ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΑΣΦΑΛΕΙΑ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Πολλαπλασιασμός πινάκων με τη χρήση ομομορφικής κρυπτογραφίας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

των

Ειρήνης Παπαδοπούλου και Χρήστου Βαρελά

Επιβλέπων : Παναγιώτης Ριζομυλιώτης

Μέλη εξεταστικής επιτροπής: Μαρία Καρύδα, Σπυρίδων Κοκολάκης

Σάμος, [Οκτώβριος 2022]

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πρόλογος και ευχαριστίες

Αυτή η εργασία σηματοδοτεί το τέλος της φοίτησης μας στο πρόγραμμα μεταπτυχιακών σπουδών «Ασφάλεια Πληροφοριακών και Επικοινωνιακών Συστημάτων» του Πανεπιστημίου Αιγαίου, η οποία ξεκίνησε τον Οκτώβριο του 2019 και μας χάρισε νέες εμπειρίες και σημαντικές γνώσεις στον χώρο της ασφάλειας υπολογιστών. Ως επαγγελματίες στον χώρο της ανάπτυξης λογισμικού, χάρη στις μεταπτυχιακές σπουδές μας μπορούμε πλέον να δούμε τον κλάδο από διαφορετική οπτική και να εφαρμόσουμε στην πράξη τις γνώσεις και δεξιότητες που αποκτήσαμε. Ελπίζουμε η παρούσα εργασία, δηλαδή η ανάλυση, ο κώδικας και τα ευρήματα μας, να βρεθεί χρήσιμη σε σπουδαστές που θα ασχοληθούν με το αντικείμενο της ομομορφικής κρυπτογραφίας.

Θα θέλαμε να ευχαριστήσουμε πρώτα απ'όλα τον επιβλέπων καθηγητή μας κ. Ριζομυλιώτη, ο οποίος ανέλαβε το έργο της καθοδήγησής μας για τον σκοπό της εργασίας, και μας βοήθησε καθ'όλη την διάρκεια της. Έπειτα, ευχαριστούμε τις οικογένειες μας, οι οποίες μας στήριξαν από την αρχή μέχρι το τέλος των σπουδών μας. Ευχαριστούμε επίσης τους καθηγητές και συμφοιτητές του μεταπτυχιακού προγράμματος, καθώς μαζί πορευτήκαμε, δουλέψαμε και ανακαλύψαμε νέους ορίζοντες.

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας περιεχομένων

Λίστα Σχημάτων

Λίστα Πινάκων

Ακρωνύμια

SIMD	Single Instruction Multiple Data
MPC	Multi Party Computation
2PC	Two Party Computation
SFE	Secure Function Evaluation
HE	Homomorphic Encryption
PHE	Partially Homomorphic Encryption
SWHE	Somewhat Homomorphic Encryption
FHE	Fully Homomorphic Encryption
LFHE	Leveled Fully Homomorphic Encryption
LWE	Learning With Errors
RLWE	Ring Learning With Errors
NIZK	Non-Interactive Zero Knowledge
CKKS	Cheon-Kim-Kim-Song
JKLS	Jiang, Kim, Lauter and Song

Περίληψη

Η κρυπτογραφία, στην εποχή μας, χρησιμοποιείται ευρέως για την προστασία των δεδομένων των χρηστών και την αποτροπή υποκλοπών. Ένα από τα ελαττώματα των τωρινών συστημάτων κρυπτογράφησης είναι ότι δεν προσφέρουν απόλυτη ασφάλεια σε περίπτωση που ευαίσθητα δεδομένα χρειαστεί να σταλθούν σε μια μη έμπιστη υπηρεσία για υπολογισμούς. Η ομομορφική κρυπτογραφία επιτρέπει σε παρόχους υπηρεσιών να εκτελούν συγκεκριμένους υπολογισμούς σε κρυπτογραφημένα δεδομένα χωρίς την αποκρυπτογράφηση τους, ενώ διατηρείται το απόρρητο των χρηστών. Αυτό το νέο σύστημα κρυπτογράφησης μπορεί να έχει εφαρμογές σε διάφορους τομείς όπως η υγειονομική περίθαλψη και οι ιατρικές εφαρμογές. Από τότε που δημιουργήθηκε το πρώτο σύστημα πλήρους ομομορφικής κρυπτογράφησης το 2009, έχουν προκύψει πολλά ακόμη συστήματα, επιδιώκοντας διαρκώς βελτιώσεις.

Σε αυτή την εργασία αναλύουμε τρεις προσεγγίσεις πολλαπλασιασμού πινάκων, και τις υλοποιούμε σε γλώσσα C++ με την χρήση της βιβλιοθήκης Microsoft SEAL. Καθώς το πλέον πιο σημαντικό ελάττωμα της ομομορφικής κρυπτογράφησης είναι η απόδοση, στο πρακτικό κομμάτι της εργασίας γίνονται μετρήσεις χώρου και χρόνου με σκοπό την σύγκριση των τριών προσεγγίσεων.

Λέξεις Κλειδιά: *κρυπτογραφία, πολλαπλασιασμός πινάκων, απόδοση, ομομορφισμός, ομομορφική κρυπτογράφηση*

Abstract

Cryptography nowadays is widely used to protect user data and prevent eavesdropping. One of the drawbacks of current encryption systems is that they do not offer absolute security in case sensitive data needs to be sent to an untrusted service in order to process it. Homomorphic encryption allows service providers to perform specific computations on encrypted data without decrypting it, and thus maintaining the privacy of the users. This new encryption system may prove very useful in various fields such as healthcare and medical applications. Since the first full homomorphic encryption system was created in 2009, many more systems have emerged, constantly seeking improvements.

In this work we analyze three matrix multiplication approaches, and implement them in C++ using the Microsoft SEAL library. As the most important current issue of homomorphic encryption is performance, in the practical part of this analysis we take space and time measurements in order to compare the three approaches.

Keywords: *cryptography, matrix multiplication, performance, homomorphism, homomorphic encryption*

1 *Εισαγωγή*

1.1 Εισαγωγή στην ασφάλεια υπολογιστών

Η ασφάλεια των πληροφοριών ορίζεται ως "η προστασία των πληροφοριών και των πληροφοριακών συστημάτων από μη εξουσιοδοτημένη πρόσβαση, χρήση, αποκάλυψη, διατάραξη, τροποποίηση ή καταστροφή" σύμφωνα με το νόμο [49].

Στην ουσία, σημαίνει ότι θέλουμε να προστατεύσουμε τα δεδομένα μας (όπου κι αν βρίσκονται αποθηκευμένα) και τα περιουσιακά στοιχεία των συστημάτων μας από εκείνους που θα προσπαθήσουν να τα καταχραστούν. Αυτό μπορεί να σημαίνει την προστασία τους από επιτιθέμενους που εισβάλλουν στα δίκτυά μας, virus/worms, φυσικές καταστροφές, δυσμενείς περιβαλλοντικές συνθήκες, διακοπές ρεύματος, κλοπή, βανδαλισμό ή άλλη ανεπιθύμητη κατάσταση. Όταν εξασφαλίζουμε ένα περιουσιακό στοιχείο, ένα σύστημα ή ένα περιβάλλον, πρέπει επίσης να εξετάζουμε πώς το επίπεδο ασφάλειας σχετίζεται με την αξία του αντικειμένου που εξασφαλίζεται. Μπορούμε, αν είμαστε πρόθυμοι να δεχτούμε τη μείωση της απόδοσης, να εφαρμόσουμε πολύ υψηλά επίπεδα ασφάλειας σε κάθε περιουσιακό στοιχείο για το οποίο είμαστε υπεύθυνοι. Όλη αυτή η ανακατανομή περιουσιακών στοιχείων για την εξασφάλιση ενός τομέα τους πρέπει επίσης να έχει νόημα. Σε κάθε περιβάλλον όπου σχεδιάζουμε να θέσουμε σε εφαρμογή αυξημένα επίπεδα ασφάλειας, πρέπει επίσης να λάβουμε υπόψη το κόστος αντικατάστασης των περιουσιακών μας στοιχείων, εάν τύχει να τα χάσουμε, και να βεβαιωθούμε ότι έχουμε καθιερώσει ένα λογικό επίπεδο προστασίας για την αξία τους. Το κόστος της ασφάλειας που θέτουμε σε εφαρμογή δεν πρέπει ποτέ να ξεπερνά την αξία του αγαθού που προστατεύει [49].

Η κρυπτογραφία είναι η επιστήμη και η μελέτη των τεχνικών για την ασφαλή επικοινωνία με την παρουσία αντίπαλης συμπεριφοράς [51]. Η κρυπτογραφία κάνει εκτεταμένη χρήση επιμέρους κλάδων μαθηματικών, όπως η θεωρία της πληροφορίας, η υπολογιστική πολυπλοκότητα, η στατιστική, η συνδυαστική, η αφηρημένη άλγεβρα, η θεωρία αριθμών και τα πεπερασμένα μαθηματικά. Είναι επίσης ένας κλάδος της μηχανικής, αλλά ασυνήθιστος, καθώς

ασχολείται με ευφυή και κακόβουλη πρόθεση από πιθανούς αντιπάλους. Υπάρχει επίσης ενεργή έρευνα που εξετάζει τη σχέση μεταξύ των κρυπτογραφικών προβλημάτων και της κβαντικής φυσικής. Η κρυπτογραφία χρησιμοποιείται ευρέως στο διαδίκτυο για την προστασία των δεδομένων των χρηστών και την αποτροπή υποκλοπών. Για να εξασφαλιστεί η μυστικότητα κατά τη μετάδοση, πολλά συστήματα χρησιμοποιούν κρυπτογραφία ιδιωτικού κλειδιού για την προστασία των μεταδιδόμενων πληροφοριών. Με τα συστήματα δημόσιου κλειδιού, μπορεί κανείς να διατηρήσει το απόρρητο χωρίς κύριο κλειδί ή μεγάλο αριθμό κλειδιών [52][53]. Τα λειτουργικά συστήματα χρησιμοποιούν την κρυπτογράφηση για να κρατούν μυστικούς τους κωδικούς πρόσβασης, να αποκρύπτουν τμήματα του συστήματος και να διασφαλίζουν ότι οι ενημερώσεις λογισμικού είναι πραγματικά από τον κατασκευαστή του συστήματος Αντί να αποθηκεύουν κωδικούς πρόσβασης σε απλό κείμενο, τα συστήματα υπολογιστών αποθηκεύουν τους κατακερματισμούς τους. Στη συνέχεια, όταν ένας χρήστης συνδέεται, το σύστημα περνάει τον συγκεκριμένο κωδικό πρόσβασης από μια συνάρτηση κρυπτογραφικού κατακερματισμού και τον συγκρίνει με την κατακερματισμένη τιμή στο αρχείο. Με αυτόν τον τρόπο, ούτε το σύστημα ούτε ένας επιτιθέμενος έχει σε οποιοδήποτε σημείο πρόσβαση στον κωδικό πρόσβασης σε απλό κείμενο [51].

1.2 Αντικείμενο διπλωματικής

Ένα από τα προβλήματα που υπάρχουν στα τωρινά συστήματα κρυπτογράφησης είναι ότι δεν προσφέρουν απόλυτη ασφάλεια σε περίπτωση που ευαίσθητα δεδομένα σταλθούν σε μια ενδιάμεση υπηρεσία. Πρέπει η υπηρεσία να είναι απόλυτα έμπιστη, διότι δεν μπορεί να γίνει επεξεργασία των δεδομένων ενώ εκείνα βρίσκονται σε κρυπτογραφημένη μορφή [4]. Η ομομορφική κρυπτογραφία (Homomorphic Encryption, HE) είναι ένα ειδικό είδος μηχανισμού κρυπτογράφησης για την αντιμετώπιση ζητημάτων κενών ασφαλείας στην ανάθεση υπολογισμών για ευαίσθητα δεδομένα σε ένα μη αξιόπιστο υπολογιστικό περιβάλλον [5]. Επιτρέπει σε παρόχους υπηρεσιών να εκτελούν συγκεκριμένους τύπους λειτουργιών στα κρυπτογραφημένα δεδομένα των χρηστών χωρίς την αποκρυπτογράφηση τους, ενώ διατηρείται το απόρρητο των κρυπτογραφημένων δεδομένων των χρηστών [4]. Το αποτέλεσμα της επεξεργασίας μετά την αποκρυπτογράφηση, ταυτίζεται με το αποτέλεσμα της ίδιας επεξεργασίας στα αρχικά δεδομένα [5].

Το αντικείμενο της εργασίας είναι η θεωρητική ανάλυση τριών προσεγγίσεων πολλαπλασιασμού πινάκων, καθώς και η υλοποίηση τους σε γλώσσα C++ με την χρήση της βιβλιοθήκης ομομορφικής κρυπτογράφησης Microsoft SEAL. Καθώς το πλέον πιο σημαντικό ελάττωμα της ομομορφικής κρυπτογράφησης είναι η απόδοση, δηλαδή η κατανάλωση πόρων σε χωρητικότητα και χρόνο, στο πρακτικό κομμάτι της εργασίας γίνονται μετρήσεις χώρου και χρόνου με σκοπό την σύγκριση των τριών προσεγγίσεων.

1.3 Secure Computation

Ο ασφαλής υπολογισμός είναι ένα υποπεδίο της κρυπτογραφίας με στόχο τη δημιουργία μεθόδων ώστε τα μέρη να υπολογίζουν από κοινού μια συνάρτηση πάνω στις εισόδους τους, διατηρώντας τις εισόδους αυτές ιδιωτικές [54]. Τα πρωτόκολλα ειδικού σκοπού για συγκεκριμένες εργασίες ξεκίνησαν στα τέλη της δεκαετίας του 1970 [55]. Αργότερα, ο ασφαλής υπολογισμός εισήχθη επίσημα ως ασφαλής υπολογισμός δύο μερών (2PC) το 1982 (για το λεγόμενο Πρόβλημα των Εκατομμυριούχων, ένα συγκεκριμένο πρόβλημα που είναι ένα Boolean κατηγόρημα), και γενικά (για κάθε εφικτό υπολογισμό) το 1986 από τον Andrew Yao [57]. Η περιοχή αναφέρεται επίσης ως ασφαλής αξιολόγηση συναρτήσεων (Secure Function Evaluation - SFE). Η περίπτωση των δύο μερών ακολουθήθηκε από μια γενίκευση στην περίπτωση των πολλών μερών από τους Goldreich, Micali και Wigderson [57]. Ο υπολογισμός βασίζεται σε μυστικό διαμοιρασμό όλων των εισόδων και σε αποδείξεις μηδενικής γνώσης για μια δυνητικά κακόβουλη περίπτωση, όπου η πλειοψηφία των ειλικρινών παικτών στην περίπτωση του κακόβουλου αντιπάλου διασφαλίζει ότι η κακή συμπεριφορά ανιχνεύεται και ο υπολογισμός συνεχίζεται με την εξάλειψη του ανέντιμου ατόμου ή την αποκάλυψη της εισόδου του. Η εργασία αυτή πρότεινε το πολύ βασικό γενικό σχήμα που θα ακολουθήσουν ουσιαστικά όλα τα μελλοντικά πρωτόκολλα πολλαπλών μερών για ασφαλείς υπολογισμούς.

Ένα πρωτόκολλο υπολογισμού πολλαπλών μερών πρέπει να είναι ασφαλές για να είναι αποτελεσματικό. Στη σύγχρονη κρυπτογραφία, η ασφάλεια ενός πρωτοκόλλου σχετίζεται με μια απόδειξη ασφάλειας. Η απόδειξη ασφάλειας είναι μια μαθηματική απόδειξη όπου η ασφάλεια ενός πρωτοκόλλου ανάγεται στην ασφάλεια των υποκείμενων πρωταρχικών του. Παρ' όλα αυτά, δεν είναι πάντοτε δυνατή η τυποποίηση της απόδειξης ασφάλειας κρυπτογραφικού πρωτοκόλλου με βάση τη γνώση των μερών και την ορθότητα του πρωτοκόλλου. Για τα πρωτόκολλα MPC, το περιβάλλον στο οποίο λειτουργεί το πρωτόκολλο σχετίζεται με το Παράδειγμα Πραγματικού Κόσμου/Ιδεατού Κόσμου. Τα μέρη δεν μπορούμε να πούμε ότι δεν μαθαίνουν τίποτα, αφού πρέπει να μάθουν την έξοδο της λειτουργίας και η έξοδος εξαρτάται από τις εισόδους. Επιπλέον, η ορθότητα της εξόδου δεν είναι εγγυημένη, αφού η ορθότητα της εξόδου εξαρτάται από τις εισόδους των μερών, και οι είσοδοι πρέπει να θεωρηθούν σωστές [58].

Υπάρχουν σημαντικές διαφορές μεταξύ των πρωτοκόλλων που προτείνονται για τον υπολογισμό δύο μερών (2PC) και τον υπολογισμό πολλών μερών (MPC).

Η περίπτωση των δύο μερών είναι ιδιαίτερα ενδιαφέρουσα, όχι μόνο από την άποψη των εφαρμογών, αλλά και επειδή στο πλαίσιο των δύο μερών μπορούν να εφαρμοστούν ειδικές τεχνικές που δεν εφαρμόζονται στην περίπτωση των πολλών μερών. Πράγματι, ο ασφαλής υπολογισμός πολλαπλών μερών (στην πραγματικότητα η περιορισμένη περίπτωση της ασφαλούς αξιολόγησης συναρτήσεων, όπου αξιολογείται μόνο μία συνάρτηση) παρουσιάστηκε για πρώτη φορά στο περιβάλλον δύο μερών. Η αρχική εργασία αναφέρεται συχνά ως προερχόμενη από μία από τις δύο εργασίες του Yao[20], αν και οι εργασίες αυτές δεν περιέχουν αυτό που σήμερα είναι γνωστό ως πρωτόκολλο αλλοιωμένου κυκλώματος του Yao.

Το βασικό πρωτόκολλο του Yao είναι ασφαλές έναντι ημι-εντιμότατων αντιπάλων και είναι εξαιρετικά αποδοτικό όσον αφορά τον αριθμό των γύρων, ο οποίος είναι σταθερός και ανεξάρτητος από τη συνάρτηση-στόχο που αξιολογείται. Η συνάρτηση θεωρείται ως ένα κύκλωμα Boole, με εισόδους σε δυαδικό σύστημα σταθερού μήκους. Ένα κύκλωμα Boole είναι μια συλλογή πυλών που συνδέονται με

τρεις διαφορετικούς τύπους καλωδίων: καλώδια εισόδου κυκλώματος, καλώδια εξόδου κυκλώματος και ενδιάμεσα καλώδια. Κάθε πύλη λαμβάνει δύο καλώδια εισόδου και έχει ένα μόνο καλώδιο εξόδου το οποίο μπορεί να είναι fan-out (δηλαδή να περάσει σε πολλαπλές πύλες στο επόμενο επίπεδο). Η απλή αξιολόγηση του κυκλώματος γίνεται με την αξιολόγηση κάθε πύλης με τη σειρά, υποθέτοντας ότι οι πύλες έχουν ταξινομηθεί τοπολογικά. Η πύλη αναπαρίσταται ως πίνακας αληθείας, έτσι ώστε για κάθε πιθανό ζεύγος bit (αυτά που προέρχονται από την πύλη των συρμάτων εισόδου) ο πίνακας να αναθέτει ένα μοναδικό bit εξόδου, το οποίο είναι η τιμή του καλωδίου εξόδου της πύλης. Τα αποτελέσματα της αξιολόγησης είναι τα bit που λαμβάνονται στα σύρματα εξόδου του κυκλώματος. Ο Yao εξήγησε πώς μπορεί να αλλοιωθεί ένα κύκλωμα (να κρυφτεί η δομή του) έτσι ώστε δύο μέρη, ο αποστολέας και ο παραλήπτης, να μπορούν να μάθουν την έξοδο του κυκλώματος και τίποτε άλλο. Σε υψηλό επίπεδο, ο αποστολέας προετοιμάζει το αλλοιωμένο κύκλωμα και το στέλνει στον παραλήπτη, ο οποίος αξιολογεί το κύκλωμα χωρίς να το αντιληφθεί, μαθαίνοντας τις κωδικοποιήσεις που αντιστοιχούν τόσο στη δική του όσο και στην έξοδο του αποστολέα. Στη συνέχεια, στέλνει απλώς πίσω τις κωδικοποιήσεις του αποστολέα, επιτρέποντας στον αποστολέα να υπολογίσει το δικό του μέρος της εξόδου. Ο αποστολέας στέλνει στον παραλήπτη την αντιστοίχιση από τις κωδικοποιήσεις εξόδου του παραλήπτη σε bits, επιτρέποντας στον παραλήπτη να λάβει την έξοδό του [58].

Τα περισσότερα πρωτόκολλα MPC, σε αντίθεση με τα πρωτόκολλα 2PC και ιδίως υπό την άνευ όρων ρύθμιση των ιδιωτικών καναλιών, χρησιμοποιούν την ανταλλαγή μυστικών. Στις μεθόδους που βασίζονται στην κοινή χρήση μυστικών, τα μέρη δεν παίζουν ειδικούς ρόλους (όπως στον Yao, του δημιουργού και του αξιολογητή). Αντ' αυτού, τα δεδομένα που σχετίζονται με κάθε καλώδιο μοιράζονται μεταξύ των μερών και στη συνέχεια χρησιμοποιείται ένα πρωτόκολλο για την αξιολόγηση κάθε πύλης. Η συνάρτηση ορίζεται τώρα ως ένα "κύκλωμα" πάνω σε ένα πεπερασμένο πεδίο, σε αντίθεση με τα δυαδικά κυκλώματα που χρησιμοποιήθηκαν για το Yao. Ένα τέτοιο κύκλωμα ονομάζεται στη βιβλιογραφία αριθμητικό κύκλωμα και αποτελείται από "πύλες" πρόσθεσης και πολλαπλασιασμού, όπου οι τιμές στις οποίες γίνεται η πράξη ορίζονται πάνω σε ένα πεπερασμένο πεδίο. Ο διαμοιρασμός μυστικών επιτρέπει τη διανομή ενός μυστικού σε έναν αριθμό μερών διανέμοντας μερίδια σε κάθε μέρος. Συνήθως χρησιμοποιούνται δύο τύποι συστημάτων διαμοιρασμού μυστικών: ο διαμοιρασμός μυστικών Shamir και ο προσθετικός διαμοιρασμός μυστικών. Και στις δύο περιπτώσεις τα μερίδια είναι τυχαία στοιχεία ενός πεπερασμένου πεδίου που αθροίζουν το μυστικό στο πεδίο. Διαισθητικά, επιτυγχάνεται ασφάλεια επειδή οποιοδήποτε μη κατάλληλο σύνολο μεριδίων μοιάζει τυχαία κατανεμημένο. Τα σχήματα μυστικού διαμοιρασμού μπορούν να ανεχθούν έναν αντίπαλο που ελέγχει έως και t μέρη από τα n συνολικά μέρη, όπου το t ποικίλλει ανάλογα με το σχήμα, ο αντίπαλος μπορεί να είναι παθητικός ή ενεργός και γίνονται διαφορετικές υποθέσεις σχετικά με την ισχύ του αντιπάλου. Πολλά συστήματα έχουν εφαρμόσει διάφορες μορφές MPC με συστήματα διαμοιρασμού μυστικών. Το πιο δημοφιλές είναι το SPDZ, το οποίο υλοποιεί το MPC με προσθετικά μυστικά μερίδια και είναι ασφαλές έναντι ενεργών αντιπάλων [59].

