Todo list

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Πολυτεχνική Σχολή

ΤΙΤΛΟΣ

Μελέτη Πρωτοκόλλων Ασφαλούς Υπολογισμού Πολλών Μερών (Study on Secure Multi Party Protocols)

Διπλωματική εργασία

του

ΘΕΟΔΩΡΟΥ ΣΥΜΕΩΝΙΔΗ

Επιβλέπων:	Ιωάννης Σταματίου Καθηγητής
Συνεπιβλέπων:	Αριστείδης Ηλίας Ε.ΔΙ.Π
Μέλος Εξεταστικής Επιτροπής:	Χρήστος Μακρής Αν. Καθηγητής

Πάτρα 3 Οκτωβρίου 2022

Περίληψη

Στην παρούσα διπλωματική εργασία γίνεται εισαγωγή και μελέτη σε διαφορετικά πρωτόκολλα Υπολογισμού Πολλών Μερών (Secure Multi Party Computation ή SMPC), μερικά από τα οποία στη συνέχεια εφαρμόζονται για τη δημιουργία, της πρώτης στη βιβλιογραφία, BLAS βιβλιοθήκης, για Ασφαλή Υπολογισμό Δύο Μερών, BLAS συναρτήσεων Επιπέδου-1 για πραγματικούς αριθμούς κινητής υποδιαστολής μονής ακρίβειας. Πριν ξεκινήσουμε τη μελέτη αυτή, γίνεται μια εισαγωγή στην κρυπτογραφία και στην αποδείξιμη θεωρητική ασφάλεια παρουσιάζοντας όλο το απαραίτητο θεωρητικό υπόβαθρο, μαθηματικό και αλγοριθμικό, που απαιτείται για τη βαθύτερη κατανόηση του αντικειμένου. Στη μελέτη αυτή εστιάζουμε σε σύγχρονα κρυπτογραφικά εργαλεία που χρησιμοποιούνται σήμερα ως συστατικά στοιχεία για την κατασκευή σύνθετων κρυπτογραφικών σχημάτων, δίνοντας περισσότερη έμφαση σε εργαλεία που χρησιμοποιούνται για την κατασκευή σχημάτων Ασφαλούς Υπολογισμού Πολλών Μερών.

Abstract

On the current diploma thesis we present an introduction and a study on various Secure Multi Party Computation Protocols (SMPC) where some of them are used next to implement the first SMPC BLAS Level-1 library for single precision real floating point numbers. Before this begin this study, we present an introduction to cryptography and provable theoretical security including all the required mathematical and algorithmic background needed for the deeper understand of the subject. On this study we focus on modern cryptographic tools that are used today as a building part to construct complex cryptographic schemes while giving more focus on the set of these tools that are used mostly in the construction of SMPC schemes.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τους ανθρώπους που συνέβαλαν στην ολοκλήρωση της παρούσας ιπλωματικής εργασίας αλλά και τους ανθρώπους που στάθηκαν δίπλα μου κατά της περίοδο εκπόνησης της. Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, Ιωάννη Σταματίου, καθώς τον συνεπιβλέπον Αριστείδη Ηλία, Ε.ΔΙ.Π. για την πολύτιμη συνεισφορά και βοήθεια τους. Επίσης, θα ήθελα να ευχαριστήσω όλα τα μέλη του εργαστηρίου του κύριου Χρήστου Μακρή, Αναπληρωτή Καθηγητή, στο οποίο βρισκόμουν όλο τον καιρό εκπόνησης αυτής της διπλωματικής εργασίας, για την πολύτιμη βοήθεία τους σε τεχνικά θέματα αλλά και για την καλή τους παρέα. Τα μέλη αυτά είναι ο Αγοράκης Μπομπότας, ο Νικήτας Καλογερόπουλος και ο ΠαναγιώτηςΚεχαγιάΤέλος. Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου, στους φίλους μου και στην κοπέλα μου για την κατανόηση, τη συμπαράσταση και τη στήριξή τους.

