study on Improvements in RSA Algorithm"
5. Skobic, Dokic, Ivanovic, "Hardware Modules of the RSA Algorithm", 2014, doi:10.2298
6. Speeding up modular exponentiation using CRT
7. Decrypting RSA using CRT
8. Speeding up modular exponantiation using fast powering algorithm
9. Fast powering algorithm
10. Exponantiation by squearing algorith
11. Montgomery modular multiplication
12. An Introduction to Modular Math
13. Padding
14. How a quantum computer could break 2048-bit RSA

Η συνολική υλοποίηση μπορεί να βρεθεί στο Github Repo.