Το 2014 περιγράφηκε ένα "μοντέλο δικαιοσύνης σε ασφαλείς υπολογισμούς στο οποίο ένα αντίπαλο μέρος που διακόπτει τη λήψη εξόδου αναγκάζεται να πληρώσει μια αμοιβαία προκαθορισμένη χρηματική ποινή" για το δίκτυο Bitcoin ή για δίκαιη λαχειοφόρο αγορά [60].

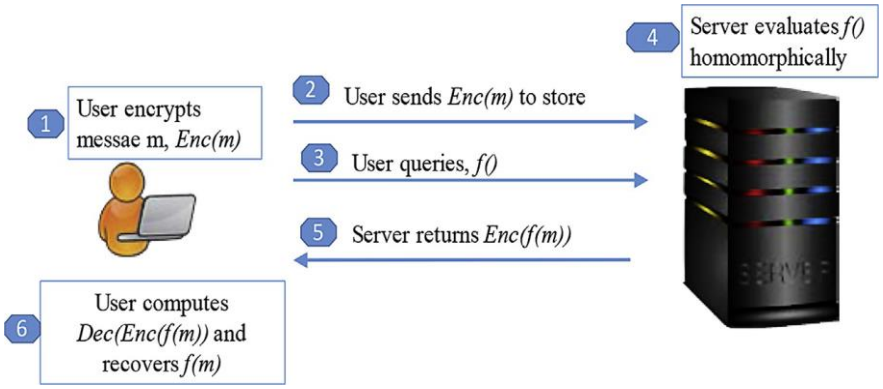
1.4 Δομή της διπλωματικής

Στην παρούσα εργασία θα δούμε μια μικρή αναφορά και ιστορική αναδρομή για την ασφάλεια και εισαγωγικές έννοιες στο secure computation. Στο δεύτερο κεφάλαιο εισάγουμε τον αναγνώστη στην ομομορφική κρυπτογραφία, στα είδη της, στα σχήματα της, τις βασικές ομομορφικές πράξεις και τις βιβλιοθήκες που την υλοποιούν. Στο τρίτο κεφάλαιο της εργασίας θα δούμε τον πολλαπλασιασμό πινάκων. Αρχικά θα αναλύσουμε τον ρόλο τους στα νευρωνικά δίκτυα και θα μεταβούμε στην συνέχεια στις προσεγγίσεις υλοποίησης του. Στο τέταρτο κεφάλαιο θα επεξηγήσουμε πλήρως την προσέγγιση μας για τις υλοποιήσεις για πολλαπλασιασμό πινάκων που εισάγουμε στον τρίτο κεφάλαιο. Τέλος, στο πέμπτο κεφάλαιο συγκρίνουμε τις υλοποιήσεις μας αναμεταξύ τους και αναγράφουμε τα συμπεράσματα μας που προέκυψαν από αυτές.

2 Ομομορφική κρυπτογραφία

2.1 Εισαγωγή και ιστορική αναδρομή

Τα συμβατικά συστήματα κρυπτογράφησης δεν προσφέρουν απόλυτη ασφάλεια σε περίπτωση που ευαίσθητα δεδομένα σταλθούν σε μια ενδιάμεση υπηρεσία όπως οι cloud διακομιστές. Καθώς δεν μπορεί να γίνει επεξεργασία των δεδομένων ενώ εκείνα βρίσκονται σε κρυπτογραφημένη μορφή, πρέπει η υπηρεσία να είναι απόλυτα έμπιστη [4].



Σχήμα 1: Βασικό σενάριο HE [4].

Η ομομορφική κρυπτογραφία (HE) είναι ένα ειδικό είδος μηχανισμού κρυπτογράφησης για την αντιμετώπιση ζητημάτων κενού ασφαλείας στην ανάθεση υπολογισμών για ευαίσθητα δεδομένα σε ένα μη αξιόπιστο υπολογιστικό περιβάλλον [5]. Επιτρέπει σε παρόχους υπηρεσιών να εκτελούν συγκεκριμένους τύπους λειτουργιών στα κρυπτογραφημένα δεδομένα των χρηστών χωρίς αποκρυπτογράφηση τους, ενώ διατηρείται το απόρρητο των κρυπτογραφημένων δεδομένων των χρηστών [4]. Το αποτέλεσμα της επεξεργασίας μετά την αποκρυπτογράφηση, ταυτίζεται με το αποτέλεσμα της ίδιας επεξεργασίας στα αρχικά δεδομένα [5].

Το 1978, οι Rivest, Adleman και Dertouzos περιέγραψαν για πρώτη φορά την δύναμη της ομομορφικής κρυπτογραφίας και τις πιθανές εφαρμογές της [1]. Εξηγούν πως η κρυπτογράφηση εως τότε ήταν μια διαδεδομένη μέθοδος για την διατήρηση της ιδιωτικότητας των δεδομένων, όμως είχε ένα σημαντικό μειονέκτημα, την αδυναμία επεξεργασίας των δεδομένων αφότου είχαν κρυπτογραφηθεί.

Για την εξάλειψη αυτού του περιορισμού, οι Rivest et al. [1] πρότειναν την δημιουργία μεθόδων με την βοήθεια των οποίων θα μπορούσε να γίνει εφικτή η επεξεργασία ενός κρυπτοκειμένου χωρίς να πρέπει πρώτα να αποκρυπτογραφηθεί.

Εάν υποθέσουμε ότι ένα αλγεβρικό σύστημα αποτελείται από ένα σύνολο S, κάποιες συναρτήσεις f1 ... fk, κάποια κατηγορήματα p1 … pl και σταθερές s1 … sm και ότι έχουμε τα παρακάτω δυο αλγεβρικά συστήματα:

U = <S; f1 ... fk ; p1 … pl ; s1 … sm >

C = <S'; f'1 ... f'k ; p'1 … p'l ; s'1 … s'm >

Όπου το σύστημα U ανήκει στον χρήστη ο οποίος είναι ο ιδιοκτήτης των δεδομένων και τα κρυπτογραφεί πριν τα αποθηκεύσει σε ένα υπολογιστικό σύστημα. Το σύστημα C αντιστοιχεί στο αλγεβρικό σύστημα του υπολογιστή ο οποίος έχει αποθηκευμένα τα κρυπτογραφημένα δεδομένα του χρήστη.

Για να είναι εφικτές οι συναρτήσεις f'1 ... f'k απευθείας πάνω σε κρυπτογραφημένα δεδομένα χωρίς ο υπολογιστής να έχει γνώση της διαδικασίας αποκρυπτογράφησης, έστω φ, και φυσικά το αποκρυπτογραφημένο αποτέλεσμα να είναι σωστό, θα πρέπει η φ να αποτελεί ομομορφισμό από το αλγεβρικό σύστημα C στο αλγεβρικό σύστημα U [1]. Έτσι οι Rivest et al. ορίζουν την έννοια του privacy homomorphism και προτείνουν τέσσερα διαφορετικά παραδείγματα τέτοιων ομομορφισμών.

Αργότερα οι Brickell και Yacobi [2] αποδεικνύουν ότι οι δύο από τους τέσσερις ομομορφισμούς είναι ευάλωτοι σε συγκεκριμένο ciphertext only attack και οι άλλοι δυο μπορούν να σπάσουν χρησιμοποιώντας ένα γνωστό plaintext attack.

Το πρώτο πλήρως ομομορφικό σχήμα κρυπτογράφησης ορίστηκε από τον Gentry [3] το 2009 δίνοντας λύση στο πρόβλημα των Rivest et al. [1]. Η λύση που δόθηκε καθιστούσε εφικτή την εκτέλεση αυθαίρετων συναρτήσεων πάνω σε κρυπτογραφημένα δεδομένα χωρίς να υπάρχει γνώση του κλειδιού αποκρυπτογράφησης. Πιο συγκεκριμένα, έχοντας μόνο τα κρυπτοκείμενα E(m1),...,E(mt) των μηνυμάτων m1,...,mt, γίνεται να παραχθεί ένα κρυπτοκείμενο το οποίο κρυπτογραφεί το f(m1,...,mt) για οποιαδήποτε υπολογίσιμη συνάρτηση f.

Αυτή η λύση άνοιξε νέους ορίζοντες στην επιστήμη της κρυπτογραφίας και δημιουργείται μια πληθώρα πιθανών εφαρμογών σε διάφορους τομείς όπως η υγειονομική περίθαλψη, οι ιατρικές εφαρμογές, ο χρηματοοικονομικός τομέας, οι εγκληματολογικές εφαρμογές, οι διαφημίσεις κοινωνικής δικτύωσης, και τα έξυπνα οχήματα, όπου μπορεί να διατηρηθεί το απόρρητο των χρηστών [4].

Σε ένα πιθανό σενάριο κάποιος χρήστης θα μπορούσε να στείλει ένα κρυπτογραφημένο ερώτημα σε μια μηχανή αναζήτησης, και εκείνη να επιστρέψει το αποτέλεσμα σε επίσης κρυπτογραφημένη μορφή χωρίς ποτέ να δει ποιο πραγματικά ήταν το ερώτημα του χρήστη [3]. Σε άλλη περίπτωση, μπορεί ο χρήστης να έχει αποθηκεύσει κρυπτογραφημένα δεδομένα σε κάποιον

server και να θέλει να διαλέξει από αυτά μόνο όσα πληρούν κάποια συνθήκη. Ο server θα μπορούσε να είναι σε θέση να υπολογίσει ποια δεδομένα είναι αυτά και να τα επιστρέψει χωρίς ποτέ να τα αποκρυπτογραφήσει. Με λίγα λόγια ο χρήστης θα μπορούσε να έχει τον πλήρη έλεγχο της ιδιωτικότητας των δεδομένων του καθώς θα είναι ο μόνος που μπορεί να αποκρυπτογραφήσει το τελικό αποτέλεσμα, χρησιμοποιώντας το μυστικό κλειδί του.

2.2 Είδη ομομορφικής κρυπτογραφίας

Έστω τα μηνύματα σε μορφή plaintext m_1, m_2 και αντίστοιχα τα ciphertexts $E(m_1)$, $E(m_2)$. Ορίζουμε ως ομομορφισμό πρόσθεσης (addition homomorphism) την παρακάτω ισότητα:

$$E(m_1) + E(m_2) = E(m_1 + m_2)$$

Ορίζουμε ως ομομορφισμό πολλαπλασιασμού (multiplication homomorphism) την παρακάτω ισότητα:

$$E(m_1) \times E(m_2) = E(m_1 \times m_2)$$

Μπορούμε να φανταστούμε τις ομομορφικές πράξεις ως την διαδικασία αποτίμησης αριθμητικών ή boolean κυκλωμάτων [3]. Υπάρχουν τρεις κατηγορίες Η.Ε., ανάλογα με τα είδη των κυκλωμάτων που υποστηρίζονται καθώς και το βάθος [4]. Επιγραμματικά τα είδη που υπάρχουν είναι:

- Partially Homomorphic Encryption (PHE)
- Somewhat Homomorphic Encryption (SWHE)
- Fully Homomorphic Encryption (FHE)
- Leveled Fully Homomorphic Encryption (LFHE)

2.2.1 Partially Homomorphic Encryption (PHE)

Στην κατηγορία PHE μόνο ένα είδος μαθηματικών πράξεων επιτρέπεται να εκτελεστεί, δηλαδή είτε αθροιστικά ή πολλαπλασιαστικά κυκλώματα αυθαίρετου βάθους [4]. Εδώ ανήκουν κρυπτογραφικά συστήματα που δημιουργήθηκαν πριν το πρώτο FHE σύστημα του Gentry το 2009. Τα παρακάτω παραδείγματα συστημάτων θεωρείται πως έχουν ομομορφικές ιδιότητες αλλά όχι πλήρως:

- RSA: Ο RSA επιτρέπει ομομορφισμό πολλαπλασιασμού [7]. Πολλαπλασιάζοντας δυο ή περισσότερα ciphertext τα οποία δημιουργήθηκαν με RSA, και στη συνέχεια αποκρυπτογραφώντας το γινόμενο, παίρνουμε το ίδιο αποτέλεσμα με τον πολλαπλασιασμό των αντίστοιχων plaintext.
- ElGamal: Ο ElGamal επίσης επιτρέπει ομομορφισμό πολλαπλασιασμού [7]. Πολλαπλασιάζοντας κάθε τμήμα από ένα ciphertext με το αντίστοιχο τμήμα ενός

δεύτερου, και στη συνέχεια αποκρυπτογραφώντας το γινόμενο, παίρνουμε το ίδιο αποτέλεσμα με τον πολλαπλασιασμό των αντίστοιχων plaintext.

- Paillier: Ο Paillier επιτρέπει ομομορφισμό πρόσθεσης [7]. Πολλαπλασιάζοντας κάθε τμήμα από ένα ciphertext με το αντίστοιχο τμήμα ενός δεύτερου, και στη συνέχεια αποκρυπτογραφώντας το γινόμενο, παίρνουμε το ίδιο αποτέλεσμα με την πρόσθεση των αντίστοιχων plaintext.

2.2.2 Somewhat Homomorphic Encryption (SWHE)

Υπάρχει δυνατότητα εκτέλεσης πολλαπλασιαστικών και αθροιστικών κυκλωμάτων, όπου το βάθος των κυκλωμάτων μπορεί να αυξηθεί όμως με συνέπεια την αύξηση του μεγέθους του κρυπτοκειμένου [12]. Τα σχήματα που ανήκουν σε αυτό το είδος είναι ικανά να αξιολογήσουν πολυώνυμα πολλών μεταβλητών χαμηλού βαθμού ομομορφικά. Συνήθως αποτελούν το αρχικό στάδιο ώστε να δημιουργηθεί ένα FHE σχήμα [6].

2.2.3 Fully Homomorphic Encryption (FHE)

Στο FHE υπάρχει δυνατότητα εκτέλεσης πολλαπλασιαστικών, αθροιστικών και boolean κυκλωμάτων χωρίς περιορισμό στο βάθος [4]. Ο πρώτος αλγόριθμος FHE δόθηκε από τον Gentry [3]. Σε υψηλό επίπεδο, η ουσία της πλήρως ομομορφικής κρυπτογράφησης είναι απλή: Με δεδομένα κρυπτογραφημένα κείμενα που κρυπτογραφούν τα μηνύματα $\pi_1,...,\pi_t$, η πλήρως ομομορφική κρυπτογράφηση θα πρέπει να επιτρέπει σε οποιονδήποτε (όχι μόνο τον κάτοχο του μυστικού κλειδιού) να παράγει το αποτέλεσμα ενός κρυπτογραφημένου κειμένου που κρυπτογραφεί το $f(\pi_1,...,\pi_t)$ για οποιαδήποτε επιθυμητή συνάρτηση f, εφόσον αυτή η συνάρτηση μπορεί να υπολογιστεί αποτελεσματικά. Καμία πληροφορία για το $\pi_1,...,\pi_t$ ή $f(\pi_1,...,\pi_t)$ ή οποιεσδήποτε ενδιάμεσες τιμές απλού κειμένου, δεν πρέπει να διαρρεύσει. Τα δεδομένα εισόδου, εξόδου και οι ενδιάμεσες τιμές παραμένουν πάντα κρυπτογραφημένες.

Το πρόβλημα στο οποίο δόθηκε λύση διατυπώθηκε ως εξής. Η κατασκευή ενός σχήματος E με μια διαδικασία Evaluate_E τέτοια ώστε:

Για κάθε έγκυρο δημόσιο κλειδί κρυπτογράφησης pk, οποιοδήποτε κύκλωμα C και κάθε κρυπτοκείμενο ψ_i, η $\text{Encrypt}_E(pk, \pi_i)$ να έχει ως έξοδο $\psi = \text{Evaluate}_E(pk, C, \psi_1,...,\psi_t)$ την έγκυρη κρυπτογράφηση του $C(\pi_1,...\pi_k)$ χρησιμοποιώντας το κλειδί pk.

Η λύση του [3] υποστηρίζει πράξεις πρόσθεσης και πολλαπλασιασμού σε κρυπτοκείμενα, από τα οποία είναι δυνατή η κατασκευή κυκλωμάτων για την εκτέλεση αυθαίρετων υπολογισμών. Η κατασκευή του σχήματος ξεκινά από ένα SWHE, το οποίο περιορίζεται στην αξιολόγηση πολυωνύμων χαμηλού βαθμού σε κρυπτογραφημένα δεδομένα. Κάθε κρυπτοκείμενο διαθέτει μια τιμή θορύβου, η οποία μεγαλώνει καθώς το κρυπτοκείμενο επεξεργάζεται ειδικά στην περίπτωση του πολλαπλασιασμού. Εάν η τιμή του θορύβου ξεπεράσει ένα συγκεκριμένο όριο, το επεξεργασμένο κρυπτοκείμενο δεν μπορεί πλέον να αποκρυπτογραφηθεί.

Στη συνέχεια, ο Gentry δείχνει τον τρόπο ώστε αυτό το σχήμα να τροποποιηθεί για να γίνει bootstrappable, δηλαδή, ικανό να αξιολογήσει το δικό του κύκλωμα αποκρυπτογράφησης και στη συνέχεια, να αξιολογήσει τουλάχιστον μία ακόμη πράξη. Τέλος, δείχνει ότι οποιοδήποτε bootstrappable ομομορφικό σχήμα κρυπτογράφησης μπορεί να μετατραπεί σε FHE μέσω μιας αναδρομικής αυτο-ενσωμάτωσης. Για το «θορυβώδες» σχήμα του Gentry, η διαδικασία bootstrapping ουσιαστικά «ανανεώνει» το κρυπτοκείμενο, εφαρμόζοντας σε αυτό τη διαδικασία αποκρυπτογράφησης ομομορφικά, λαμβάνοντας έτσι ένα νέο κρυπτοκείμενο που κρυπτογραφεί την ίδια τιμή όπως πριν, αλλά με χαμηλότερη τιμή θορύβου. Ανανεώνοντας περιοδικά το κρυπτοκείμενο κάθε φορά που ο θόρυβος μεγαλώνει πολύ, είναι δυνατή η εκτέλεση ενός κυκλώματος C το οποίο αποτελείται από έναν συνδυασμό πολλαπλασιασμών και προσθέσεων, αυθαίρετου βάθους και χωρίς να αυξηθεί σημαντικά η τιμή του θορύβου.

Η βασική ιδέα που προτείνεται στο έργο του Gentry είναι το bootstrapping, δηλαδή η διαδικασία της ανανέωσης ενός κρυπτογραφημένου κειμένου με σκοπό την διατήρηση ενός χαμηλότερου επιπέδου θορύβου. Έτσι επιτρέπεται η παραγωγή ενός νέου κρυπτογραφημένου κειμένου για να γίνει εφικτό να αξιολογηθούν περισσότερες ακόμη ομομορφικές πράξεις. Ωστόσο, το bootstrapping απαιτούσε προσθήκη πολλών κρυπτογραφημένων κειμένων στο δημόσιο κλειδί και κρυπτογράφηση των μεμονωμένων bit (ή συντελεστών) του μυστικού κλειδιού. Αυτό οδήγησε σε πολύ μεγάλα δημόσια κλειδιά και το απλό κείμενο έπρεπε να κρυπτογραφηθεί ανά bit, αυξάνοντας την χωρητικότητα του κρυπτογραφημένου κειμένου σε κάθε βήμα, καθιστώντας το υπερβολικά ακριβό όσον αφορά τους υπολογισμούς [41].

Ο Gentry στήριξε την ασφάλεια του σχήματός του στην δυσκολία λύσης δύο μαθηματικών προβλημάτων: ορισμένα προβλήματα χειρότερης περίπτωσης σε ideal lattices και το πρόβλημα αθροίσματος αραιού υποσυνόλου. Αργότερα δημιουργήθηκαν και άλλα πλήρως ομομορφικά σχήματα βασιζόμενα στον μηχανισμό του Gentry σε μια επιτυχημένη προσπάθεια βελτίωσης κυρίως της απόδοσης του αλγορίθμου που αποτελούσε και το πιο σημαντικό εμπόδιο στην χρήση του σε πραγματικά σενάρια [8], [9], [10], [11] .

2.2.4 Leveled Fully Homomorphic Encryption (LFHE)

Επιτρέπεται η εκτέλεση ενός συνδυασμού πολλαπλασιαστικών και αθροιστικών κυκλωμάτων με συγκεκριμένο και προκαθορισμένο μέγιστο βάθος χωρίς να επηρεαστεί το μέγεθος του κρυπτοκειμένου [6], [12]. Ένα LFHE μπορεί να αποτελεί εξελιγμένη μορφή ενός SWHE [3]. Ως παράδειγμα ενός τέτοιου συστήματος παρουσιάζουμε την δουλειά των Brakerski, C. Gentry και V. Vaikuntanathan [6], όπου δείχνουν μια νέα προσέγγιση FHE που βελτιώνει την απόδοση συγκριτικά με την πρώτη προσπάθεια του Gentry [3], βασίζοντας την ασφάλεια σε πιο αδύναμες υποθέσεις. Η κεντρική ιδέα είναι ένας νέος τρόπος κατασκευής leveled πλήρως ομομορφικών σχημάτων κρυπτογράφησης (με ικανότητα αξιολόγησης κυκλωμάτων αυθαίρετου πολυωνυμικού μεγέθους), χωρίς την τεχνική bootstrapping του Gentry.

Η απόδοση της πλήρως ομομορφικής κρυπτογράφησης είναι το μεγάλο ερώτημα που ακολούθησε την εφεύρεσή του. Στο [6] δίνεται βάρος στο γενικό κόστος υπολογισμού ανά πύλη

του FHE σχήματος, που ορίζεται ως η αναλογία μεταξύ του χρόνου που χρειάζεται για να εκτελεστεί ομομορφικά ένα κύκλωμα σε κρυπτογραφημένες εισόδους και του χρόνου που χρειάζεται για να υπολογιστεί σε εισόδους απλού (μη-κρυπτογραφημένου) κειμένου. Σύμφωνα με το [6], τα σχήματα FHE που ακολουθούν το προσχέδιο του Gentry [3] έχουν αρκετά κακή απόδοση: η επιβάρυνση του υπολογισμού ανά πύλη είναι p(λ), ένα μεγάλο πολυώνυμο στην παράμετρο ασφαλείας. Αυτό το μειονέκτημα στην απόδοση φαίνεται πως είναι εγγενές για συστήματα που ακολουθούν αυτό το σχέδιο. Έτσι, προσπαθώντας να βελτιώσουν την απόδοση, οι Brakerski et al. κατασκεύασαν δύο νέες επιλογές οι οποίες φαίνεται πως έχουν καλύτερη απόδοση στις ανά πύλη πράξεις, που μέχρι τότε ήταν στην καλύτερη περίπτωση $\Omega(\lambda^4)$.