Αφιέρωση

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Acronyms

This document is incomplete. The external file associated with the glossary `acronym' (which should be called main.acr) hasn't been created.

Check the contents of the file main.acn. If it's empty, that means you haven't indexed any of your entries in this glossary (using commands like \gls or \glsadd) so this list can't be generated. If the file isn't empty, the document build process hasn't been completed.

You may need to rerun LaTeX. If you already have, it may be that TeX's shell escape doesn't allow you to run makeindex. Check the transcript file main.log. If the shell escape is disabled, try one of the following:

- Run the external (Lua) application:

  makeglossaries-lite "main"

- Run the external (Perl) application:

  makeglossaries "main"

Then rerun LaTeX on this document.
This message will be removed once the problem has been fixed.

Στην παρούσα εργασία οι όροι κρυπτογραφικό σχήμα, κρυπτοσύστημα χρησιμοποιούνται ως συνώνυμες, σε αντίθεση με τον όρο κρυπτογραφικό κατασκεύασμα που χρησιμοποιείται κυρίως για να αναφερθούμε σε παλαιότερους τρόπους κρυπτογράφησης που χρησιμοποιούνταν στην αρχαιότητα και βασίζονταν κυρίως σε κατασκευαστικούς μηχανισμούς.

Κεφάλαιο 1

Εισαγωγή

Στα πλαίσια αυτής της διπλωματικής εργασίας κάνουμε μια μελέτη στις βασικές έννοιες της Κρυπτογραφίας, Αποδείξιμης Ασφάλειας και του Ασφαλούς και Επαληθεύσιμου Υπολογισμού, κυρίως μέσω των πρωτοκόλλων Ασφαλούς Υπολογισμού Πολλών Μερών. Θα μελετήσουμε δηλαδή τα βασικά συστατικά στοιχεία που χρειάζονται στην κατασκευή και στην απόδειξη της ασφάλειας ενός πρωτοκόλλου αυτού του είδους. Μελετάτε ένα ευρύ φάσμα αυτών των πρωτοκόλλων, ξεκινώντας από τα πρώτα πρωτόκολλα που προτάθηκαν στη βιβλιογραφία φτάνοντας μέχρι και τα πιο σύγχρονα. Τέλος, παρουσιάζεται μια Ασφαλούς Υπολογισμού Δύο Μερών BLAS Level-1 βιβλιοθήκης, η οποία κατά τη γνώση μας, παρουσιάζεται πρώτη φορά στη βιβλιογραφία, και χρησιμοποιεί κρυπτογραφικά εργαλεία και έννοιες που θα αναλύσουμε στη συνέχεια.

1.1 Κρυπτογραφία

Η λέξη κρυπτογραφία προέρχεται από τη σύνθεση δύο ελληνικών λέξεων, 'κρύπτο' που σημαίνει μυστικό και 'γράφω'.

1.1.1 Ιστορική αναδρομή

Από την αρχαιότητα κιόλας, υπήρχε η ανάγκη για ασφαλή ανταλλαγή μηνυμάτων. Έτσι, σε πολλούς αρχαίους πολιτισμούς έχει παρατηρηθεί ότι χρησιμοποιούνταν πρωταρχικά κρυπτογραφικά κατασκευάσματα, όπως αυτά της Σκυτάλης [1] και του Καίσαρα [2]. Ύστερα, παράλληλα με την πρόοδο των μαθηματικών, παρατηρούνται να εμφανίζονται όλο και πιο σύνθετα κρυπτογραφικά κατασκευάσματα βασισμένα στα αυτά. Ιστορικά, η κρυπτογραφία στον πόλεμο αποτελούσε μια μορφής όπλο, ο αντιμαχόμενος που είχε το ισχυρότερο σύστημα ή αυτός που μπορούσε να παραβιάσει την ασφάλεια του συστήματος του αντιπάλου, είχε πρόσβαση σε πληροφορίες που το εξασφάλιζαν σημαντικό πλεονέκτημα. Όπως και στην