- Υποθέτοντας ένα δακτύλιο LWE για έναν παράγοντα προσέγγισης εκθετικό σε L, έχουμε ένα leveled FHE σχήμα που μπορεί να αξιολογήσει αριθμητικά κυκλώματα επιπέδου L χωρίς χρήση bootstrapping. Το σχήμα έχει $O(\lambda \cdot L^3)$ υπολογισμό ανά πύλη (δηλαδή, σχεδόν γραμμικό στην παράμετρο ασφαλείας) [6].
- Εναλλακτικά, υποθέτοντας ότι ο δακτύλιος LWE είναι δύσκολος για σχεδόν πολυωνυμικούς παράγοντες, έχουμε ένα leveled FHE σχήμα που χρησιμοποιεί το bootstrapping ως βελτιστοποίηση, όπου ο υπολογισμός ανά πύλη (η οποία περιλαμβάνει τη διαδικασία bootstrapping) είναι $O(\lambda^2)$, ανεξάρτητο του L [6].

2.3 Σχήματα

Υπάρχουν 3 κατηγορίες σχημάτων HE και η επιλογή ανάμεσα σε αυτές γίνεται ανάλογα με την περίπτωση. Κάθε κατηγορία σχημάτων είναι αποτελεσματική για μια συγκεκριμένη εφαρμογή. Ετσι, όταν συγκρίνει κανείς την αποτελεσματικότητα διαφορετικών ομομορφικών σχημάτων, πρέπει να λάβει υπόψη τη δεδομένη περίπτωση χρήσης [14].

Bitwise Encryption

Την πρώτη κατηγορία αποτελούν σχήματα που κρυπτογραφούν την εισαγωγή τους κατά bit. Αυτό σημαίνει ότι κάθε bit της εισόδου κρυπτογραφείται σε διαφορετικό κρυπτοκείμενο. Αυτά τα συστήματα πιστεύεται ότι είναι τα πιο αποτελεσματικά σε σχέση με τον συνολικό χρόνο λειτουργίας [14]. Οι πράξεις εκτελούνται σε κάθε bit ξεχωριστά. Παραδείγματα τέτοιων σχημάτων περιλαμβάνουν το FHEW και TFHE [15], [16].

Packing

Η επόμενη κατηγορία αντιστοιχεί σε σχήματα κρυπτογράφησης που επιτρέπουν την εισαγωγή πολλαπλών τιμών δεδομένων σε ένα μόνο κρυπτοκείμενο (packing) και την εκτέλεση υπολογισμών σε αυτές τις τιμές με μία εντολή σε πολλά δεδομένα (SIMD) [14]. Συγκεκριμένα, οι

κρυπτογραφημένες τιμές αποθηκεύονται σε διαφορετικές υποδοχές, έτσι ώστε οι πράξεις που εκτελούνται σε ένα μόνο κρυπτογραφημένο κείμενο να εκτελούνται αυτόματα σε κάθε υποδοχή ανεξάρτητα. Αν και οι ομομορφικές πράξεις σε αυτά τα σχήματα έχουν μικρότερη ακρίβεια σε σχέση με τα σχήματα κρυπτογράφησης bitwise, ο χρόνος λειτουργίας τους ανά SIMD μπορεί να είναι καλύτερος [14]. Στην κατηγορία περιλαμβάνονται τα BGV [6] και BFV [17], [10].

CKKS

Το σχήμα CKKS [18], το οποίο επιτρέπει την εκτέλεση υπολογισμών σε δυαδικούς αριθμούς, αποτελεί την τρίτη κατηγορία. Μοιάζει με τη δεύτερη κατηγορία με την έννοια ότι υποστηρίζει packing. Το σχήμα CKKS δεν έχει τους αλγεβρικούς περιορισμούς που μειώνουν την ικανότητα packing των BGV και BFV. Ως εκ τούτου, είναι συνήθως δυνατό να συσκευαστούν περισσότερα στοιχεία σε ένα κρυπτοκείμενο CKKS, με αποτέλεσμα καλύτερο κόστος ανά SIMD. Σε αντίθεση με τα προηγούμενα σχήματα, το CKKS δεν περιορίζεται σε ακέραιους αριθμούς αλλά κωδικοποιεί πραγματικούς αριθμούς. Ωστόσο, οι ομομορφικοί υπολογισμοί γίνονται κατά προσέγγιση, πράγμα που σημαίνει ότι τα αποκρυπτογραφημένα αποτελέσματα ισχύουν μόνο μέχρι μια ορισμένη ακρίβεια [14].

Είναι κοινώς αποδεκτό ότι τα σχήματα της πρώτης κατηγορίας είναι τα περισσότερο αποτελεσματικά για γενικές εφαρμογές. Δεδομένου ότι λειτουργούν σε επίπεδο bit, μπορούν να υπολογίσουν κάθε λογική πύλη πολύ αποτελεσματικά. Ο συνολικός χρόνος λειτουργίας είναι σε αυτή την περίπτωση το άθροισμα των χρόνων που απαιτούνται για την αξιολόγηση κάθε πύλης του κυκλώματος. Καθώς το μέγεθος του κυκλώματος μεγαλώνει, είναι δυνατή η βελτιστοποίηση μόνο ορισμένων τμημάτων του κυκλώματος προσδιορίζοντας ορισμένα μοτίβα [14]. Ένα άλλο πλεονέκτημα αυτών των σχημάτων είναι ότι έχουν πολύ γρήγορους αλγόριθμους bootstrapping που ανανεώνει τα κρυπτογραφημένα κείμενα για περαιτέρω υπολογισμό. Αυτό είναι πολύ βολικό στην πράξη καθώς μπορεί κανείς να ορίσει ένα τυπικό σύνολο παραμέτρων κρυπτογράφησης χωρίς να ξέρει τι συνάρτηση πρέπει να υπολογιστεί.

Τα σχήματα της δεύτερης κατηγορίας είναι πολύ αποτελεσματικά στην αξιολόγηση πολυωνυμικών συναρτήσεων. Ωστόσο, αυτά τα σχήματα γίνονται πολύ λιγότερο αποτελεσματικά όταν εξετάζονται άλλα είδη υπολογισμών, π.χ. πράξεις σύγκρισης, συναρτήσεις βημάτων [14]. Επιπλέον, το bootstrapping στους αλγορίθμους αυτών των σχημάτων έχει μεγάλο κόστος και συνήθως αποφεύγεται στην πράξη.

Το CKKS, όπως και τα σχήματα δεύτερης κατηγορίας, είναι πολύ αποτελεσματικό όταν λειτουργεί στα αριθμητικά κυκλώματα. Ωστόσο, σε αντίθεση με άλλα σχήματα που λειτουργούν με αριθμητική υπολοίπων, επιτρέπει την εκτέλεση υπολογισμών σε μιγαδικούς (και επομένως πραγματικούς) αριθμούς. Αν και αυτό είναι ένα σημαντικό πλεονέκτημα για πολλές περιπτώσεις χρήσης, το CKKS δεν διαθέτει εργαλεία απλοποίησης για την αξιολόγηση ορισμένων συναρτήσεων λόγω φαινομένων θεωρίας αριθμών σε σύγκριση με τη δεύτερη κατηγορία [14].Ο αλγόριθμος bootstrapping του CKKS είναι θεμελιωδώς διαφορετικός από τα σχήματα των άλλων

κατηγοριών καθώς ανανεώνει τα κρυπτογραφημένα κείμενα μόνο εν μέρει και εισάγει πρόσθετη απώλεια ακρίβειας στην έξοδο. Επομένως, το bootstrapping στο CKKS συνήθως αποφεύγεται στην πράξη.

Αν και το FHE προσφέρει τώρα μια σχετικά αποτελεσματική εναλλακτική λύση για ασφαλείς υπολογισμούς, ορισμένες συναρτήσεις παραμένουν δύσκολο να αξιολογηθούν αποτελεσματικά ανεξάρτητα από το υπό εξέταση σχήμα. Οι συναρτήσεις βήματος, οι οποίες απαιτούνται σε πολλές πρακτικές εφαρμογές, αποτελούν ένα καλό παράδειγμα τέτοιων συναρτήσεων λόγω της ασυνέχειας τους. Η δυσκολία αξιολόγησης των ασυνεχών συναρτήσεων προέρχεται από την προϋπόθεση να αξιολογηθεί μια αρκετά βασική και σχετικά απλή συνάρτηση: η συνάρτηση σύγκρισης. Αν και η σύγκριση είναι μια στοιχειώδης λειτουργία που απαιτείται σε πολλές εφαρμογές, παραμένει δύσκολο να αξιολογηθεί ομομορφικά [14].
Μέχρι τώρα, τα σχήματα της πρώτης κατηγορίας φαίνονται πολύ πιο κατάλληλα για τέτοιες μη αριθμητικές εργασίες, αλλά είναι αναποτελεσματικές για την αξιολόγηση αριθμητικών λειτουργιών. Ως εκ τούτου, θα πρέπει να καταφύγουμε σε αλγόριθμους μετατροπής για να αξιοποιήσουμε τις ιδιότητες διαφορετικών σχημάτων [14].

2.4 Βασικές ομομορφικές πράξεις

Η ομομορφική κρυπτογράφηση (HE) είναι ένα σύστημα που επιτρέπει αυθαίρετους υπολογισμούς σε κρυπτογραφημένα δεδομένα. Βασίζεται στην ιδιότητα ότι το αποκρυπτογραφημένο αποτέλεσμα των πράξεων μεταξύ κρυπτογραφημένων κειμένων είναι ίσο με το αποτέλεσμα των πράξεων μεταξύ απλών κειμένων, επομένως ο διακομιστής μπορεί να λειτουργήσει στα δεδομένα του πελάτη χωρίς να τα αποκρυπτογραφήσει. Ο στόχος των λειτουργιών HE είναι να κατασκευαστεί μια συνάρτηση HE που να δίνει το κρυπτογραφημένο αποτέλεσμα που ταιριάζει με τη λειτουργία απλού κειμένου. Για παράδειγμα, το αποτέλεσμα της πρόσθεσης απλού κειμένου, a+b πρέπει να ταιριάζει με το αποτέλεσμα της πρόσθεσης HE, ct.a+HE ct.b όπου +HE αντιπροσωπεύει την πρόσθεση HE. Αυτές οι βασικές πράξεις HE χτίζονται από το συνδυασμό των πράξεων πύλης HE (λέγοντας πύλη εννοούμε τις λογικές πύλες των κυκλωμάτων) [41].

Ο Craig Gentry, χρησιμοποιώντας κρυπτογραφία βασισμένη σε πλέγμα, περιέγραψε την πρώτη εύλογη κατασκευή για ένα πλήρως ομομορφικό σχήμα κρυπτογράφησης [3]. Το σχήμα του Gentry υποστηρίζει τόσο πράξεις πρόσθεσης όσο και πολλαπλασιασμού σε κρυπτογραφημένα κείμενα, από τα οποία είναι δυνατή η κατασκευή κυκλωμάτων για την εκτέλεση αυθαίρετων υπολογισμών. Η κατασκευή ξεκινάει από ένα κάπως ομομορφικό σχήμα κρυπτογράφησης (SWHE), το οποίο περιορίζεται στην αξιολόγηση πολυωνύμων χαμηλού βαθμού πάνω σε κρυπτογραφημένα δεδομένα - περιορίζεται επειδή κάθε κρυπτογράφημα είναι θορυβώδες υπό κάποια έννοια, και αυτός ο θόρυβος αυξάνεται καθώς προσθέτουμε και πολλαπλασιάζουμε κρυπτογραφήματα, μέχρι που τελικά ο θόρυβος καθιστά το προκύπτον κρυπτογράφημα μη αποκρυπτογραφήσιμο. Ένας χρήστης αποθηκεύει κρυπτογραφημένα αρχεία σε έναν

απομακρυσμένο διακομιστή αρχείων και μπορεί αργότερα να ζητήσει από τον διακομιστή να ανακτήσει μόνο τα αρχεία που (όταν αποκρυπτογραφηθούν) ικανοποιούν κάποιο boolean περιορισμό, παρόλο που ο διακομιστής δεν μπορεί να αποκρυπτογραφήσει τα αρχεία από μόνος του.

Με πλήρως ομομορφική κρυπτογράφηση, μπορεί κανείς να κατασκευάσει μη-διαδραστικές αποδείξεις μηδενικής γνώσης (non-interactive zero knowledge proofs (NIZK)) μικρού μεγέθους. Για παράδειγμα, ας υποθέσουμε ότι κάποιος θέλει να αποδείξει ότι $\pi_1, \ldots, \pi_t$ είναι μια ικανοποιητική ανάθεση ενός boolean κυκλώματος C. Δημιουργώντας ένα δημόσιο κλειδί pk για το πλήρως ομομορφικό σχήμα κρυπτογράφησης, τα κρυπτογραφήματα εισόδου $\psi_i \leftarrow \text{Encrypt}(pk, \pi_i)$ και το κρυπτογράφημα εξόδου $\psi* \leftarrow \text{Evaluate}(pk, C, \psi_1, \ldots, \psi_t)$. Η NIZK που ικανοποιεί η ανάθεσή της αποτελείται από αποδείξεις NIZK, σύμφωνα με οποιοδήποτε σχήμα NIZK, ότι τα pk, $\{\psi_i\}$ και $\psi*$ είναι καλά σχηματισμένα, όπου καλά σχηματισμένα για τα κρυπτογραφήματα σημαίνει ότι κάθε ψi είναι έγκυρη κρυπτογράφηση του "0" ή του "1" και το $\psi*$ είναι έγκυρη κρυπτογράφηση του "1". Ο επαληθευτής ελέγχει τα NIZKs για καλή διαμόρφωση και επιβεβαιώνει ότι $\psi* = \text{Evaluate}(pk, C, \psi_1, \ldots, \psi_t)$. Διαισθητικά, η απόδειξη NIZK λειτουργεί επειδή, εάν ο επαληθευτής πιστεύει ότι το pk και τα κρυπτογραφήματα εισόδου είναι καλά διαμορφωμένα, τότε η ορθότητα του σχήματος κρυπτογράφησης συνεπάγεται ότι το κρυπτογραφημένο κείμενο εξόδου μπορεί να κρυπτογραφήσει '1'. μόνο αν $C(\pi_1, \ldots, \pi_t) = 1$. Το μέγεθος αυτής της απόδειξης NIZK είναι ανάλογο του αριθμού των εισόδων του κυκλώματος, αλλά κατά τα άλλα είναι ανεξάρτητο από το μέγεθος του κυκλώματος [3].

Μπορεί κανείς να κατανοήσει το σχήμα αρκετά καλά μόνο με όρους δακτυλίων και ιδανικών (όχι πλεγμάτων). Οι δακτύλιοι και τα ιδανικά είναι απλά αλγεβρικά αντικείμενα. Παραδείγματα δακτυλίων είναι ο Z (οι ακέραιοι αριθμοί) και ο πολυωνυμικός δακτύλιος Z[x]/(f(x)), που αποτελείται από τα υπόλοιπα των ακέραιων πολυωνύμων modulo ένα μονοσήμαντο πολυώνυμο f(x). Οι δακτύλιοι είναι κλειστοί υπό την πρόσθεση "+", πολλαπλασιασμό '×' και την προσθετική αντιστροφή και έχουν προσθετική ταυτότητα '0' και πολλαπλασιαστικό ταυτότητα '1' [3].

Σε πρώτη προσέγγιση, ένα πλήρως ομοιομορφικό σχήμα κρυπτογράφησης που βασίζεται σε δακτυλίους και ιδεώδη θα μπορούσε να λειτουργήσει ως εξής. Το δημόσιο κλειδί pk περιέχει ένα ιδεώδες I και ένα χώρο απλού κειμένου P, όπου το τελευταίο αποτελείται βασικά από ένα σύνολο "διακεκριμένων αντιπροσώπων" των συνόλων του I - το μυστικό κλειδί sk αποτελείται από κάποια "μυστική γνώση" σχετικά με το I. Για να κρυπτογραφήσει $\pi \in P$, ο κρυπτογράφος στέλνει $\psi \; R \leftarrow \pi + I$, ένα "τυχαίο" μέλος του συνόλου $\pi + I$. Ο αποκρυπτογράφος χρησιμοποιεί τη μυστική του γνώση για να ανακτήσει τον "διακεκριμένο εκπρόσωπο" π (διακεκριμένο ως προς το P) του συνόλου $\pi + I$. Για να προσθέσουμε και να πολλαπλασιάσουμε κρυπτογραφημένα κείμενα, χρησιμοποιούμε απλώς τις πράξεις δακτυλίου "+" και "×" [3]:

Πρόσθεση: $(pk, \psi_1, \psi_2) = \psi_1 + \psi_2 \in (\pi_1 + \pi_2) + I$

Πολλαπλασιασμός: $(pk, \psi_1, \psi_2) = \psi_1 \times \psi_2 \in (\pi_1 \times \pi_2) + I$

Οι πράξεις δακτυλίου σε κρυπτογραφήματα προκαλούν πράξεις mod-I στα υποκείμενα κρυπτογραφήματα. Γενικά, για ένα αριθμητικό mod-I κύκλωμα C, θα είχαμε:

$\text{Evaluate}_E\ (pk, C, \psi_1, \ldots, \psi t) \in C(\pi_1, \ldots, \pi_1) + I$ [3]

Η σημασιολογική ασφάλεια αυτού του συστήματος βασίζεται στη δυσκολία ενός προβλήματος ιδανικής συμμετοχής, δηλαδή δεδομένων των π' και ψ, είναι $\psi - \pi' \in I$. Αυτή είναι η προσέγγιση του συστήματος Polly Cracker από τους Fellows και Koblitz [42].

Για να το λύσει αυτό ο Gentry χρησιμοποίησε μια διαφορετική προσέγγιση που περιλαμβάνει δύο ιδεώδη [3]. Ο καθένας μπορεί να χρησιμοποιήσει ένα κοινό ιδεώδες I, που αντιπροσωπεύεται από τη βάση B_I . Στη συνέχεια, κάθε χρήστης δημιουργεί το δικό του ιδεώδες J, με μυστική και δημόσια βάση $B_J(sk)$ και $B_J(pk)$, το οποίο J είναι σχετικά πρώτο στο I (δηλαδή, $I + J = R$). Όπως και προηγουμένως, ο χώρος απλού κειμένου P αποτελείται από διακεκριμένους αντιπροσώπους των συνόλων του I. Το δημόσιο κλειδί pk περιλαμβάνει επίσης την περιγραφή μιας κατανομής D. Για την κρυπτογράφηση $\pi \in P$, ο κρυπτογράφος θέτει $\pi* D \leftarrow \pi + I$, και στέλνει $\psi \leftarrow \pi\ \pi* \bmod B_J(pk)$. Με άλλα λόγια, το κρυπτογραφημένο κείμενο έχει τη μορφή $\psi = \pi + i + j$ για $i \in I$ και $j \in J$, όπου το $\pi + i$ προέρχεται από την καθορισμένη κατανομή D. Ο αποκρυπτογράφος θέτει: $\pi \leftarrow (\psi \bmod B_J(sk)) \bmod B_I$.

Για να λειτουργήσει η αποκρυπτογράφηση, το μυστικό κλειδί $B_J(sk)$ πρέπει να επιλεγεί έτσι ώστε να είναι συμβατό με την κατανομή D, έτσι ώστε το $\pi + i$ να είναι πάντα ο διακεκριμένος εκπρόσωπος του $\pi + i + J$ σε σχέση με το $B_J(sk)$. Στην περίπτωση αυτή, η πράξη mod-$B_J(sk)$ επιστρέφει $\pi + i$, οπότε το π ανακτάται εύκολα. Αυτό το κριτήριο αποκρυπτογράφησης γίνεται πιο περίπλοκο καθώς προσθέτουμε και πολλαπλασιάζουμε κρυπτογραφημένα κείμενα χρησιμοποιώντας τις βασικές πράξεις δακτυλίου. Για το αριθμητικό κύκλωμα C που χρησιμοποιεί πρόσθεση και πολλαπλασιασμό modulo I (με βάση B_I), έχουμε:

$\text{Evaluate}_E\ (pk, C, \psi_1, \ldots, \psi_t) = C(\psi_1, \ldots, \psi_t) \in C(\pi_1 + i_1, \ldots, \pi_t + i_t) + J$ όπου $i_1, \ldots, i_t \in I$ [3]

Για να μειωθεί η πολυπλοκότητα της αποκρυπτογράφησης χωρίς να επηρεαστεί καθόλου η "αξιολογική ικανότητα" του συστήματος, η προσέγγιση είναι να δοθεί η δυνατότητα στον κρυπτογράφο να ξεκινήσει την αποκρυπτογράφηση, διευκολύνοντας έτσι τον αποκρυπτογράφο [3]. Είναι ενδιαφέρον ότι η ρύθμιση είναι παρόμοια με την κρυπτογραφία με τη βοήθεια διακομιστή, όπου ένας χρήστης μεταφορτώνει κάποιο μέρος μιας υπολογιστικά εντατικής κρυπτογραφικής εργασίας, όπως η αποκρυπτογράφηση, σε έναν μη αξιόπιστο διακομιστή- στην περίπτωσή μας, ο ο ίδιος ο κρυπτογράφος παίζει το ρόλο του διακομιστή.

Το 2013, οι Craig Gentry, Amit Sahai και Brent Waters (GSW) πρότειναν μια νέα τεχνική για την κατασκευή συστημάτων FHE που αποφεύγει ένα ακριβό βήμα "επαναγραμμικοποίησης"

στον ομομορφικό πολλαπλασιασμό. [43] Οι Zvika Brakerski και Vinod Vaikuntanathan παρατήρησαν ότι για ορισμένους τύπους κυκλωμάτων, το κρυπτοσύστημα GSW διαθέτει ακόμη πιο αργό ρυθμό αύξησης του θορύβου και, ως εκ τούτου, καλύτερη απόδοση και ισχυρότερη ασφάλεια.[44] Οι Jacob Alperin-Sheriff και Chris Peikert περιέγραψαν στη συνέχεια μια πολύ αποδοτική τεχνική bootstrapping βασισμένη σε αυτή την παρατήρηση[45].

Αυτές οι τεχνικές βελτιώθηκαν περαιτέρω για την ανάπτυξη αποτελεσματικών παραλλαγών δακτυλίου του κρυπτοσυστήματος GSW: FHEW (2014)[46] και TFHE (2016)[47]. Το σύστημα FHEW ήταν το πρώτο που έδειξε ότι με την ανανέωση των κρυπτοκειμένων μετά από κάθε μεμονωμένη λειτουργία, είναι δυνατή η μείωση του χρόνου εκκίνησης σε κλάσμα του δευτερολέπτου. Το FHEW εισήγαγε μια νέα μέθοδο για τον υπολογισμό Boolean gates σε κρυπτογραφημένα δεδομένα που απλοποιεί σημαντικά το bootstrapping και υλοποίησε μια παραλλαγή της διαδικασίας bootstrapping [45]. Η αποδοτικότητα του FHEW βελτιώθηκε περαιτέρω από το σχήμα TFHE, το οποίο υλοποιεί μια παραλλαγή δακτυλίου της διαδικασίας bootstrapping [48] χρησιμοποιώντας μια μέθοδο παρόμοια με αυτή του FHEW. Οι βελτιώσεις είναι αποτέλεσμα της αντιμετώπισης της αποκρυπτογράφησης ως αριθμητικής συνάρτησης, σε αντίθεση με τις περισσότερες προηγούμενες υλοποιήσεις που αντιμετώπιζαν την αποκρυπτογράφηση ως κύκλωμα boolean. Με τον τρόπο αυτό αποφεύγεται η κυκλική και αναποτελεσματική οδός της κατασκευής ενός ρηχού κυκλώματος και στη συνέχεια της μετατροπής του μέσω του θεωρήματος του Barrington σε ένα πρόγραμμα διακλάδωσης μεγάλου πολυωνυμικού μήκους.

2.5 Βιβλιοθήκες

Η HElib είναι μια βιβλιοθήκη λογισμικού C++ που υλοποιεί την ομομορφική κρυπτογράφηση (HE), συγκεκριμένα το σχήμα Brakerski-Gentry-Vaikuntanathan (BGV) [6], εστιάζοντας στην αποτελεσματική χρήση των τεχνικών Smart-Vercauteren για το packing κρυπτοκειμένου και των βελτιστοποιήσεων Gentry-Halevi-Smart. Το υποκείμενο κρυπτοσύστημα χρησιμεύει ως το ισοδύναμο μιας "πλατφόρμας υλικού" για το HElib, δεδομένου ότι ορίζει ένα σύνολο πράξεων που μπορούν να εφαρμοστούν ομοιομορφικά και καθορίζει το κόστος τους. Αυτή η "πλατφόρμα" είναι ένα περιβάλλον SIMD (κάπως παρόμοιο με το Intel SSE), αλλά με ένα μοναδικό σύστημα μετρήσεων και παραμέτρους. Περιλαμβάνει μια υλοποίηση του ίδιου του σχήματος BGV με όλες τις βασικές ομομορφικές λειτουργίες του, καθώς και ορισμένες διαδικασίες υψηλότερου επιπέδου που υλοποιούν τις διαδικασίες μετακίνησης δεδομένων GHS και απλής γραμμικής άλγεβρας. Μια χρήσιμη αναλογία που πρέπει να έχετε κατά νου είναι να σκεφτείτε ότι το χαμηλότερο επίπεδο της HElib υλοποιεί μια "γλώσσα συναρμολόγησης" η οποία εκτελείται σε μια "πλατφόρμα υλικού" που δίνεται από το υποκείμενο σύστημα HE. Η "πλατφόρμα" ορίζει ένα σύνολο πράξεων που μπορούν να εφαρμοστούν ομοιομορφικά και το κόστος αυτών των πράξεων [36].

Η ομομορφική κρυπτογράφηση (HE) επιτρέπει την εκτέλεση αριθμητικών πράξεων σε κρυπτογραφημένα δεδομένα ακόμη και χωρίς να γνωρίζει ο παραλήπτης το μυστικό κλειδί αποκρυπτογράφησης. Όλα τα σχήματα HE μέχρι σήμερα ακολουθούν κατά προσέγγιση το μοτίβο του πρώτου υποψηφίου του Gentry από το 2009, στο οποίο τα νέα κρυπτογραφημένα κείμενα είναι "θορυβώδη" για να εξασφαλιστεί η ασφάλεια και ο θόρυβος αυτός αυξάνεται με κάθε πράξη μέχρι να γίνει τόσο μεγάλος ώστε να προκαλεί σφάλματα αποκρυπτογράφησης. Αυτό έχει ως αποτέλεσμα ένα "κάπως ομομορφικό" σχήμα κρυπτογράφησης (SWHE) που μπορεί να αξιολογήσει μόνο κυκλώματα χαμηλού βάθους, τα οποία μπορούν στη συνέχεια να μετατραπούν σε ένα "πλήρως ομομορφικό" σχήμα κρυπτογράφησης (FHE) χρησιμοποιώντας bootstrapping.

Τα χαρακτηριστικά που καθορίζουν την "πλατφόρμα υλικού" για το HElib είναι κοινά σε πολλά σύγχρονα συστήματα HE, συμπεριλαμβανομένων των παραλλαγών ring-LWE του BGV και του scale-invariant συστήματος του Brakerski, το σύστημα HE με βάση το NTRU, και ίσως ακόμη και ορισμένα συστήματα με βάση το LWE [35].

Δύο εξέχοντα χαρακτηριστικά αυτών των κρυπτοσυστημάτων είναι τα εξής:

Αυξανόμενος θόρυβος: Όλα τα σύγχρονα συστήματα SWHE χρησιμοποιούν θορυβώδη κρυπτογραφήματα, όπου ένα νέο κρυπτογράφημα περιλαμβάνει μια συνιστώσα θορύβου που αυξάνεται με κάθε ομομορφική πράξη, μέχρι να γίνει τόσο μεγάλη ώστε προκαλεί σφάλματα αποκρυπτογράφησης. Ωστόσο, οι διάφορες λειτουργίες έχουν πολύ διαφορετική συμπεριφορά αύξησης του θορύβου. Για παράδειγμα, ο πολλαπλασιασμός αυξάνει τον θόρυβο πολύ περισσότερο από την πρόσθεση [34][36].

Διανύσματα καθαρού κειμένου: Ο χώρος απλού κειμένου αυτών των συστημάτων μπορεί να θεωρηθεί ως ένας διανυσματικός χώρος πάνω σε κάποιο πεπερασμένο πεδίο (ή μια ενότητα πάνω σε έναν πεπερασμένο δακτύλιο). Αυτό σημαίνει ότι κάθε εγγενές απλό κείμενο του κρυπτοσυστήματος αντιστοιχεί σε ένα διάνυσμα τιμών απλού κειμένου που ενδιαφέρει την εφαρμογή. Το υποκείμενο πεδίο (ή ο δακτύλιος) και η διάσταση του διανύσματος προκύπτουν αμφότερα από ορισμένες παραμέτρους των κρυπτοσυστημάτων. Όταν χρησιμοποιούμε τέτοια κρυπτοσυστήματα για συγκεκριμένους ομομορφικούς υπολογισμούς, αντιμετωπίζουμε συνήθως ένα πρόβλημα βελτιστοποίησης 2 παραμέτρων, προσπαθώντας να ελαχιστοποιήσουμε τόσο την αύξηση του θορύβου όσο και τον χρόνο εκτέλεσης [34].

Το PALISADE είναι ένα έργο ανοικτού κώδικα που παρέχει αποδοτικές υλοποιήσεις δομικών στοιχείων κρυπτογραφίας πλέγματος και κορυφαίων ομομορφικών συστημάτων κρυπτογράφησης. Υιοθέτησε τις αρχές ανοικτού αρθρωτού σχεδιασμού της προηγούμενης βιβλιοθήκης λογισμικού SIPHER από το πρόγραμμα PROCEED της DARPA. Η ανάπτυξη του SIPHER ξεκίνησε το 2010, με έμφαση στις αρχές αρθρωτής ανοικτής σχεδίασης για την υποστήριξη της ταχείας ανάπτυξης εφαρμογών σε πολλαπλά σχήματα FHE και back-ends επιταχυντών υλικού, συμπεριλαμβανομένων των κινητών, FPGA και υπολογιστικών συστημάτων

που βασίζονται σε CPU. Άρχισε να βασίζεται σε προηγούμενα σχέδια του SIPHER το 2014, με μια έκδοση ανοικτού κώδικα το 2017 και ουσιαστικές βελτιώσεις κάθε επόμενο εξάμηνο. Η ανάπτυξη του χρηματοδοτήθηκε αρχικά από τα προγράμματα DARPA PROCEED και SafeWare, ενώ οι μετέπειτα βελτιώσεις χρηματοδοτήθηκαν από πρόσθετα προγράμματα DARPA, IARPA, NSA, NIH, ONR, Πολεμικό Ναυτικό των ΗΠΑ, Sloan Foundation και εμπορικές οντότητες όπως η Duality Technologies. Χρησιμοποιήθηκε στη συνέχεια σε εμπορικές προσφορές, όπως από την Duality Technologies, η οποία συγκέντρωσε χρηματοδότηση σε ένα seed round και αργότερα σε ένα γύρο σειράς Α υπό την ηγεσία της Intel Capital [37][38].

Είναι σχεδιασμένο για ευχρηστία, παρέχοντας απλούστερα APIs, modularity, υποστήριξη πολλαπλών πλατφορμών και ενσωμάτωση επιταχυντών υλικού. Τα επίπεδα του αναφέρονται στην συνέχεια. Στρώμα μαθηματικών πράξεων που υποστηρίζει αρθρωτή αριθμητική χαμηλού επιπέδου, αριθμοθεωρητικούς μετασχηματισμούς και δειγματοληψία ακέραιων αριθμών, αυτό το στρώμα έχει υλοποιηθεί ώστε να είναι φορητό σε πολλαπλά υποστρώματα υπολογισμού υλικού. Στρώμα λειτουργιών πλέγματος που υποστηρίζει λειτουργίες πλέγματος, άλγεβρα δακτυλίου και δειγματοληψία παγίδων πλέγματος. Στρώμα κρυπτογράφησης που περιέχει αποδοτικές υλοποιήσεις σχημάτων κρυπτογραφίας πλέγματος. Στρώμα κωδικοποίησης που υποστηρίζει πολλαπλές κωδικοποιήσεις απλού κειμένου για κρυπτογραφικά σχήματα. Από προεπιλογή, η βιβλιοθήκη κατασκευάζεται χωρίς εξωτερικές εξαρτήσεις. Όμως, παρέχονται επίσης επιλογές για να προσθέσει ο χρήστης τις βιβλιοθήκες GMP/NTL, tcmalloc και/ή Intel HEXL τρίτων κατασκευαστών, αν το επιθυμεί. Έχει υλοποιηθεί σε έναν ικανοποιητικό αριθμών γλωσσών προγραμματισμού όπως C++, Javascript/WebAssembly, Python και περιβάλλον FreeBSD.

Συμμορφώνεται με τα πρότυπα ασφαλείας του HomomorphicEncryption.org για την ομομορφική κρυπτογράφηση. Προσφέρεται υπό την άδεια ανοιχτού κώδικα BSD με 2 ρήτρες, καθιστώντας ευκολότερη την ενσωμάτωση και την αναδιανομή του σε προϊόντα. Υποστηρίζει τα σχήματα BGV(Brakerski-Gentry-Vaikuntanathan), BFV (Brakerski/Fan-Vercauteren scheme), CKKS (Cheon-Kim-Kim-Song) και FHEW (Ducas-Micciancio) και μια πιο ασφαλή παραλλαγή του σχήματος TFHE(Chillotti-Gama-Georgieva-Izabachene), συμπεριλαμβανομένης της εκκίνησης. Παρέχει επίσης μετα-κβαντική κρυπτογράφηση δημόσιου κλειδιού, επανακρυπτογράφηση μεσολάβησης, κατώτατο όριο FHE (Fully homomorphic encryption) για υπολογισμούς πολλαπλών μερών, κρυπτογράφηση με βάση την ταυτότητα, κρυπτογράφηση με βάση τα χαρακτηριστικά και υποστήριξη ψηφιακής υπογραφής [37][38].

Το Microsoft SEAL είναι μια εύχρηστη βιβλιοθήκη ομομορφικής κρυπτογράφησης ανοικτού κώδικα (με άδεια MIT) που αναπτύχθηκε από την ομάδα έρευνας κρυπτογραφίας και απορρήτου της Microsoft. Η ανάπτυξη του Microsoft SEAL προήλθε αρχικά από την εργασία των Cryptonets, η οποία έδειξε ότι οι αλγόριθμοι τεχνητής νοημοσύνης μπορούν να εκτελούνται σε ομοιομορφικά κρυπτογραφημένα δεδομένα[39][40].

Το Microsoft SEAL είναι γραμμένο σε σύγχρονη τυπική C++ και είναι εύκολο να μεταγλωττιστεί και να εκτελεστεί σε πολλά διαφορετικά περιβάλλοντα. Ένα επίσημο .NET wrapper γραμμένο σε C# είναι διαθέσιμο και διευκολύνει την αλληλεπίδραση των .NET

εφαρμογών με το SEAL. Η διαθεσιμότητα του όμως δεν περιορίζεται μόνο εκεί έχει επεκταθεί και σε Python, Javascript και Typescript. Το Microsoft SEAL δεν έχει απαιτούμενες εξαρτήσεις, αλλά ορισμένα προαιρετικά χαρακτηριστικά μπορούν να ενεργοποιηθούν κατά τη μεταγλώττιση με υποστήριξη συγκεκριμένων βιβλιοθηκών τρίτων κατασκευαστών. Κατά τη χειροκίνητη δημιουργία, μπορεί κανείς να επιλέξει να ζητήσει από το σύστημα δημιουργίας του Microsoft SEAL να κατεβάσει και να δημιουργήσει τις εξαρτήσεις ή εναλλακτικά να αναζητήσει στους καταλόγους του συστήματος για προεγκατεστημένες εξαρτήσεις. Οι προαιρετικές εξαρτήσεις είναι οι εξής: Intel HEXL, Microsoft GSL, ZLIB, Zstandard, GoogleTest και GoogleBenchmark. Το σύστημα CKKS μπορεί να μην είναι φιλικό προς τους αρχάριους. Ακόμα και σχετικά απλοί υπολογισμοί μπορεί να είναι δύσκολο να λειτουργήσουν λόγω των περιορισμών της λειτουργίας αναβαθμολόγησης και της απαίτησης ευθυγράμμισης των κλιμάκων σε διαφορετικά επίπεδα. Η ομάδα SEAL της Microsoft δημιούργησε επίσης ένα νέο εργαλείο μεταγλώττισης με την ονομασία EVA που βοηθά στην επίλυση αυτών των προκλήσεων σε μεγάλο βαθμό. Το EVA επιτρέπει στους προγραμματιστές να εκφράζουν τους επιθυμητούς κρυπτογραφημένους υπολογισμούς στην Python. Βελτιστοποιεί τους υπολογισμούς για το Microsoft SEAL, επιλέγει τις κατάλληλες παραμέτρους κρυπτογράφησης και παρέχει ένα βολικό API Python για την κρυπτογράφηση της εισόδου, την εκτέλεση του υπολογισμού και την αποκρυπτογράφηση του αποτελέσματος [39].

Επιτρέπει την εκτέλεση προσθέσεων και πολλαπλασιασμών σε κρυπτογραφημένους ακέραιους ή πραγματικούς αριθμούς. Άλλες λειτουργίες, όπως η κρυπτογραφημένη σύγκριση, η ταξινόμηση ή οι κανονικές εκφράσεις, δεν είναι στις περισσότερες περιπτώσεις εφικτό να αξιολογηθούν σε κρυπτογραφημένα δεδομένα με τη χρήση αυτής της τεχνολογίας. Ως εκ τούτου, μόνο συγκεκριμένα κρίσιμα για την προστασία της ιδιωτικής ζωής τμήματα υπολογισμών στο νέφος των προγραμμάτων επιτρέπει υλοποιούνται με το Microsoft SEAL. Διαθέτει δύο διαφορετικά σχήματα ομομορφικής κρυπτογράφησης με πολύ διαφορετικές ιδιότητες. Τα σχήματα BFV και BGV επιτρέπουν την εκτέλεση modular αριθμητικής σε κρυπτογραφημένους ακέραιους αριθμούς. Το σχήμα CKKS επιτρέπει προσθέσεις και πολλαπλασιασμούς σε κρυπτογραφημένους πραγματικούς ή μιγαδικούς αριθμούς, αλλά αποδίδει μόνο προσεγγιστικά αποτελέσματα. Σε εφαρμογές όπως η άθροιση κρυπτογραφημένων πραγματικών αριθμών, η αξιολόγηση μοντέλων μηχανικής μάθησης σε κρυπτογραφημένα δεδομένα ή ο υπολογισμός αποστάσεων κρυπτογραφημένων θέσεων το CKKS θα είναι μακράν η καλύτερη επιλογή. Για εφαρμογές όπου απαιτούνται ακριβείς τιμές, τα σχήματα BFV και BGV είναι καταλληλότερα στο Microsoft SEAL [39][40].

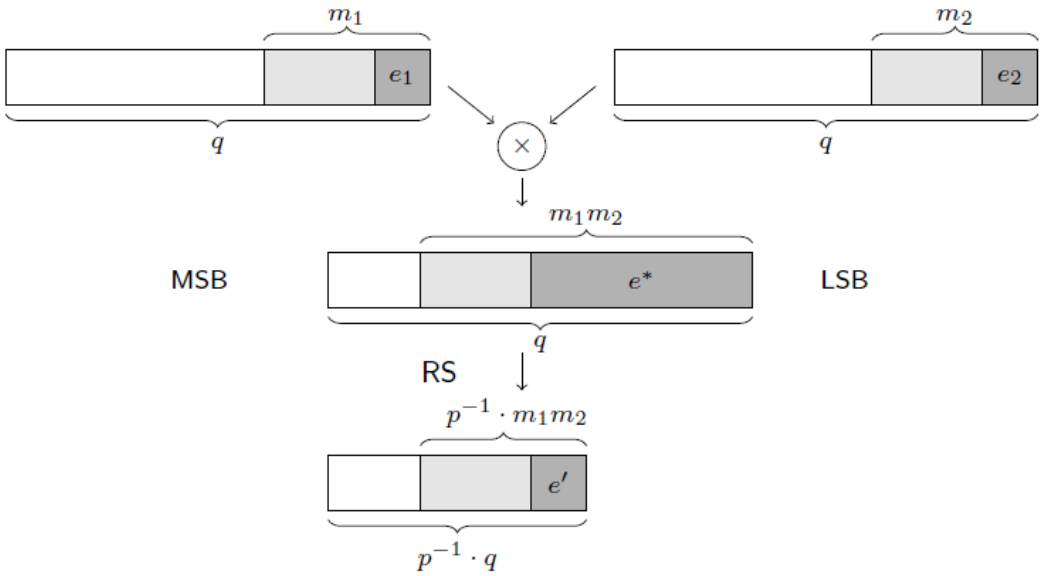
2.6 Σχήμα CKKS

Τα περισσότερα δεδομένα που αφορούν τον πραγματικό κόσμο περιέχουν αποκλίσεις από τις πραγματικές τους τιμές. Για παράδειγμα, σε στατιστικές μελέτες μόνο ένα δείγμα ολόκληρου

του πληθυσμού παρατηρείται, οπότε υπάρχει ένα σφάλμα στην τιμή της ποσότητας. Στην πράξη, τα δεδομένα πρέπει να είναι διακριτά σε μια κατά προσέγγιση τιμή όπως έναν αριθμό κινητής υποδιαστολής, ώστε να αντιπροσωπεύεται από έναν ορισμένο αριθμό bit στα συστήματα υπολογιστών. Σε αυτήν την περίπτωση, εφαρμόζοντας μια μικρή στρογγυλοποίηση στα αρχικά δεδομένα, το σφάλμα δεν έχει μεγάλη επίδραση στο αποτέλεσμα υπολογισμού [6].

Η βασική ιδέα του CKKS είναι ότι θεωρούμε τον «θόρυβο» που προκύπτει από τις πράξεις ως υπολογιστική απόκλιση. Μπορούμε να «θυσιάσουμε» μερικά λιγότερο σημαντικά bits του αποτελέσματος ώστε να διατηρηθεί το μέγεθος σταθερό. Αποθηκεύονται μερικά σημαντικά ψηφία στα πιο σημαντικά bits, τα MSB (most significant bits) και εκτελούνται αριθμητικές πράξεις μεταξύ τους. Η προκύπτουσα τιμή πρέπει να στρογγυλοποιηθεί και πάλι αφαιρώντας μερικά ανακριβή bit με την μικρότερη σημασία (LSB) για να διατηρηθεί το μέγεθος bit του τμήματος του αριθμού που περιέχει τα σημαντικά ψηφία (mantissa). Η στρογγυλοποίηση αυτή δεν ήταν εύκολη να γίνει σε προηγούμενα HE συστήματα διότι δεν μπορεί να αναπαρασταθεί εύκολα ως πολυώνυμο μικρού βαθμού. Προηγούμενες προσπάθειες στρογγυλοποίησης σε ομομορφική κρυπτογράφηση χρειάστηκαν το μέγεθος σε bit του κρυπτοκειμένου να είναι εκθετικά μεγαλύτερο σε σχέση με το βάθος του κυκλώματος [6].

Το κρυπτοκείμενο c ενός μηνύματος m, κρυπτογραφημένο με το κρυφό κλειδί sk, θα έχει μια αποκρυπτογράφηση της μορφής (c, sk) = m + e (mod q) όπου e είναι ένα μικρό σφάλμα που εισάγεται για να εγγυηθεί την ασφάλεια προβλημάτων που βασίζονται στην δυσκολία υπολογισμού όπως τα προβλήματα learning with errors (LWE), Ring-LWE (RLWE) και NTRU. Αν το e είναι αρκετά μικρό σε σχέση με το m, τότε ο θόρυβος δεν θα χαλάσει τα σημαντικά bit του m και η τιμή αθροίσματος m + e μπορεί να αντικαταστήσει το αρχικό μήνυμα κατά προσέγγιση [6].



Σχήμα 2: Πράξη πολλαπλασιασμού και διαδικασία rescaling στο CKKS [6]

Στην αποκρυπτογράφηση, το αποτέλεσμα διατηρείται πάντα μικρό σε σχέση με το ciphertext modulus με σκοπό να είναι σε μέγεθος μικρότερο από το q. Ωστόσο, το μέγεθος των bit αυξάνεται εκθετικά με το βάθος του κυκλώματος. Για να το αντιμετωπίσουν αυτό οι Brakerski et al. χρησιμοποιούν μια τεχνική που ονομάζουν rescaling (σχήμα 2). Για ένα κρυπτοκείμενο c ενός μηνύματος m, τέτοιο ώστε (c, sk) = m + e (mod q) η διαδικασία rescaling έχει ως έξοδο ένα κρυπτοκείμενο $[p^{-1} . c]$ (mod q/p), το οποίο είναι μια σωστή κρυπτογράφηση του m/p με θόρυβο e/p. Έτσι μειώνεται το μέγεθος του ciphertext modulus και επομένως αφαιρεί το σφάλμα ποτ βρίσκεται στα LSBs του μηνύματος. Η διαδικασία αυτή είναι παρόμοια με την στρογγυλοποίηση ενός αριθμού κινητής υποδιαστολής διατηρώντας όμως κάποια ακρίβεια.

Η ακρίβεια που χάνεται στο αποτέλεσμα εξαρτάται από το βάθος του κυκλώματος και είναι το πολύ ένα bit μεγαλύτερη σε σύγκριση με την μη-κρυπτογραφημένη μορφή. Εάν υπάρχουν d κρυπτοκείμενα με ακρίβεια n bits το CKKS με βάθος κυκλώματος $\lceil \log(d) \rceil$ μπορεί να υπολογίσει το γινόμενο τους με (n – log (d) -1) bits ακριβείας εκτελώντας d πολλαπλασιασμούς. Σε μη κρυπτογραφημένη μορφή, το mantissa μπορεί να υπολογιστεί με πολλαπλασιασμό χρησιμοποιώντας (n – log (d)) bits ακριβείας. Συγκρίνοντας το CKKS με μεθόδους που σχεδιάστηκαν πριν από αυτό, ο ίδιος υπολογισμός χρειάζεται $\Omega(n^2 d)$ ομομορφικές πράξεις χρησιμοποιώντας bitwise κρυπτογράφηση ή πολύ μεγαλύτερο μέγεθος plaintext ή τεχνικές bootstrapping που επίσης επιφέρουν ένα κόστος ως προς την απόδοση. Μετά την διαδικασία του rescaling, το μεγαλύτερο πιθανό ciphertext modulus μπορεί να μειωθεί σε μέγεθος O(n log(d)). Οι παράμετροι είναι μικρότερες από τις προηγούμενες μεθόδους και αυτό το πλεονέκτημα δίνει τη δυνατότητα να εκτελεστούν αποδοτικά κατά προσέγγιση υπολογισμοί υπερβατικών συναρτήσεων όπως η εκθετική, λογαριθμικές και τριγωνομετρικές συναρτήσεις με τον υπολογισμό της επέκτασης της σειράς Taylor τους. Συγκεκριμένα, οι Brakerski et al. προτείνουν έναν ειδικό αλγόριθμο για τον υπολογισμό του πολλαπλασιαστικού αντίστροφου με μειωμένη πολυπλοκότητα, ο οποίος επιτρέπει την αποτελεσματική αξιολόγηση των ορθολογικών συναρτήσεων.

Σε ένα περιβάλλον cloud computing, ένας μεγάλος όγκος δεδομένων δημιουργείται και μαζί δημιουργείται η ανάγκη να χειριστούμε αυτά τα δεδομένα. Το CKKS θα μπορούσε να είναι μια πρακτική λύση ανάλυσης δεδομένων καθώς επιτρέπει την κρυπτογράφηση πολλών πληροφοριών σε ένα μόνο κρυπτογραφημένο κείμενο ώστε να μπορούμε να παραλληλοποιήσουμε τόσο τον χώρο όσο και τον υπολογισμό ταυτόχρονα.

3 *Πολλαπλασιασμός πινάκων*

Στα μαθηματικά, ένας πίνακας είναι μια ορθογώνια διάταξη αριθμών, συμβόλων, ή εκφράσεων, διατεταγμένων σε σειρές και στήλες. Τα μεμονωμένα στοιχεία σε ένα πίνακα ονομάζονται στοιχεία ή εγγραφές του. Ο κανόνας για τον πολλαπλασιασμό πινάκων είναι ότι οι δύο πίνακες μπορούν να πολλαπλασιαστούν μόνο όταν ο αριθμός των στηλών του πρώτου ισούται με τον αριθμό των γραμμών του δευτέρου. Η χρησιμότητα των πινάκων οφείλεται στον ιδιόμορφο τρόπο με τον οποίο πολλαπλασιάζουμε πίνακες [13].

Αν ο A είναι ένας m × n πίνακας, και ο B ένας n × p πίνακας, τότε το γινόμενο πινάκων C = A · B θα είναι ένας πίνακας m × p:

$c_{ij} = a_{i1}\, b_{1j} + a_{i2}\, b_{2j} + \ldots + a_{in}\, b_{nj} = \sum_{k=1}^{n} a_{ik} b_{kj}$

3.1 Ρόλος στα νευρωνικά δίκτυα

Ένα νευρωνικό δίκτυο είναι ένα δίκτυο ή κύκλωμα βιολογικών νευρώνων ή με τη σύγχρονη έννοια, ένα τεχνητό νευρωνικό δίκτυο, που αποτελείται από τεχνητούς νευρώνες ή κόμβους. Έτσι, ένα νευρωνικό δίκτυο είναι είτε ένα βιολογικό νευρωνικό δίκτυο, που αποτελείται από βιολογικούς νευρώνες, είτε ένα τεχνητό νευρωνικό δίκτυο, που χρησιμοποιείται για την επίλυση προβλημάτων τεχνητής νοημοσύνης (AI). Οι συνδέσεις του βιολογικού νευρώνα μοντελοποιούνται στα τεχνητά νευρωνικά δίκτυα ως βάρη μεταξύ των κόμβων. Ένα θετικό βάρος αντικατοπτρίζει μια διεγερτική σύνδεση, ενώ αρνητικές τιμές σημαίνουν ανασταλτικές συνδέσεις. Όλες οι είσοδοι τροποποιούνται με ένα βάρος και αθροίζονται. Αυτή η δραστηριότητα αναφέρεται ως γραμμικός συνδυασμός. Τέλος, μια συνάρτηση ενεργοποίησης ελέγχει το πλάτος της εξόδου. Για παράδειγμα, ένα αποδεκτό εύρος εξόδου είναι συνήθως μεταξύ 0 και 1, ή θα μπορούσε να είναι -1 και 1 [19].

Ένα ANN (Artificial Neural Network) βασίζεται σε μια συλλογή συνδεδεμένων μονάδων ή κόμβων που ονομάζονται τεχνητοί νευρώνες, οι οποίοι μοντελοποιούν χαλαρά τους νευρώνες ενός βιολογικού εγκεφάλου. Κάθε σύνδεση, όπως οι συνάψεις σε έναν βιολογικό εγκέφαλο, μπορεί να μεταδώσει ένα σήμα σε άλλους νευρώνες. Ένας τεχνητός νευρώνας λαμβάνει σήματα, στη συνέχεια τα επεξεργάζεται και μπορεί να δώσει σήμα στους νευρώνες που συνδέονται με αυτόν. Το "σήμα" σε μια σύνδεση είναι ένας πραγματικός αριθμός και η έξοδος κάθε νευρώνα υπολογίζεται από κάποια μη γραμμική συνάρτηση του αθροίσματος των εισόδων του. Οι συνδέσεις ονομάζονται ακμές. Οι νευρώνες και οι ακμές έχουν συνήθως ένα βάρος που προσαρμόζεται καθώς προχωρά η μάθηση. Το βάρος αυξάνει ή μειώνει την ισχύ του σήματος σε μια σύνδεση. Οι νευρώνες μπορεί να έχουν ένα κατώφλι, έτσι ώστε ένα σήμα να αποστέλλεται μόνο εάν το συνολικό σήμα διασχίζει αυτό το κατώφλι. Συνήθως, οι νευρώνες συγκεντρώνονται σε επίπεδα. Διαφορετικά στρώματα μπορούν να εκτελούν διαφορετικούς μετασχηματισμούς στις εισόδους τους. Τα σήματα ταξιδεύουν από το πρώτο στρώμα (το στρώμα εισόδου), στο τελευταίο στρώμα (το στρώμα εξόδου), ενδεχομένως αφού διασχίσουν τα στρώματα πολλές φορές [20].

Αυτά τα τεχνητά δίκτυα μπορούν να χρησιμοποιηθούν για προγνωστική μοντελοποίηση, προσαρμοστικό έλεγχο και εφαρμογές όπου μπορούν να εκπαιδευτούν μέσω ενός συνόλου δεδομένων. Η αυτομάθηση που προκύπτει από την εμπειρία μπορεί να συμβεί μέσα στα δίκτυα, τα οποία μπορούν να εξάγουν συμπεράσματα από ένα πολύπλοκο και φαινομενικά άσχετο σύνολο πληροφοριών [21].

Πρόσφατα έχει αποδειχθεί ότι τα μοντέλα που βασίζονται σε βαθιά νευρωνικά δίκτυα επιτυγχάνουν μεγάλη επιτυχία σε μια σειρά από εφαρμογές υγειονομικής περίθαλψης [33], και ένα φυσικό ερώτημα είναι αν μπορούμε να αναθέσουμε την εκμάθηση τέτοιων μοντέλων σε τρίτους και να αξιολογήσουμε νέα δείγματα με ασφαλή τρόπο. Αυτό έρχεται να λύσει ο πολλαπλασιασμός των πινάκων με το ακόλουθο σενάριο. Ο ιδιοκτήτης των δεδομένων εκπαιδεύει ένα μοντέλο και το καθιστά διαθέσιμο σε έναν πάροχο υπηρεσιών πληροφορικής, ώστε να χρησιμοποιηθεί για την πραγματοποίηση προβλέψεων σχετικά με κρυπτογραφημένες εισόδους από άλλους κατόχους δεδομένων. Εναλλακτικά, ένας πάροχος υπηρεσιών υπολογιστικού νέφους εκπαιδεύει ένα μοντέλο σε κρυπτογραφημένα δεδομένα από ορισμένους ιδιοκτήτες δεδομένων και χρησιμοποιεί το κρυπτογραφημένο εκπαιδευμένο μοντέλο για να κάνει προβλέψεις σε νέες κρυπτογραφημένες εισόδους.

Υπάρχουν δύο διαφορετικοί υπολογισμοί στα νευρωνικά δίκτυα, η τροφοδότηση προς τα εμπρός και η οπισθοδιάδοση. Οι υπολογισμοί τους είναι παρόμοιοι στο ότι και οι δύο χρησιμοποιούν κανονικό πολλαπλασιασμό πινάκων, δεν είναι απαραίτητος ούτε το γινόμενο Hadamard ούτε το γινόμενο Kronecker. Ωστόσο, ορισμένες υλοποιήσεις μπορούν να χρησιμοποιήσουν το γινόμενο Hadamard για τη βελτιστοποίηση της υλοποίησης. [24][28]. Ένα νευρωνικό δίκτυο με τροφοδότηση (FNN) είναι ένα τεχνητό νευρωνικό δίκτυο στο οποίο οι συνδέσεις μεταξύ των κόμβων δεν σχηματίζουν κύκλο. Ως εκ τούτου, διαφέρει από τον απόγονό του: τα επαναλαμβανόμενα νευρωνικά δίκτυα.Το νευρωνικό δίκτυο τροφοδότησης ήταν ο πρώτος και απλούστερος τύπος τεχνητού νευρωνικού δικτύου που επινοήθηκε. Σε αυτό το δίκτυο, η πληροφορία κινείται προς μία μόνο κατεύθυνση - προς τα εμπρός - από τους κόμβους εισόδου,

μέσω των κρυφών κόμβων (εάν υπάρχουν) και προς τους κόμβους εξόδου. Δεν υπάρχουν κύκλοι ή βρόχοι στο δίκτυο [24][25]. Το απλούστερο είδος νευρωνικού δικτύου είναι ένα δίκτυο perceptron ενός στρώματος, το οποίο αποτελείται από ένα μόνο στρώμα κόμβων εξόδου- οι είσοδοι τροφοδοτούνται απευθείας στις εξόδους μέσω μιας σειράς βαρών. Το άθροισμα των γινομένων των βαρών και των εισόδων υπολογίζεται σε κάθε κόμβο, και αν η τιμή είναι πάνω από κάποιο κατώφλι (συνήθως 0) ο νευρώνας πυροδοτείται και παίρνει την ενεργοποιημένη τιμή (συνήθως 1) διαφορετικά παίρνει την απενεργοποιημένη τιμή (συνήθως -1). Οι νευρώνες με αυτού του είδους τη συνάρτηση ενεργοποίησης ονομάζονται επίσης τεχνητοί νευρώνες ή μονάδες γραμμικού κατωφλίου[26]. Ένα perceptron μπορεί να δημιουργηθεί χρησιμοποιώντας οποιεσδήποτε τιμές για τις ενεργοποιημένες και απενεργοποιημένες καταστάσεις, αρκεί η τιμή κατωφλίου να βρίσκεται μεταξύ των δύο. Τα perceptron μπορούν να εκπαιδευτούν με έναν απλό αλγόριθμο μάθησης που συνήθως ονομάζεται κανόνας δέλτα. Υπολογίζει τα σφάλματα μεταξύ της υπολογιζόμενης εξόδου και των δειγματικών δεδομένων εξόδου και τα χρησιμοποιεί για να δημιουργήσει μια προσαρμογή των βαρών, εφαρμόζοντας έτσι μια μορφή βαθμωτής καθόδου. Τα perceptrons ενός στρώματος είναι ικανά να μαθαίνουν μόνο γραμμικά διαχωρίσιμα μοτίβα- το 1969, σε μια διάσημη μονογραφία με τίτλο Perceptrons, οι Marvin Minsky και Seymour Papert έδειξαν ότι ήταν αδύνατο για ένα δίκτυο perceptron ενός στρώματος να μάθει μια συνάρτηση XOR (παρ' όλα αυτά, ήταν γνωστό ότι τα perceptrons πολλαπλών στρωμάτων είναι ικανά να παράγουν κάθε δυνατή συνάρτηση boolean). Παρόλο που η υπολογιστική ισχύς μιας μεμονωμένης μονάδας κατωφλίου είναι αρκετά περιορισμένη, έχει αποδειχθεί ότι τα δίκτυα παράλληλων μονάδων κατωφλίου μπορούν να προσεγγίσουν οποιαδήποτε συνεχή συνάρτηση από ένα συμπαγές διάστημα των πραγματικών αριθμών στο διάστημα [-1,1]. Αυτό το αποτέλεσμα μπορεί να βρεθεί στο Peter Auer, Harald Burgsteiner και Wolfgang Maass "A learning rule for very simple universal approximators consisting of a single layer of perceptrons" [27].

Αυτό μας οδηγεί ωστόσο στα Perceptron πολλαπλών στρωμάτων. Αυτή η κατηγορία δικτύων αποτελείται από πολλαπλά στρώματα υπολογιστικών μονάδων, συνήθως διασυνδεδεμένων μεταξύ τους με τρόπο "feed-forward". Κάθε νευρώνας σε ένα στρώμα έχει κατευθυνόμενες συνδέσεις με τους νευρώνες του επόμενου στρώματος. Σε πολλές εφαρμογές οι μονάδες αυτών των δικτύων εφαρμόζουν μια σιγμοειδή συνάρτηση ως συνάρτηση ενεργοποίησης. Ωστόσο, οι σιγμοειδείς συναρτήσεις ενεργοποίησης έχουν πολύ μικρές τιμές παραγώγου εκτός ενός μικρού εύρους και δεν λειτουργούν καλά στα βαθιά νευρωνικά δίκτυα λόγω του προβλήματος της εξαφανιζόμενης κλίσης. Έχουν προταθεί εναλλακτικές λύσεις για τις σιγμοειδείς συναρτήσεις ενεργοποίησης που ανακουφίζουν τα προβλήματα της εξαφανιζόμενης κλίσης και επιτρέπουν την εκπαίδευση βαθιών δικτύων [29][30][31]. Το καθολικό θεώρημα προσέγγισης για τα νευρωνικά δίκτυα δηλώνει ότι κάθε συνεχής συνάρτηση που απεικονίζει διαστήματα πραγματικών αριθμών σε κάποιο διάστημα εξόδου πραγματικών αριθμών μπορεί να προσεγγιστεί αυθαίρετα στενά από ένα perceptron πολλαπλών στρωμάτων με ένα μόνο κρυφό στρώμα. Το αποτέλεσμα αυτό ισχύει για ένα ευρύ φάσμα συναρτήσεων ενεργοποίησης, π.χ. για τις σιγμοειδείς συναρτήσεις. Τα δίκτυα πολλαπλών στρωμάτων χρησιμοποιούν διάφορες τεχνικές μάθησης, με πιο δημοφιλή την οπισθοδιάδοση.

Στη μηχανική μάθηση, ο αλγόριθμος οπισθοδιάδοσης (backpropagation) είναι ένας ευρέως χρησιμοποιούμενος αλγόριθμος για την εκπαίδευση νευρωνικών δικτύων τροφοδότησης. Υπάρχουν γενικεύσεις του backpropagation για άλλα τεχνητά νευρωνικά δίκτυα (ANN) και γενικά για συναρτήσεις. Όλες αυτές οι κατηγορίες αλγορίθμων αναφέρονται γενικά ως "backpropagation". Κατά την προσαρμογή ενός νευρωνικού δικτύου, ο αλγόριθμος backpropagation υπολογίζει την κλίση της συνάρτησης απωλειών ως προς τα βάρη του δικτύου για ένα μόνο παράδειγμα εισόδου-εξόδου, και το κάνει αποτελεσματικά, σε αντίθεση με έναν αφελή άμεσο υπολογισμό της κλίσης ως προς κάθε βάρος ξεχωριστά. Αυτή η αποδοτικότητα καθιστά εφικτή τη χρήση μεθόδων κλίσης για την εκπαίδευση πολυεπίπεδων δικτύων, ενημερώνοντας τα βάρη για την ελαχιστοποίηση των απωλειών, συνήθως χρησιμοποιούνται η κάθοδος κλίσης ή παραλλαγές όπως η στοχαστική κάθοδος κλίσης. Ο αλγόριθμος οπισθοδιάδοσης λειτουργεί υπολογίζοντας την κλίση της συνάρτησης απώλειας ως προς κάθε βάρος με τον κανόνα της αλυσίδας, υπολογίζοντας την κλίση ένα στρώμα κάθε φορά, επαναλαμβάνοντας προς τα πίσω από το τελευταίο στρώμα για να αποφευχθούν περιττοί υπολογισμοί των ενδιάμεσων όρων στον κανόνα της αλυσίδας - αυτό είναι ένα παράδειγμα δυναμικού προγραμματισμού. Εδώ, οι τιμές εξόδου συγκρίνονται με τη σωστή απάντηση για τον υπολογισμό της τιμής κάποιας προκαθορισμένης συνάρτησης σφάλματος. Με διάφορες τεχνικές, το σφάλμα στη συνέχεια ανατροφοδοτείται μέσω του δικτύου. Χρησιμοποιώντας αυτή την πληροφορία, ο αλγόριθμος προσαρμόζει τα βάρη κάθε σύνδεσης προκειμένου να μειώσει την τιμή της συνάρτησης σφάλματος κατά κάποιο μικρό ποσό. Αφού επαναληφθεί αυτή η διαδικασία για έναν αρκετά μεγάλο αριθμό κύκλων εκπαίδευσης, το δίκτυο συγκλίνει συνήθως σε κάποια κατάσταση όπου το σφάλμα των υπολογισμών είναι μικρό. Στην περίπτωση αυτή, θα λέγαμε ότι το δίκτυο έχει μάθει μια συγκεκριμένη συνάρτηση-στόχο [28].

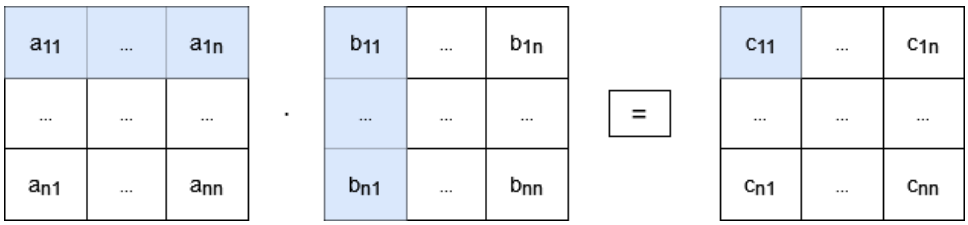
Ωστόσο, σε ένα συνελικτικό νευρωνικό δίκτυο (CNN), τα φίλτρα χρησιμοποιούν μια παραλλαγή του γινομένου Hadamard. Στα μαθηματικά, το γινόμενο Hadamard είναι μια δυαδική πράξη που παίρνει δύο πίνακες των ίδιων διαστάσεων και παράγει έναν άλλο πίνακα της ίδιας διάστασης με τους τελεστές, όπου κάθε στοιχείο i, j είναι το γινόμενο των στοιχείων i, j των δύο αρχικών πινάκων. Πρέπει να διακρίνεται από το πιο συνηθισμένο γινόμενο πινάκων [32]. Τα συνεπτυγμένα νευρωνικά δίκτυα (CNN) προσθέτουν ένα πρόσθετο βήμα φιλτραρίσματος πριν περάσουν από τα βάρη. Περνάει ένα φίλτρο μέσα από τους πίνακες για να πάρει μια τιμή που αντιπροσωπεύει μια γειτονιά γύρω από μια τιμή. Το φίλτρο παίρνει το γινόμενο Hadamard και στη συνέχεια αθροίζει όλα τα στοιχεία του προκύπτοντος πίνακα. Τότε η προκύπτουσα πράξη είναι ένας στοιχειομετρικός πολλαπλασιασμός και πρόσθεση των όρων. Αυτός ο πυρήνας ας τον ονομάσουμε g μετατοπίζεται σε ολόκληρη τη συνάρτηση που ονομάζεται f. Εκτελώντας ένα γινόμενο Hadamard σε κάθε βήμα και στη συνέχεια αθροίζοντας τα στοιχεία. Το γινόμενο Hadamard ορίζεται συχνά για πίνακες με ακριβώς τις ίδιες διαστάσεις, κάτι που χαλαρώνει στα CNN λόγω του ότι το φίλτρο κινείται διαδοχικά μέσα στην εικόνα. Είναι επίσης δυνατό να εκτελεστεί το γινόμενο Hadamard στις άκρες της εικόνας, όπου μπορούν να χρησιμοποιηθούν διαφορετικές τεχνικές συμπλήρωσης.

3.2 Οι τρεις προσεγγίσεις

Στην παρούσα εργασία αποφασίσαμε να υλοποιήσουμε τρεις διαφορετικές προσεγγίσεις για τον πολλαπλασιασμό δυο κρυπτογραφημένων πινάκων με ομομορφικό τρόπο, με σκοπό την σύγκριση των προσεγγίσεων ως προς την απόδοση χρόνου και πόρων αποθήκευσης. Πριν την ανάλυση της υλοποίησής μας, θα παρουσιάσουμε τις τρεις προσεγγίσεις στις οποίες βασιστήκαμε. Στην υλοποίηση μας κάνουμε την παραδοχή ότι οι δυο πίνακες που πολλαπλασιάζονται μεταξύ τους είναι τετραγωνικοί, οπότε η επεξήγηση των προσεγγίσεων και τα παραδείγματα ακολουθούν την ίδια παραδοχή.

3.2.1 Naive τρόπος

Έστω δυο πίνακες A και B οι οποίοι είναι τετραγωνικοί, και ως προϋπόθεση πολλαπλασιασμού ο αριθμός στηλών του A είναι ίδιος με τον αριθμό γραμμών του B. Η διάσταση λοιπόν και των δυο πινάκων είναι n × n.

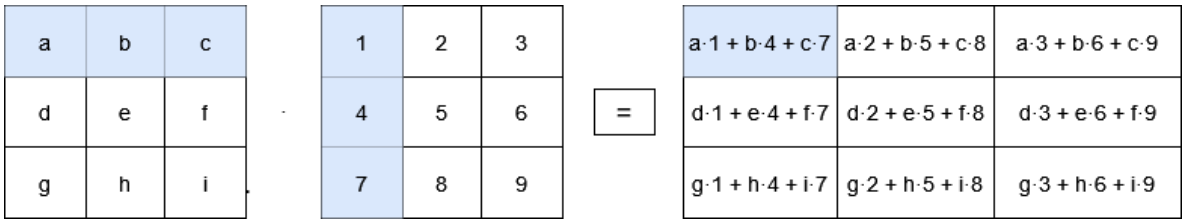


Σχήμα 3: Πολλαπλασιασμός πινάκων με τον απλό τρόπο

Ο πιο απλός τρόπος για να τους πολλαπλασιάσουμε είναι να πολλαπλασιάσουμε την κάθε γραμμή του A με κάθε στήλη του B ώστε να πάρουμε ως αποτέλεσμα τον πίνακα C, ο οποίος είναι επίσης διάστασης n × n. Πιο συγκεκριμένα, σύμφωνα με το [13] κάθε στοιχείο του πίνακα C προκύπτει ως:

$c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + \ldots + a_{in} \cdot b_{nj}$

Παρακάτω εισάγουμε ένα παράδειγμα για μια μικρή διάσταση n = 3.

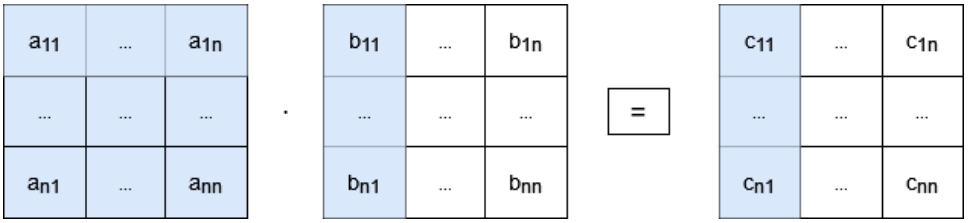


Σχήμα 4: Πολλαπλασιασμός πινάκων με τον απλό τρόπο (παράδειγμα)

Σύμφωνα με αυτή τη προσέγγιση, ο πιο απλός τρόπος να υπολογιστεί το γινόμενο των κρυπτογραφημένων πινάκων είναι να πάρουμε κάθε στοιχείο της $1^{ης}$ γραμμής του πίνακα A, ως ξεχωριστό κρυπτοκείμενο, να το πολλαπλασιάσουμε με το αντίστοιχο στοιχείο της $1^{ης}$ στήλης του πίνακα B, και προσθέτοντας τα n γινόμενα να πάρουμε ως αποτέλεσμα ένα κρυπτογραφημένο στοιχείο του πίνακα C. Η διαδικασία αυτή πρέπει να γίνει συνολικά n^2 φορές ώστε να πάρουμε τα n^2 στοιχεία του πίνακα C και στη συνέχεια να τα αποκρυπτογραφήσουμε.

Γνωρίζουμε ότι ο ομομορφικός πολλαπλασιασμός είναι η πιο χρονοβόρα πράξη, και με τη συγκεκριμένη μέθοδο καλούμαστε να εκτελέσουμε $n^2 \cdot n = n^3$ πολλαπλασιασμούς, και να χρησιμοποιήσουμε ξεχωριστό κρυπτοκείμενο για κάθε ένα στοιχείο ξεχωριστά. Καταλαβαίνουμε ήδη ότι η πρώτη αυτή μέθοδος δεν είναι ιδιαίτερα αποδοτική ούτε ως προς τον χρόνο, ούτε ως προς τον χώρο αποθήκευσης. Στη συνέχεια θα αναλύσουμε δυο πιο αποδοτικές μεθόδους.

3.2.2 Halevi and Shoup

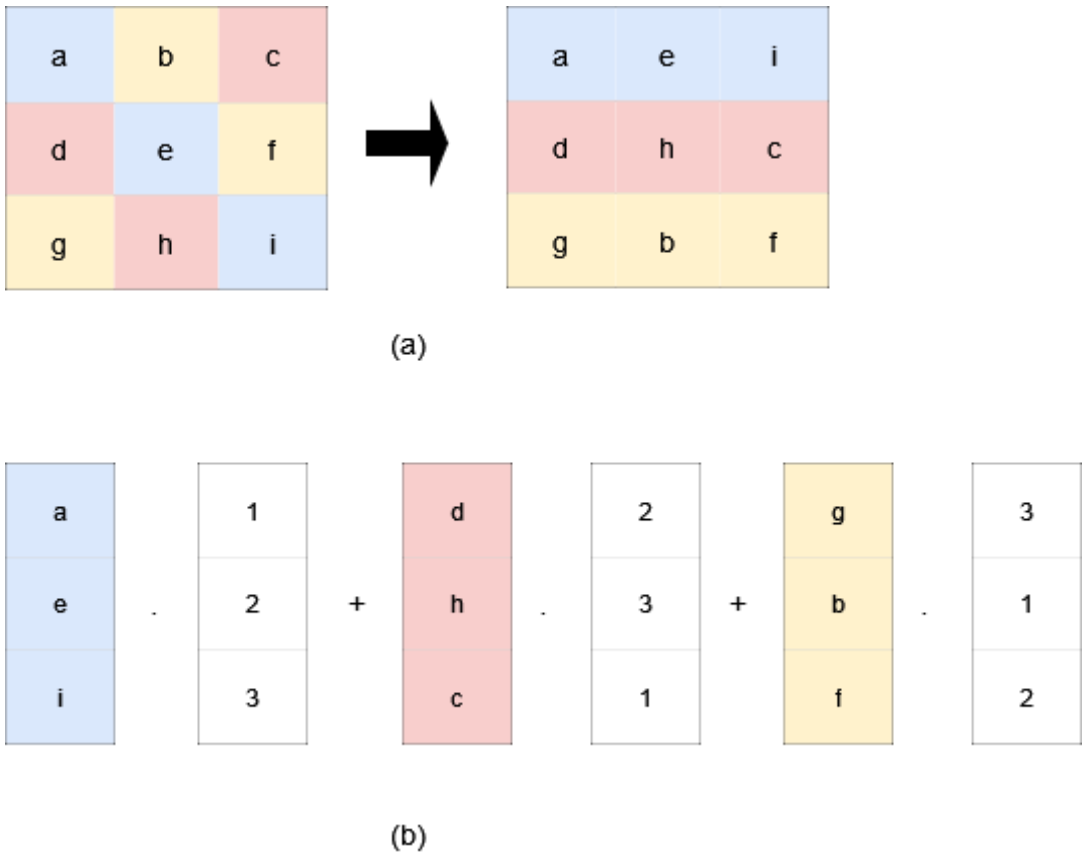


Σχήμα 5: Πολλαπλασιασμός πινάκων με την μέθοδο Halevi & Shoup

Οι Shai Halevi και Victor Shoup εφάρμοσαν μια νέα μέθοδο για πολλαπλασιασμό πίνακα με διάνυσμα [22] η οποία μπορεί να επεκταθεί και στην περίπτωση πολλαπλασιασμού πίνακα με πίνακα. Πιο συγκεκριμένα, η μέθοδος παίρνει ως είσοδο έναν τετραγωνικό πίνακα $n \times n$ και ένα κάθετο διάνυσμα διάστασης n και παράγει ως έξοδο ένα κάθετο διάνυσμα διάστασης n. Όπως βλέπουμε στο σχήμα 4 παραπάνω, με κάθε επανάληψη της μεθόδου παίρνουμε μία στήλη του πίνακα-γινομένου. Οπότε επαναλαμβάνοντας την μέθοδο συνολικά n φορές, μπορούμε να κατασκευάσουμε τον πίνακα C. Την πρώτη φορά πολλαπλασιάζουμε τον πίνακα A με την 1η στήλη του πίνακα B. Την δεύτερη φορά πολλαπλασιάζουμε τον πίνακα A με την 2η στήλη του πίνακα B κ.ο.κ., μέχρι και την n-οστή στήλη.

Η μέθοδος επιχειρεί να μειώσει τον αριθμό των κρυπτοκειμένων που χρειάζονται καθώς και τον αριθμό των πολλαπλασιαστικών πράξεων. Αρχικά γίνεται μια ανακατάταξη των στοιχείων του πίνακα με τρόπο τέτοιο ώστε μετά τον πολλαπλασιασμό SIMD, τα στοιχεία που πρέπει να προστεθούν μεταξύ τους να αποθηκευτούν σε διαφορετικά κρυπτοκείμενα [23]. Για να γίνει αυτό, χρησιμοποιείται η τεχνική packing, που επιτρέπει την κρυπτογράφηση και τον πολλαπλασιασμό ενός ολόκληρου διανύσματος με ένα άλλο διάνυσμα. Σε αυτήν την περίπτωση τα στοιχεία του κάθε διανύσματος πρέπει να βρίσκονται σε σωστές θέσεις ώστε να πολλαπλασιαστεί ή να

προστεθεί κάθε στοιχείο του ενός με κάθε στοιχείο του άλλου με μια μόνο πράξη, σε αντίθεση με την naive μέθοδο η οποία εκτελεί πράξεις με μεμονωμένα στοιχεία. Τα βήματα της μεθόδου Halevi & Shoup παρουσιάζονται πιο αναλυτικά παρακάτω.



Σχήμα 6: Halevi & Shoup, ανακατάταξη των στοιχείων του πίνακα εισόδου (a) και πολλαπλασιασμός με το διάνυσμα εισόδου σε περιστροφή

Στο σχήμα 5 (a) απεικονίζεται ένα παράδειγμα πίνακα εισόδου και η αναδιάταξη των στοιχείων του. Τα στοιχεία διατάσσονται ώστε κάθε διαγώνιος του πίνακα να αποτελεί ένα διάνυσμα-γραμμή. Δηλαδή, τα στοιχεία A_{ij} όπου $i=j$ και $0 \leq i,j < n$ αποτελούν την κύρια διαγώνιο του πίνακα και μετατρέπονται στην πρώτη γραμμή. Η δεύτερη γραμμή θα περιέχει τα στοιχεία της δεύτερης διαγώνιου δηλαδή τα $A_{1,0}$, $A_{2,1}$ … $A_{0,n-1}$. Με τον ίδιο τρόπο απομονώνονται όλες οι διαγώνιοι του πίνακα Α. Αναλυτικός αλγόριθμος για την διαδικασία με τετραγωνικό πίνακα εισόδου αυθαίρετης διάστασης θα παρουσιαστεί στην ενότητα της υλοποίησης.

Έχοντας απομονώσει τις διαγώνιους, και αφού κωδικοποιηθεί με την μέθοδο packing η καθεμία σε ένα κρυπτοκείμενο, τις πολλαπλασιάζουμε με SIMD με το διάνυσμα εισόδου. Το

διάνυσμα εισόδου πρέπει επίσης να κωδικοποιηθεί και να κρυπτογραφηθεί με τον ίδιο τρόπο. Αρχικά πολλαπλασιάζουμε την κύρια διαγώνιο με το διάνυσμα όπως είναι. Στη συνέχεια, περιστρέφουμε το διάνυσμα εισόδου κατά μια θέση, έτσι ώστε όλα τα στοιχεία του να μεταφερθούν μια θέση προς την αρχή. Οπότε το πρώτο στοιχείο θα πάρει την τελευταία θέση, το δεύτερο στοιχείο θα γίνει πρώτο, το τρίτο θα γίνει δεύτερο, κ.ο.κ.. Η επόμενη διαγώνιος του πίνακα πολλαπλασιάζεται με το περιστραμμένο διάνυσμα. Η διαδικασία επαναλαμβάνεται για κάθε διαγώνιο, και το τελικό αποτέλεσμα δίνεται από την πρόσθεση όλων των επιμέρους γινομένων, βλ. σχήμα 5 (b).

3.2.3 Jiang, Kim, Lauter and Song

Το 2019 οι Jiang, Kim, Lauter και Song εφήυραν μια νέα μέθοδο κρυπτογράφησης πινάκων και έναν πρακτικό τρόπο για τον πολλαπλασιασμό τους [5]. Για περισσότερη ευκολία, κατά την διάρκεια της εργασίας θα αναφερόμαστε στην μέθοδο με τα αρχικά των συγγραφέων (JKLS) για να την διαφοροποιήσουμε από τις άλλες δύο. Η λύση τους έρχεται με σκοπό να δώσει ακόμη καλύτερη απόδοση από την μέθοδο των Halevi και Shoup [22], χρησιμοποιώντας καινοτόμες στρατηγικές για την κωδικοποίηση ενός πίνακα και την εκτέλεση βασικών ομομορφικών πράξεων. Σύμφωνα με τους Jiang et al., η μέθοδος τους μπορεί να υποστηριχτεί από τις περισσότερες βιβλιοθήκες ομομορφικής κρυπτογραφίας και καταφέρνει να ολοκληρώσει έναν πολλαπλασιασμό τετραγωνικών πινάκων μεγέθους 64 σε 0.6 δευτερόλεπτα. Επίσης καταφέρνει την αντιμετάθεση τετραγωνικού πίνακα μεγέθους 64 σε 0.09 δευτερόλεπτα. Η μέθοδος τους χρησιμοποιείται στο E2DM framework, το οποίο έχει υλοποιηθεί από τους ίδιους, και αποτελεί το πρώτο framework που υποστηρίζει ασφαλή υπολογισμό στην φάση της πρόβλεψης ενός νευρωνικού δικτύου, βασισμένο τόσο σε κρυπτογραφημένα δεδομένα όσο και σε κρυπτογραφημένο μοντέλο.

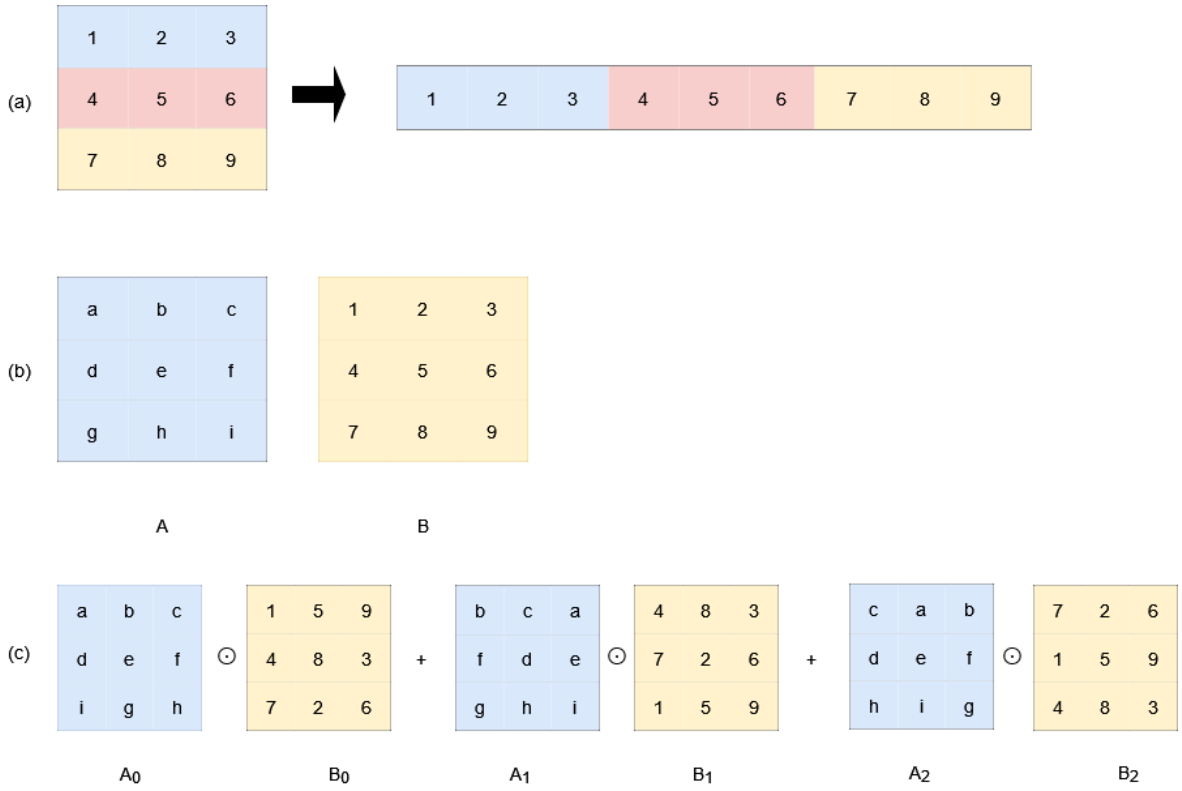
Έστω δυο τετραγωνικοί κρυπτογραφημένοι πίνακες A και B διάστασης n. Συνοπτικά η μέθοδος των Jiang et al. μπορεί να συνοψιστεί με την παρακάτω ισότητα:

$$A \cdot B = \sum_{i=0}^{d-1} Ai \odot Bi \quad (1)$$

Όπου με το σύμβολο ⊙ εννοούμε τον πολλαπλασιασμό αντίστοιχων στοιχείων μεταξύ των πινάκων (Hadamard multiplication), και όπου A_i, B_i ορίζονται διάφορες ανακατατάξεις των πινάκων A και B.

Η μέθοδος διαφοροποιείται από τις προηγούμενες σε ένα βασικό σημείο. Οι πίνακες που συμμετέχουν στον πολλαπλασιασμό μετατρέπονται σε διανύσματα για όλες τις πράξεις που ακολουθούν [5]. Έτσι λοιπόν, ένας πίνακας διάστασης n, θα μετατραπεί στην αρχή της διαδικασίας σε διάνυσμα μεγέθους n^2 (βλ. σχήμα 6, (a)). Με λίγα λόγια, οι διάφορες ανακατατάξεις που προκύπτουν στο δεξί μέλος της ισότητας (1) είναι διανύσματα τα στοιχεία των

οποίων αλλάζουν θέση για κάθε i. Το άθροισμα των επιμέρους γινομένων των A_i και B_i έχει ως αποτέλεσμα ένα τελικό διάνυσμα μεγέθους n^2, το οποίο μπορεί να αποκρυπτογραφηθεί και να μετατραπεί στο οριστικό πίνακα-αποτέλεσμα. Η μετατροπή πίνακα σε διάνυσμα δίνει ένα σημαντικό πλεονέκτημα καθώς η πρόσθεση δυο διανυσμάτων στοιχείο προς στοιχείο είναι γρήγορη, οι αναδιατάξεις μπορούν να γίνουν με κόστος O(1) και επιπλέον ολόκληρος ο πίνακας μπορεί να χωρέσει σε ένα κρυπτοκείμενο με την τεχνική packing.



Σχήμα 7: Μέθοδος των Jiang et al. [5]

Οι εικόνες στο σχήμα 6 απεικονίζουν ένα παράδειγμα της διαδικασίας που μόλις περιγράφηκε. Έχοντας δυο πίνακες A και B (σχήμα 6, (b)), για να πάρουμε το γινόμενο τους αρκεί αρχικά να μετατραπεί ο κάθε πίνακας σε ένα διάνυσμα. Στη συνέχεια, παίρνουμε από τον κάθε πίνακα τις n διαφορετικές αντιμεταθέσεις και πολλαπλασιάζουμε αντίστοιχα μεταξύ τους. Προσθέτουμε τα n γινόμενα που προκύπτουν και έχουμε το τελικό αποτέλεσμα. Σημειώστε ότι ενώ στο σχήμα 6 (b) η απεικόνιση γίνεται σε μορφή πίνακα ώστε να διακρίνονται καλύτερα οι μεταθέσεις των στοιχείων, στην πραγματικότητα τα A_0, B_0, … A_2, B_2 είναι διανύσματα. Η ισότητα (1) μπορεί να εκφραστεί για αυτό το παράδειγμα ως εξής:

$$A \cdot B = A_0 \odot B_0 + A_1 \odot B_1 + A_2 \odot B_2$$

Κατασκευή A_0, B_0

Από τους A_0, B_0 κατασκευάζονται οι πίνακες για τις επόμενες επαναλήψεις, δηλαδή οι A_1, B_1 … A_{n-1}, B_{n-1}. Χρειαζόμαστε δυο πίνακες U_σ, U_τ με τους οποίους θα πολλαπλασιαστούν οι πίνακες A και B ώστε να παραχθούν οι A_0, B_0. Οι U_σ, U_τ είναι αραιοί πίνακες διάστασης n^2 τα στοιχεία των οποίων έχουν την τιμή 0 ή 1, όπου ο U_σ διαθέτει $(2n – 1)$ μη-μηδενικές διαγωνίους, και ο U_τ διαθέτει n μη-μηδενικές διαγωνίους. Αυτοί οι πίνακες λειτουργούν σαν "μάσκες" και πολλαπλασιάζοντας τον κάθε ένα με το αντίστοιχο πίνακα σε μορφή διανύσματος A ή B παίρνουμε ως αποτέλεσμα το διάνυσμα A_0 ή B_0. Οι τύποι όπως ορίζονται στο [5] για την κατασκευή τους είναι:

$U_\sigma[n \cdot i + j, k] = 1$ εάν $k = n \cdot i + [i + j]_n$, αλλιώς 0

$U_\tau[n \cdot i + j, k] = 1$ εάν $k = n \cdot [i + j]_n + j$, αλλιώς 0

Όπου $0 \leq i, j < n$ και $0 \leq k < n^2$. Επίσης με $[i + j]_n$ δηλώνουμε το modulo n του αθροίσματος (i+j).

Έχοντας τους πίνακες U_σ, U_τ πολλαπλασιάζονται με τα διανύσματα A και B αντίστοιχα, χρησιμοποιώντας την μέθοδο των διαγωνίων Halevi and Shoup [22].

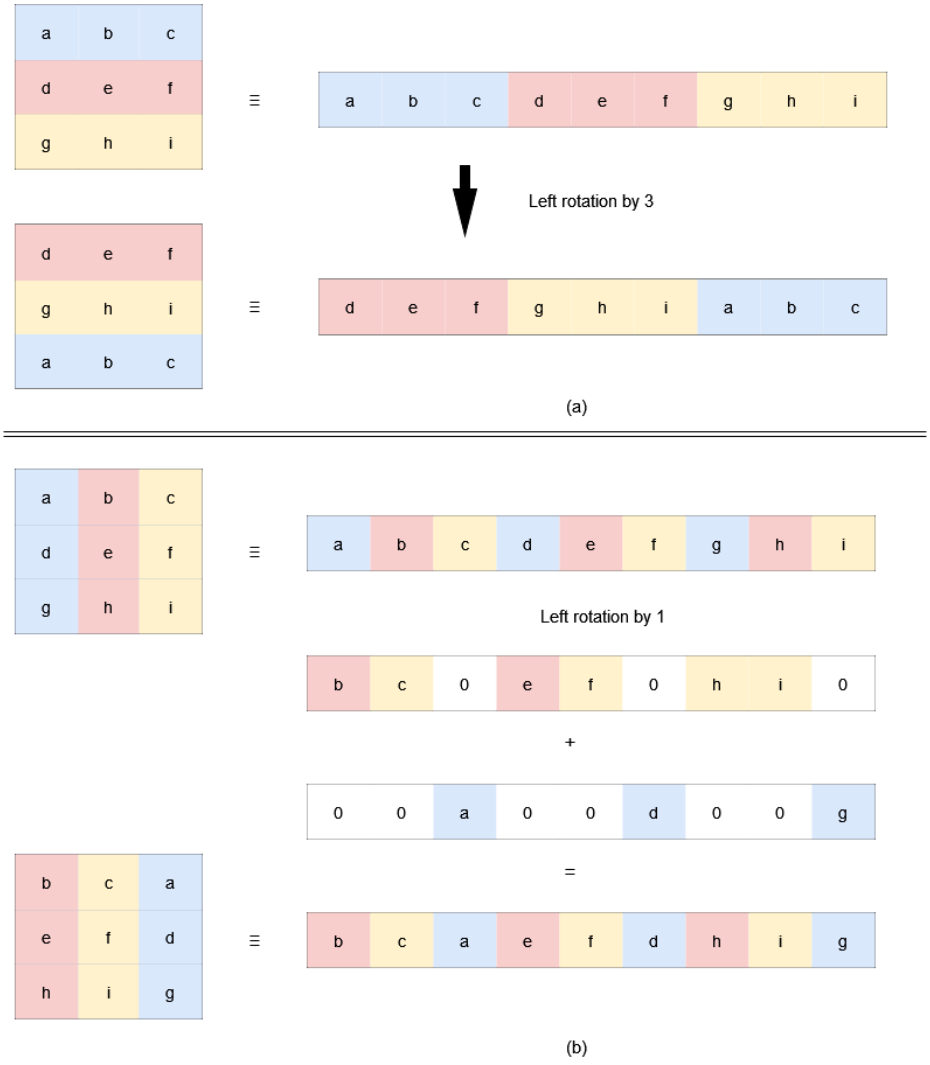
$U_\sigma \cdot A = A_0$

$U_\tau \cdot B = B_0$

Αντιμεταθέσεις

Οι αντιμεταθέσεις για τους επόμενους πολλαπλασιασμούς παράγονται χρησιμοποιώντας rotations ή συνδυασμό rotation με πολλαπλασιασμό SIMD διανυσμάτων. Πιο συγκεκριμένα, οι ανακατατάξεις του B_0 προκύπτουν ουσιαστικά από την αλλαγή της σειράς των γραμμών του, το οποίο σε μορφή διανύσματος μπορεί να επιτευχθεί περιστρέφοντας κατά n θέσεις αριστερά (left rotation). Στο παράδειγμα μας, το B_1 επομένως παράγεται από το B_0 με ευκολία, περιστρέφοντας το B_0 κατά 3 θέσεις αριστερά. Το αποτέλεσμα της περιστροφής φαίνεται στο σχήμα 6 (c) και παράδειγμα της περιστροφής στο σχήμα 7 (a). Οι ανακατατάξεις του A_0 προκύπτουν από την αλλαγή σειράς των στηλών του, δηλαδή μετακινώντας τις στήλες κατά μια θέση αριστερά σε σχέση με την προηγούμενη αντιμετάθεση. Αυτή η διαδικασία είναι λίγο περισσότερο περίπλοκη όμως συνεχίζει να μην επηρεάζει σημαντικά το κόστος. Σύμφωνα με το [5] το A_1 παράγεται από την πρόσθεση ενός διανύσματος-μάσκας με το A_0 περιστραμμένο κατά 1 θέση αριστερά και

μηδενίζοντας τις θέσεις του που αποτελούν πολλαπλάσια του n-1. Παράδειγμα δίνεται στο σχήμα 7 (b) παρακάτω.



Σχήμα 8: Ανακατάταξη γραμμών πίνακα (a), ανακατάταξη στηλών πίνακα (b)

Οι πράξεις μεταξύ των αντιμεταθέσεων των πινάκων, δηλαδή ο πολλαπλασιασμός και η πρόσθεση, γίνεται με SIMD. Το στοιχείο στη θέση i του A_0 πολλαπλασιάζεται με το στοιχείο στη θέση i του B_0, όπου $0 <= i < n-1$. Η κάθε πράξη εκτελείται ταυτόχρονα και στις n θέσεις, καθώς χρησιμοποιούμε packing.

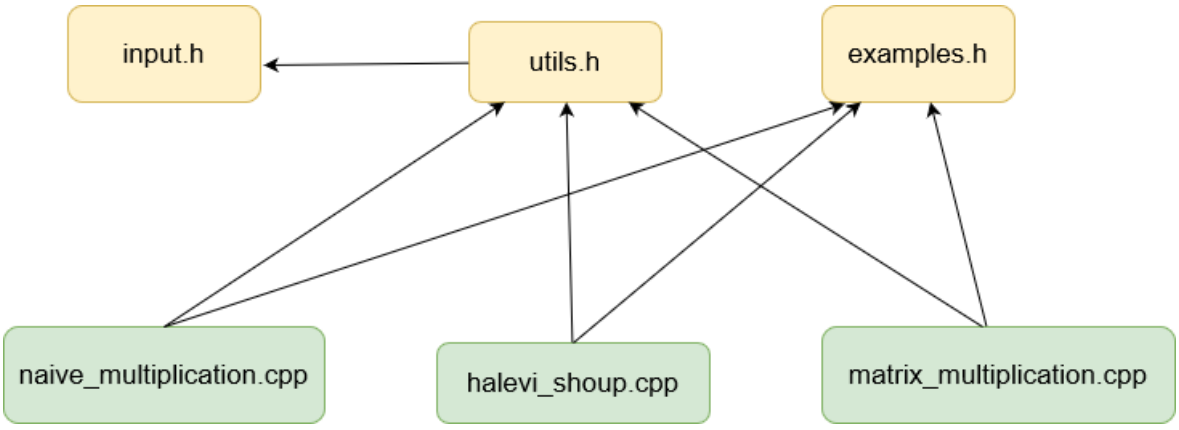
Μέθοδος	***Αριθμός απαιτ. κρυπτοκειμένων***	***Πολυπλοκότητα***	***Απαιτούμενο βάθος***
Naive	d^2	$O(d^3)$	1 πολλ/μός
Halevi and Shoup [22]	d	$O(d^2)$	1 πολλ/μός
Jiang, Kim, Lauter and Song [5]	1	O(d)	1 πολλ/μός + 1 πολλ/μός με σταθερές

Πίνακας 1: Σύνοψη απόδοσης των τριών προσεγγίσεων όπου d είναι η διάσταση του τετραγωνικού πίνακα [5]

4 *Υλοποίηση*

4.1 Δομή αρχείων

Ο κώδικας της εργασίας βρίσκεται στο repository https://bitbucket.org/nireaskalcas/projectseal/src/main/ όπου στο branch "main" διατηρείται πάντα η πιο πρόσφατη έκδοση. Για την υλοποίηση επιλέξαμε να χρησιμοποιήσουμε την γλώσσα προγραμματισμού C++ και την έκδοση 3.7.0 του Microsoft SEAL (https://github.com/microsoft/SEAL/tree/3.7.0).



Σχήμα 9: Εξαρτήσεις αρχείων κώδικα

Ο κώδικας είναι κατανεμημένος στα αρχεία του σχήματος 9, όπου οι ακμές δηλώνουν την εξάρτηση μεταξύ τους. Κάθε ακμή έχει κατεύθυνση προς το αρχείο το οποίο γίνεται include ως βιβλιοθήκη στο αρχείο από το οποίο ξεκινάει η ακμή. Ακολουθεί η λίστα των αρχείων και μια σύντομη περιγραφή για τον σκοπό και περιεχόμενο του κάθε αρχείου.

- **input.h**: Περιέχει παραμέτρους και μεταβλητές που ελέγχουν το μέγεθος των τετραγωνικών πινάκων N, την αυτόματη δημιουργία περιεχομένου πινάκων, επιλογές εμφάνισης στην οθόνη μηνυμάτων αποσφαλμάτωσης (debugging) και εμφάνιση στην οθόνη αρχικών και ενδιάμεσων πινάκων. Πρόκειται για το μόνο αρχείο που μπορεί να αλλάξει ο χρήστης πριν να τρέξει κάποια από τις μεθόδους, και οι αλλαγές που αποθηκεύονται έχουν εφαρμογή και στις τρεις μεθόδους.
- **utils.h**: Περιέχει διάφορες βοηθητικές μεθόδους για καλύτερη οργάνωση και επαναχρησιμοποίηση κώδικα.
- **examples.h**: Αρχείο που ανήκει στην βιβλιοθήκη Microsoft SEAL και περιέχει βοηθητικές μεθόδους που αφορούν το SEAL.
- **naive_multiplication.cpp**: Η υλοποίηση της προσέγγισης πολλαπλασιασμού πίνακα με διάνυσμα με προσέγγιση naive (ενότητα 3.2.1).
- **halevi_shoup.cpp**: Η υλοποίηση της προσέγγισης πολλαπλασιασμού πίνακα με διάνυσμα με προσέγγιση Halevi-Shoup (ενότητα 3.2.2).
- **matrix_multiplication.cpp**: Η υλοποίηση της προσέγγισης πολλαπλασιασμού πίνακα με πίνακα με προσέγγιση JKLS (ενότητα 3.2.3).

4.2 Εκτέλεση

Προαπαιτούμενα για την εκτέλεση του κώδικα αποτελούν η εγκατάσταση της γλώσσας C++, ενός συμβατού compiler, η εγκατάσταση της βιβλιοθήκης Microsoft SEAL. Για την περίπτωση περιβάλλοντος Linux (Ubuntu) οδηγίες για τα παραπάνω παραθέτονται στο Παράρτημα ΙΙ, ενώ για διαφορετικά Linux συστήματα ίσως να χρειαστούν μικρές αλλαγές στις εντολές. Επίσης να σημειωθεί ότι για διαφορετικές εκδόσεις του Microsoft SEAL ενδεχομένως να υπάρχουν διαφορετικές οδηγίες εγκατάστασης αλλά και θέματα συμβατότητας με τον υπάρχοντα κώδικα.

Για την εκτέλεση οποιασδήποτε από τις τρεις προσεγγίσεις, πρέπει να ακολουθηθούν τα παρακάτω βήματα.

1) Πρέπει να γίνει clone ο κώδικας της εργασίας από το bitbucket repository (https://bitbucket.org/nireaskalcas/projectseal/src/main/) και ο χρήστης να βεβαιωθεί ότι βρίσκεται στο branch “main” τρέχοντας την εντολή “git branch”.
2) Ανοίγοντας ένα terminal στο root directory του repository (/projectseal) εκτελείται η εντολή “cmake .”
3) Στη συνέχεια εκτελείται η εντολή “make”.
4) Εάν οι προηγούμενες εντολές εκτελέστηκαν χωρίς σφάλματα, ο χρήστης μπορεί να τρέξει ένα από τα τρία δυνατά εκτελέσιμα αρχεία με μια από τις εντολές “./**naive_multiplication”, “./halevi_shoup”, “./matrix_multiplication”.** Μόνο μια προσέγγιση μπορεί να εκτελεστεί την φορά, χωρίς όριο στο πόσες φορές εκτελείται. Όλες

οι προσεγγίσεις χρησιμοποιούν τις ίδιες παραμέτρους εισόδου που δηλώνονται στο αρχείο input.h.

Με κάθε αλλαγή που γίνεται στο αρχείο input.h (π.χ. αλλαγή του N από 3 σε 12) και αποθήκευση της αλλαγής, πρέπει να εκτελεστούν τα βήματα 3 και 4 ώστε να γίνει αντιληπτή η αλλαγή από την εκάστοτε υλοποίηση.

4.3 Υλοποίηση των προσεγγίσεων

Σε κάθε μια από τις υλοποιήσεις πριν από κάθε αρχικοποίηση plaintext/ciphertext χρειάζεται ο ορισμός κάποιων παραμέτρων της βιβλιοθήκης SEAL. Σε όλες τις μεθόδους χρησιμοποιούμε το σχήμα BFV οπότε συγκεκριμένα για αυτό το σχήμα οι παράμετροι που πρέπει να οριστούν, σύμφωνα με τις επίσημες οδηγίες του SEAL [61] είναι:

- **poly_modulus_degree**: Είναι πάντα δύναμη του 2 και αντιπροσωπεύει τον βαθμό του κυκλοτομικού πολυωνύμου. Όσο μεγαλύτερη είναι αυτή η τιμή, τόσο μεγαλύτερα είναι τα μεγέθη των ciphertext, και τόσο περισσότερο χρόνο παίρνουν οι πράξεις. Όμως, μεγαλύτερη τιμή poly_modulus_degree μας δίνει την δυνατότητα για πιο περίπλοκες πράξεις.
- **coeff_modulus**: Ένας πολύ μεγάλος ακέραιος, ο οποίος είναι το γινόμενο διακριτών πρώτων αριθμών όπου ο καθένας από αυτούς είναι μεγέθους 60 bits ή μικρότερος. Όσο μεγαλύτερη τιμή έχει, τόσο μεγαλύτερο noise budget έχουμε στα κρυπτοκείμενα και επομένως περισσότερες δυνατότητες υπολογισμών. Η μέγιστη τιμή που μπορεί να πάρει καθορίζεται από το poly_modulus_degree.
- **plain_modulus**: Το plain_modulus μπορεί να είναι οποιοσδήποτε θετικός αριθμός και δηλώνει το μέγεθος του τύπου δεδομένων του plaintext καθώς και την κατανάλωση του noise budget σε πολλαπλασιασμούς. Είναι λοιπόν καλή πρακτική να διατηρείται όσο μικρότερο γίνεται, για μικρότερη κατανάλωση.

Ο πιο σημαντικός παράγοντας στην επιλογή των κατάλληλων παραμέτρων κρυπτογράφησης είναι το πολλαπλασιαστικό βάθος του αριθμητικού κυκλώματος που ο χρήστης θέλει να αξιολογήσει σε κρυπτογραφημένα δεδομένα. Αν το noise budget ενός κρυπτογραφημένου κειμένου φτάσει στο μηδέν, γίνεται αδύνατο να αποκρυπτογραφηθεί. Επομένως, είναι απαραίτητο να επιλέξετε τις παραμέτρους αρκετά μεγάλες ώστε να υποστηρίζεται ο υπολογισμός στην κάθε περίπτωση. Ακολουθεί τμήμα κώδικα όπου ορίζονται οι παράμετροι.

```
EncryptionParameters parms(scheme_type::bfv);
size_t poly_modulus_degree = 4096;
```

```
parms.set_poly_modulus_degree(poly_modulus_degree);
parms.set_coeff_modulus(CoeffModulus::BFVDefault(poly_modulus_degree));
parms.set_plain_modulus(1024);
```

4.3.1 Υλοποίηση Naive

Η υλοποίηση της απλής προσέγγισης δεν έχει σημεία ιδιαίτερου ενδιαφέροντος, καθώς ακολουθήθηκε ο πιο εύκολος προγραμματιστικά (και λογικά) τρόπος, με κόστος την απόδοση. Η διαδικασία περιγράφεται σε θεωρητικό επίπεδο στο κεφάλαιο 3.2.1. Στην υλοποίηση έχουμε ως είσοδο ένα διάνυσμα N στοιχείων και έναν πίνακα διάστασης N γραμμών και N στηλών. Κάθε γραμμή του πίνακα απομονώνεται, και πολλαπλασιάζεται με το διάνυσμα εισόδου, στοιχείο προς στοιχείο, διατηρώντας το άθροισμα των επιμέρους γινομένων. Στο τέλος έχουμε N αθροίσματα, τα οποία και αποτελούν το τελικό διάνυσμα αποτέλεσμα του πολλαπλασιασμού. Κάθε ακέραιος κρυπτογραφείται σε ξεχωριστό ciphertext επειδή δεν χρησιμοποιούμε batching. Στο παράρτημα Ι παραθέτουμε την συνάρτηση **multiply_add_ciphertexts()** η οποία πολλαπλασιάζει δυο ciphertexts-εισόδους και τα προσθέτει μεταξύ τους, προσθέτοντας επίσης και το προηγούμενο άθροισμα.

4.3.2 Υλοποίηση Halevi-Shoup

Για την προσέγγιση Halevi-Shoup χρειάζεται μια προεργασία του πίνακα πριν ξεκινήσουν οι πράξεις. Καλείται η βοηθητική συνάρτηση **createDiagonalsMatrix()** η οποία παίρνει ως είσοδο τον αρχικό πίνακα, αποσπά από αυτόν τις διαγωνίους, και τις αποθηκεύει ως γραμμές σε έναν δεύτερο πίνακα **diagonalsMatrix** ο οποίος επίσης δίνεται ως είσοδος χωρίς να έχει αρχικοποιηθεί. Στην συνέχεια, ο πίνακας **diagonalsMatrix** δίνεται ως είσοδος στην συνάρτηση **encrypt_matrix_halevi_shoup().** Κάθε γραμμή του **diagonalsMatrix** κρυπτογραφείται σε διαφορετικό ciphertext, και ένα μόνο ciphertext χρησιμοποιείται και για την κρυπτογράφηση ολόκληρου του διανύσματος εισόδου, με το οποίο πρέπει να πολλαπλασιαστεί ο πίνακας. Είναι εφικτό να χρησιμοποιήσουμε μόνο ένα ciphertext ανά γραμμή πίνακα / διάνυσμα επειδή γίνεται η χρήση του batching mode του SEAL. Η πρώτη γραμμή του **diagonalsMatrix**, σε μορφή ciphertext, πολλαπλασιάζεται με το ciphertext του διανύσματος. Στη συνέχεια κάθε επόμενη γραμμή του **diagonalsMatrix** πολλαπλασιάζεται με ένα rotation του ciphertext διανύσματος. Μετά από κάθε πολλαπλασιασμό, είναι υποχρεωτική η χρήση του **relinearize_inplace()** με είσοδο το ciphertext και ένα ειδικό σετ κλειδιών relin_keys ώστε να μειωθεί ο θόρυβος του ciphertext σε ικανοποιητικό επίπεδο. Ειδικά για μεγάλα N υπάρχει κίνδυνος το noise budget να γίνει 0 μετά από N διαδοχικούς πολλαπλασιασμούς χωρίς την χρήση του. Η βιβλιοθήκη παρέχει την μέθοδο **rotate_rows_inplace()** για την περιστροφή των στοιχείων του ciphertext κατά k θέσεις αριστερά, στην περίπτωση μας k=1. Μπορεί να γίνει και δεξιά περιστροφή εάν δώσουμε σαν παράμετρο αρνητικό ακέραιο. Στο τέλος όλα τα γινόμενα αθροίζονται, και αποκρυπτογραφούνται ώστε να πάρουμε το τελικό διάνυσμα. Η συνάρτηση **encrypt_matrix_halevi_shoup()** βρίσκεται στο Παράρτημα Ι.

4.3.3 Υλοποίηση JKLS

Η τελευταία υλοποίηση χρειάζεται την μεγαλύτερη προετοιμασία, καθώς πρέπει να δημιουργηθούν οι πίνακες **u_sigma**, **u_tau** καλώντας τις βοηθητικές μεθόδους **construct_u_sigma()** και **construct_u_tau()**, με τον κάθε πίνακα να είναι μεγέθους N^2 επί N^2. Στην συνέχεια οι πίνακες A και B μετατρέπονται σε μορφή διανύσματος **a_vector** και **b_vector**. Κατασκευάζονται οι πίνακες διαγωνίων των **u_sigma**, **u_tau** (**u_sigma_diagonals**, **u_tau_diagonals**) με τον ίδιο τρόπο όπως και στην Halevi-Shoup (καλώντας την createDiagonalsMatrix). Χρησιμοποιώντας ως εισόδους τους πίνακες **u_sigma_diagonals** και **a_vector**, δημιουργούμε το ciphertext **a0** που αντιστοιχεί στον πίνακα A_0 . Με ίδιο τρόπο χρησιμοποιώντας τους πίνακες **u_tau_diagonals** και **b_vector** δημιουργείται και το ciphertext **b0** που αντιστοιχεί στον πίνακα B_0. Εκτελούνται N επαναλήψεις όπου για κάθε 0<i<N τα ciphertexts ai και bi πολλαπλασιάζονται μεταξύ τους, το αποτέλεσμα προστίθεται στο προηγούμενο άθροισμα, και γίνονται οι κατάλληλες μετατροπές όπως περιγράφονται στο κεφάλαιο 3.2.3, με την βοήθεια του **rotate_rows_inplace()**.

5 *Σύγκριση υλοποιήσεων*

5.1 Περιβάλλον και μεθοδολογία

Σε αυτό το κεφάλαιο θα παρουσιάσουμε μετρήσεις που έγιναν χρησιμοποιώντας τις υλοποιήσεις των μεθόδων του κεφαλαίου 3.2. Σκοπός του κεφαλαίου είναι η σύγκριση της απόδοσης σε χρόνο και χώρο ανάμεσα στις τρεις διαφορετικές υλοποιήσεις. Αποφασίσαμε να δώσουμε ως είσοδο στην κάθε μέθοδο δύο τετραγωνικούς πίνακες συγκεκριμένων διαστάσεων και για κάθε διάσταση να μετρηθεί ο χρόνος εκτέλεσης πράξεων σε κρυπτογραφημένα δεδομένα, ο συνολικός χρόνος από την αρχή εκτέλεσης του προγράμματος ως το τέλος, και ο χώρος που χρειάζονται τα δεδομένα. Ο χρόνος εκτέλεσης πράξεων περιλαμβάνει πράξεις μόνο σε κρυπτογραφημένα δεδομένα (πολλαπλασιασμός, πρόσθεση) και καμία άλλη τυχόν προεργασία. Καθώς οι μέθοδοι Naive και Halevi & Shoup είναι μέθοδοι πολλαπλασιασμού πίνακα με διάνυσμα, οι υλοποιήσεις μας αυτόματα πολλαπλασιάζουν τον χρόνο που μετρήθηκε επί N, με σκοπό να προσεγγίσουμε τον πραγματικό χρόνο πολλαπλασιασμού με κάποιον πίνακα αντί για διάνυσμα (θεωρώντας ότι το διάνυσμα θα μπορούσε να αποτελεί μία από τις N στήλες του δεύτερου πίνακα). Επίσης αφαιρείται από αυτόν τον χρόνο και ο χρόνος κρυπτογράφησης/αποκρυπτογράφησης. Τα δεδομένα που συλλέξαμε είναι από την εκτέλεση κάθε μεθόδου, απο 10 φορές για κάθε N, και κρατώντας τον μέσο όρο, ώστε να έχουμε ένα πιο σίγουρο αποτέλεσμα και να αποφύγουμε τυχαίες αυξομειώσεις στους χρόνους. Όσον αφορά τις απαιτήσεις σε χώρο, μετρήθηκε μόνο ο ένας από τους δυο πίνακες για λόγους απλότητας. Ωστόσο είναι εύκολο να βρεθούν οι αντίστοιχες απαιτήσεις σε χώρο και του πίνακα B, καθώς έχει ακριβώς το ίδιο μέγεθος με τον A, και επομένως τα δεδομένα δεν αποκλίνουν. Τα αποτελέσματα παραθέτονται παρακάτω σε μορφή πινάκων, και η μικρότερη τιμή για κάθε μέτρηση χρωματίζεται με ανοιχτό πράσινο. Οι μονάδες μέτρησης που επιλέξαμε είναι microsecond (μs) και second (s) για τον χρόνο, καθώς και kilobyte (kB) για τον χώρο. Θυμίζουμε ότι οι μέθοδοι που εκτελέσαμε είναι οι παρακάτω:

- Naive (ενότητα 3.2.1)
- Halevi & Shoup (ενότητα 3.2.2)
- Jiang, Kim, Lauter, Song (JKLS) (ενότητα 3.2.3)

Τα μεγέθη που θα συγκρίνουμε είναι N = [3, 12, 32, 64] , όπου N η διάσταση των πινάκων που πολλαπλασιάζονται (N γραμμές επί N στήλες). Ως περιβάλλον χρησιμοποιήσαμε το Oracle VM Virtualbox σε Windows 10 host μηχάνημα, για να εγκαταστήσουμε το λειτουργικό σύστημα μας καθώς και τα υπόλοιπα εργαλεία που χρειάστηκαν κατά την υλοποίηση και εκτέλεση. Τα χαρακτηριστικά του host μηχανήματος στο οποίο εγκαταστάθηκε το VM είναι τα εξής:

- Intel Core i7-8700k 3.70 Ghz
- Ram 16 GB
- Windows 10 Pro 21H2 version

Το ίδιο το VM έχει λειτουργικό σύστημα Ubuntu 20.04.3 και του έχουν δωθεί 8 GB RAM. Αφού εγκαταστάθηκε το λογισμικό του VM, εγκαταστήσαμε το cmake, git και το clang. Στην συνέχεια έγινε εγκατάσταση του Microsoft SEAL.

5.2 Μέτρηση χρόνου

	JKLS	**Halevi-Shoup**	**Naive**
N=3	1368121.5 μs / 1.4 s	2407202.7 μs / 2.4 s	730236.6 μs / 0.7 s
N=12	9928636.9 μs / 9.9 s	15025596 μs / 15.0 s	16063513.2 μs / 16 s
N=32	74945147.6 μs / 74.9 s	79342044.8 μs / 79.3 s	256963238.4 μs / 256.9 s
N=64	1594594278 μs / 1594.6 s	297090022.4 μs / 297.1 s	2067391155 μs / 2067.4 s

Πίνακας 2: Συνολικός χρόνος εκτέλεσης των μεθόδων

	JKLS	**Halevi-Shoup**	**Naive**
N=3	230045.3 μs / 0.2 s	450457.5 μs / 0.5 s	353193 μs / 0.3 s
N=12	1115118.3 μs / 1.1 s	5772306.4 μs / 5.8 s	9659437.2 μs / 9.6 s

N=32	2802442.6 μs / 2.8 s	40226284.8 μs / 40.2 s	159199209.6 μs / 159.2 s
N=64	26439695 μs / 26.4 s	163997484.8 μs / 164 s	1297161862 μs / 1297.16 s

Πίνακας 3: Χρόνος εκτέλεσης χωρίς encryption/decryption

5.3 Μέτρηση χώρου

Στην συνέχεια για τα ίδια μεγέθη του N υπολογίσαμε και τον χώρο όπως φαίνεται στους ακόλουθους πίνακες.

	JKLS	**Halevi-Shoup**	**Naive**
Batching	Ναι	Ναι	Όχι
Matrix A before encoding	0,036 kB	0,036 kB	0,036 kB
Matrix A single plaintext	327 kB	327 kB	98 kB
Matrix A single ciphertext	655 kB	655 kB	196 kB
Size of total matrix A ciphertexts	655 kB	1965 kB	1764 kB

Πίνακας 4: Μέτρηση χώρου για N = 3

	JKLS	**Halevi-Shoup**	**Naive**
Batching	Ναι	Ναι	Όχι
Matrix A before encoding	0,576 kB	0,576 kB	0,576 kB
Matrix A single plaintext	327 kB	327 kB	98 kB
Matrix A single ciphertext	655 kB	655 kB	196 kB
Size of total matrix A ciphertexts	655 kB	7860 kB	28224 kB

Πίνακας 5: Μέτρηση χώρου για N = 12

	JKLS	**Halevi-Shoup**	**Naive**
Batching	Ναι	Ναι	Όχι
Matrix A before encoding	4096 kB	4096 kB	4096 kB
Matrix A single plaintext	327 kB	327 kB	98 kB
Matrix A single ciphertext	655 kB	655 kB	196 kB
Size of total matrix A ciphertexts	655 kB	20960 kB	200704 kB

Πίνακας 6: Μέτρηση χώρου για N = 32

	JKLS	**Halevi-Shoup**	**Naive**
Matrix A before encoding	16384 kB	16384 kB	16384 kB
Matrix A single plaintext	1179 kB	327 kB	98 kB
Matrix A single ciphertext	2359 kB	655 kB	196 kB
Size of total matrix A ciphertexts	2359 kB	41920 kB	802816 kB

Πίνακας 7: Μέτρηση χώρου για N = 64

5.4 Ανάλυση μετρήσεων

Ο πίνακας 2 απεικονίζει τις μετρήσεις του συνολικού χρόνου εκτέλεσης των μεθόδων, Παρατηρούμε ότι για N=3 η μέθοδος Naive είναι η πιο γρήγορη. Για N=12, 32 η μέθοδος JKLS είναι η γρηγορότερη και για N=64 η μέθοδος Halevi-Shoup. Επειδή σε αυτούς τους χρόνους συμπεριλαμβάνονται και οι πράξεις προετοιμασίας του ciphertext που θα μπορούσε κάποιος θεωρητικά να στείλει σε έναν server (π.χ. η κατασκευή των πινάκων U_σ, U_τ στην JKLS) τα αποτελέσματα δεν τείνουν προς μια συγκεκριμένη μέθοδο ως καλύτερη. Για N=3 η μηδενική προετοιμασία που χρειάζεται η μέθοδος Naive, της δίνει το πλεονέκτημα, και για τον ίδιο λόγο για N=64 η μέθοδος JKLS παίρνει περισσότερο χρόνο ώστε να δημιουργήσει τους βοηθητικούς πίνακες.

Τα αποτελέσματα που μας ενδιαφέρουν περισσότερο όσον αφορά τον χρόνο εκτέλεσης βρίσκονται στον πίνακα 3. Εδώ μετράμε μόνο τον χρόνο εκτέλεσης πράξεων πρόσθεσης και πολλαπλασιασμού μεταξύ των ciphertext. Όπως είναι αναμενόμενο, η JKLS είναι η γρηγορότερη μέθοδος από τις τρεις, ακολουθεί η Halevi-Shoup και τέλος η πιο αργή μέθοδος είναι η Naive. Καθοριστικό παράγοντα αποτελεί ο αριθμός των πολλαπλασιασμών που χρειάζεται σε κάθε μέθοδο, επειδή είναι η πιο χρονοβόρα πράξη στην

ομομορφική κρυπτογραφία, όπως είδαμε στο κεφάλαιο 3. Η μέθοδος JKLS χρειάζεται N πολλαπλασιασμούς, η Halevi-Shoup N^2 πολλαπλασιασμού και η Naive N^3 πολλαπλασιασμούς. Τα αποτελέσματα μας συμφωνούν με τον πίνακα 1 [5] του κεφαλαίου 3.2.3.

Στις μετρήσεις του χώρου, για όλες τις διαφορετικές τιμές N, παρατηρούμε ότι το μικρότερο μέγεθος μόνο ενός plaintext / ciphertext διαθέτει η μέθοδος Naive. Είναι η μόνη που δεν χρησιμοποιεί batching, οπότε το μέγεθος ενός plaintext ή ciphertext που χωράει έναν μοναδικό ακέραιο αριθμό είναι μικρότερο σε σχέση με τις άλλες δυο μεθόδους που χρησιμοποιούν batching. Όμως, για να μετατραπεί να κρυπτογραφημένη μορφή ολόκληρος ο πίνακας χρειάζεται μια συλλογή από N^2 ciphertext (π.χ. ένα array) οπότε το άθροισμα των επιμέρους μεγεθών είναι αρκετά μεγάλο. Σε αυτήν την περίπτωση η JKLS είναι και πάλι η καλύτερη μέθοδος καθώς χρησιμοποιεί μόνο ένα ciphertext ανά πίνακα, το οποίο ξεκινάει από την μορφή του βοηθητικού πίνακα A_0 και στη συνέχεια γίνονται μετατροπές πάνω στο ίδιο ciphertext χωρίς την ανάγκη για επιπλέον χώρο.

Βιβλιογραφία

[1] R. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms.", Foundations of Secure Computation, pp. 160-171, 1978.

[2] E. Brickell and Y. Yacobi, "On privacy homomorphisms", in: "Advances in Cryptology – Eurocrypt", Springer, Berlin, Heidelberg, D. Chaum and W. L. Price, Eds., 1988, pp. 117-125.

[3] C. Gentry, "A fully homomorphic encryption scheme.", Ph.D. dissertation, Dept. of Computer Science, Stanford University, 2009.

[4] R. Shrestha and S. Kim, "Integration of IoT with blockchain and homomorphic encryption: Challenging issues and opportunities", in: "Advanced in Computers", vol. 115, Elsevier, ch. 10, 2019, pp. 302-304.

[5] X. Jiang, M. Kim, K. Lauter, and Y. Song, "Secure Outsourced Matrix Computation and Application to Neural Networks", University of Texas, Microsoft Research, 2019.

[6] Z. Brakerski, C. Gentry and V. Vaikuntanathan, "(Leveled) Fully Homomorphic Encryption without Bootstrapping", Stanford University, IBM Research, University of Toronto, 2013.

[7] L. Morris, "Analysis of Partially and Fully Homomorphic Encryption", Rochester Institute of Technology, 2013.

[8] Z. Brakerski, C. Gentry and V. Vaikuntanathan, "Fully Homomorphic Encryption without Bootstrapping", Weizmann Institute of Science, IBM T.J. Watson Research Center, University of Toronto, 2011.

[9] A. L´opez-Alt, E. Tromer, V. Vaikuntanathan, "On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption", New York University, Tel Aviv University, MIT, 2013.

[10] J. Fan, F. Vercauteren, "Somewhat Practical Fully Homomorphic Encryption", Katholieke Universiteit Leuven, COSIC & IBBT, 2012.

[11] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme", Microsoft Research, University of Bristol, 2013.

[12] Brown, William C. "Matrices and vector spaces" , University of Mannheim, Department of Telematics, NTNU, Department of Mathematical Sciences, NTNU, 2015.

[13] Brown, William C., "Matrices and vector spaces", New York, NY: Marcel Dekker, ISBN 978-0-8247-8419-5m, 1991.

[14] I. Iliashenko and V. Zucca, "Faster homomorphic comparison operations for BGV and BFV", Proceedings on Privacy Enhancing Technologies, De Gruyter Open, pp.246-264, 10.2478/popets-2021-0046. hal-03506798, 2021.

[15] L. Ducas and D. Micciancio, "FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second", Elisabeth Oswald and Marc Fischlin, Eds., Advances in Cryptology - EUROCRYPT 2015, pp. 617-640, Berlin, Heidelberg, Springer Berlin Heidelberg, 2015.

[16] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene, "Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds", Jung Hee Cheon and Tsuyoshi Takagi, Eds., Advances in Cryptology – ASI - ACRYPT 2016, pp. 3-33, Berlin, Heidelberg, Springer Berlin Heidelberg, 2016.

[17] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical GapSVP", Reihaneh Safavi-Naini and Ran Canetti, Eds., CRYPTO 2012, volume 7417 of LNCS, pp.868-886, Springer, Heidelberg, 2012.

[18] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers", Tsuyoshi Takagi and Thomas Peyrin, Eds., Advances in Cryptology - ASIACRYPT 2017, pp. 409-437, Cham, Springer International Publishing, 2017.

[19] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proc. Natl. Acad. Sci. U.S.A., pp. 2554–2558, 1982.

[20] L. Hardesty, "Explained: Neural networks", MIT News Office, 2017.

[21] W.S. McCulloch and W. Pitts, "A Logical Calculus of Ideas Immanent in nervous activity",. Bull. Mathematical Biophysics, Vol.5, pp.115-133, 1943.

[22] S. Halevi and V. Shoup, "Algorithms in HElib", Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, Proceedings, Part I, pp. 554 – 571, 2014.

[23] C. Juvekar, V. Vaikuntanathan, A. Chandrakasan, "GAZELLE: A Low Latency Framework for Secure Neural Network Inference", MIT, 2018.

[24] A. Zell, "Simulation Neuronaler Netze" [Simulation of Neural Networks] (in German), 1st ed., Addison-Wesley, 1994.

[25] J. Schmidhuber, "Deep learning in neural networks: An overview". Neural Networks, 61: 85–117. arXiv:1404.7828, 2015.

[26] M. Minsky and S. Papert, "Perceptrons: an introduction to computational geometry", 1969.

[27] P. Auer, H. Burgsteiner, W. Maas. "A learning rule for very simple universal approximators consisting of a single layer of perceptrons", Neural networks : the official journal of the International Neural Network Society, pp. 786-95, 2008.

[28] I. Goodfellow, Y. Bengio, A. Courville, "6.5 Back-Propagation and Other Differentiation Algorithms", Deep Learning, MIT Press, 2016.

[29] P. Ramachandran, B. Zoph, Q. V. Le, "Searching for Activation Functions". arXiv:1710.05941 [cs.NE], 2017.

[30] M. M. Noel, A. Trivedi, P. Dutta, "Growing Cosine Unit: A Novel Oscillatory Activation Function That Can Speedup Training and Reduce Parameters in Convolutional Neural Networks", arXiv preprint arXiv:2108.12943, 2021.

[31] M. M. Noel, S. Bharadwaj, V. Muthiah-Nakarajan, P. Dutta, G. B. Amali, "Biologically Inspired Oscillating Activation Functions Can Bridge the Performance Gap between Biological and Artificial Neurons", arXiv preprint arXiv:2111.04020, 2021.

[32] E. Million, "The Hadamard Product", April 12, 2007.

[33] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges", Brief. Bioinform., 2017.

[34] S. Halevi and V. Shoup, "Algorithms in HElib", Cryptology ePrint Archive, 2014.

[35] S. Halevi and V. Shoup, "Bootstrapping for HElib", Cryptology ePrint Archive, 2020.

[36] S. Halevi and V. Shoup, "HElib design principles", ", Cryptology ePrint Archive, 2020.

[37] PALISADE Homomorphic Encryption Software Library, https://palisade-crypto.org/, accessed 13-08-2022.

[38] PALISADE, https://gitlab.com/palisade, accessed 14-08-2022.

[39] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, J. Wernsing, "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy". Proceedings of the 33rd International Conference on Machine Learning, 2018.

[40] The Microsoft Simple Encrypted Arithmetic Library goes open source. Microsoft Research, 2018, Retrieved 2022-08-20.

[41] M. Y. Hong, J. S. Yoo, J. W. Yoon, "Homomorphic Model Selection for Data Analysis in an Encrypted Domain", MDPI Open Access Journals, 2020.

[42] M. Fellows and N. Koblitz, "Combinatorial cryptosystems galore! In Contemporary Mathematics", volume 168 of Finite Fields: Theory, Applications, and Algorithms, FQ2, pp 51–61, 1993.

[43] C. Gentry, A. Sahai, and B. Waters, "Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based", Advances in Cryptology – CRYPTO 2013, pp 75–92, Annual Cryptology Conference, 2013.

[44] Z. Brakerski and V. Vaikuntanathan, " Lattice-Based FHE as Secure as PKE", ITCS '14: Proceedings of the 5th conference on Innovations in theoretical computer science pp. 1–122013, 2014.

[45] J. Alperin-Sheriff and C. Peikert, "Faster Bootstrapping with Polynomial Error", Advances in Cryptology – CRYPTO 2014, pp 297–314 (revised), Annual Cryptology Conference, 2014.

[46] L. Ducas and D. Micciancio, "FHEW: A Fully Homomorphic Encryption library", GitHub, https://github.com/lducas/FHEW, Retrieved 21 August 2022.

[47] I. Chillotti, N. Gama, M. Georgieva, M. Izabachene, "Faster Fully Homomorphic Encryption: Bootstrapping in less than 0.1 Seconds, ASIACRYPT 2016, 2016.

[48] N. Gama, M. Izabachène, P.Q. Nguyen, and X. Xie, "Structural Lattice Reduction: Generalized Worst-Case to Average-Case Reductions and Homomorphic Cryptosystems", Advances in Cryptology – EUROCRYPT 2016, Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 528–558, 2016.

[49] C. Joshi, U. K. Singh, "Information security risks management framework – A step towards mitigating security risks in university network", Journal of Information Security and Applications, 2017.

[50] E. Volna, M. Kotyrba, V. Kocian, M. Janosek, "Cryptography based on neural network", Department of Informatics and Computers, University of Ostrava, 2012.

[51] L. R. Rivest, "Cryptography". In J. Van Leeuwen Handbook of Theoretical Computer Science, Vol. 1, Elsevier, 1990.

[52] W. Diffie, M. Hellman, "New directions in cryptography", IEEE Transactions on Information Theory, 2006.

[53] F. Cohen, "2.4 - Applications of Cryptography", 1995.

[54] R. Canetti, U. Feige, O. Goldreich, M. Naor, "Adaptively Secure Multi-party", STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing, pp. 639–648, 1996.

[55] A. Shamir, R. Rivest, and L. Adleman, "Mental Poker", Technical Report LCS/TR-125, Massachusetts Institute of Technology, 1979.

[56] A. C. Yao, “Protocols for secure computations”, University of California Berkeley, 1982.

[57] O. Goldreich, S. Micali, A. Wigderson, “How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority”, Providing Sound Foundations, 2019.

[58] M. Backes, B. Pfitzmann, M. Waidner. "A general composition theorem for secure reactive systems”, TCC 2004: Theory of Cryptography pp. 336–35, Theory of Cryptography Conference, 2004.

[59] I. Damgård, V. Pastro, N. Smart and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption", CRYPTO 2012: Advances in Cryptology – CRYPTO 2012, pp. 643–662, Annual Cryptology Conference, 2012.

[60] I. Bentov, R. Kumaresan, "How to Use Bitcoin to Design Fair Protocols.”, CRYPTO 2014: Advances in Cryptology – CRYPTO 2014, pp. 421–439, Annual Cryptology Conference, 2014

[61] Microsoft SEAL repository documentation, https://github.com/microsoft/SEAL/blob/main/native/examples/1_bfv_basics.cpp, accessed 3 August 2022.

Παράρτημα Ι [ΚΩΔΙΚΑΣ]

Τμήμα υλοποίησης της προσέγγισης Naïve:

```
Ciphertext multiply_add_ciphertexts(SEALContext context, Ciphertext a_cipher, Ciphertext b_cipher, Ciphertext value_to_add){
   Evaluator evaluator(context);
   Ciphertext product_of_a_and_b;
   Ciphertext result;
   evaluator.multiply(a_cipher, b_cipher, product_of_a_and_b);
   evaluator.add(product_of_a_and_b, value_to_add, result);
   return result;
}
```

Τμήμα υλοποίησης της προσέγγισης Halevi-Shoup:

```
void encrypt_matrix_halevi_shoup(int diagonalMatrix[][N]){

   EncryptionParameters parms(scheme_type::bfv);
   size_t poly_modulus_degree = 8192;
   parms.set_poly_modulus_degree(poly_modulus_degree);
   parms.set_coeff_modulus(CoeffModulus::BFVDefault(poly_modulus_degree));
   parms.set_plain_modulus(PlainModulus::Batching(poly_modulus_degree, 20));
   SEALContext context(parms);

   auto qualifiers = context.first_context_data()->qualifiers();

   KeyGenerator keygen(context);
   SecretKey secret_key = keygen.secret_key();
   PublicKey public_key;
   keygen.create_public_key(public_key);
   RelinKeys relin_keys;
   keygen.create_relin_keys(relin_keys);
   Encryptor encryptor(context, public_key);
   Evaluator evaluator(context);
```

```
    Decryptor decryptor(context, secret_key);
    GaloisKeys galois_keys;
    keygen.create_galois_keys(galois_keys);
    BatchEncoder batch_encoder(context);

    size_t slot_count = batch_encoder.slot_count();
    size_t row_size = slot_count / 2;

    vector<uint64_t> vector_to_encode = createVectorToEncode(vector_global, slot_count);

    Plaintext plaintext_vector;
    batch_encoder.encode(vector_to_encode, plaintext_vector);
    Ciphertext encrypted_vector;
    encryptor.encrypt(plaintext_vector, encrypted_vector);

    Ciphertext previousResult;

    for(int i=0; i<N; i++){
      vector<uint64_t> matrix_row_to_encode(slot_count, 0ULL);

      for(int p=0; p<N; p++){
        matrix_row_to_encode[p] = diagonalMatrix[i][p];
      }

      Plaintext plaintext_matrix_row;
      batch_encoder.encode(matrix_row_to_encode, plaintext_matrix_row);
      Ciphertext encrypted_matrix_row;
      encryptor.encrypt(plaintext_matrix_row, encrypted_matrix_row);

      // multiply the rotated vector with the row
      evaluator.multiply_inplace(encrypted_matrix_row, encrypted_vector);
      // rotate by 1 slot to the left
      evaluator.rotate_rows_inplace(encrypted_vector, 1, galois_keys);
      if(i!=0){
        evaluator.add_inplace(encrypted_matrix_row, previousResult);
      }
      evaluator.relinearize_inplace(encrypted_matrix_row, relin_keys);
      previousResult = encrypted_matrix_row;
      Plaintext plain_result;
      vector<uint64_t> pod_result;

      decryptor.decrypt(encrypted_matrix_row, plain_result);
      batch_encoder.decode(plain_result, pod_result);
    }
}
```

Παράρτημα ΙΙ [ΟΔΗΓΟΣ ΕΓΚΑΤΑΣΤΑΣΗΣ]

Οι παρακάτω οδηγίες ισχύουν για λειτουργικό σύστημα Linux Ubuntu 20.04.3.

1) Εγκατάσταση Cmake με την χρήση του Ubuntu Software λογισμικού που υπάρχει προ-εγκατεστημένο.

2) Εγκατάσταση git: “sudo apt install git”

3) Εγκατάσταση clang: “sudo apt install clang”

4) Κάνουμε clone το repository του SEAL: “git clone https://github.com/microsoft/SEAL.git"

5) Στο root directory του repository (όπου υπάρχει το αρχείο CMakeLists.txt) εκτελούμε την εντολή “cmake .”

6) Στο ίδιο directory εκτελούμε “make -j”.

7) Τέλος, εκτελούμε “sudo make install” για την εγκατάσταση του Microsoft SEAL.