περίπτωση του πολεμικού εξοπλισμού, άρχισε να εμφανίζεται ανταγωνισμός (arms race) μεταξύ των κρυπτογραφικών κατασκευασμάτων [3]. Ιστορικά, το πιο γνωστό τέτοιο παράδειγμα είναι η μηχανή Enigma που χρησιμοποιούσαν οι Γερμανοί στον 2ο Π.Π. την οποία οι Συμμαχικές Δυνάμεις, με τη βοήθεια του Άλαν Τούρινγκ και τον συνάδελφων του, κατάφεραν να παραβιάσουν και να αποκτήσουν σοβαρό πλεονέκτημα στον πόλεμο. Έτσι, παράλληλα με την κρυπτογραφία, αλλά κυρίως από τότε που άρχισε να παίρνει πιο μαθηματική διατύπωση, αναπτύσσεται και ο κλάδος της κρυπτανάλυσης. Η κρυπτανάλυση, στην πιο απλή της μορφή, πρόκειται για την ανάλυση ενός κρυπτοσυστήματος, δίχως πρόσβαση στην κρυφή πληροφορία, συνήθως με τη μορφή κλειδιού, που επιτρέπει αποτελεσματικά την αποκωδικοποίηση του κρυπτοκειμένου, με σκοπό την παραβίαση του και την απόκτηση του απλού κειμένου [4]. Οι έννοιες κρυπτογραφικό κλειδί και κρυπτοκείμενο θα οριστούν στην επόμενη ενότητα. Η κρυπτογραφία και η κρυπτανάλυση εντάσσονται στον ευρύτερο κλάδο της κρυπτολογίας.

1.1.2 Σύγχρονη Κρυπτογραφία

Ο κλάδος άρχισε γνωρίζει μεγάλη άνθηση από τότε που οι άνθρωποι άρχισαν να χρησιμοποιούν, μαζικά, τηλεπικοινωνιακά συστήματα για να επικοινωνούν σε μεγάλες αποστάσεις, δηλαδή στις αρχές του 20ού αιώνα, αλλά και πιο συγκεκριμένα όταν η πληροφορία που μετέφεραν άρχισε να γίνεται απόρρητη και να αποκτά μεγάλη αξία, δηλαδή με την έλευση του 2ου Π.Π. Τα πρώτα θεμέλια της μαθηματικής κρυπτογραφίας μπήκαν από τις επιστημονικές εργασίες του Kerchoffs [5] το 1883 και του Shannon [6] το 1949. Στην πρώτη, ο Kerchoffs έθεσε τις βασικές σχεδιαστικές αρχές που πρέπει να διέπουν ένα κρυπτογραφικό σχήμα, μεταξύ των οποίων την πασίγνωστη σχεδιαστική αρχή κατά την οποία η ασφάλεια ενός σχήματος πρέπει να έγκειται μόνο στη μυστικότητα του κλειδιού και να μην εξαρτάται από τη μυστικότητα του αλγορίθμου κρυπτογράφησης. Στην δεύτερη, η κρυπτογραφία μετατρέπεται σε αυστηρό επιστημονικό πεδίο, όπου ορίζεται η έννοια του κρυπτοσυστήματος και η απόλυτη ασφάλεια. Ιστορικά, να αναφέρουμε πως όλα τα κρυπτογραφικά κατασκευάσματα μέχρι τα μέσα του 20ου αιώνα, που άρχισαν να γνωρίζουν άνθηση οι υπολογιστές, κατατάσσονται στην εποχή της Κλασσικής Κρυπτογραφίας, από εκεί και έπειτα που άρχισε να διατυπώνεται με αυστηρό μαθηματικό τρόπο ξεκινάει η εποχή της Σύγχρονης Κρυπτογραφίας. Επαναστατικές ήταν επίσης και οι εργασίες των Diffi-Hellman [7] το 1976 και των Rivest, Shamir, Adleman το 1977 [8], η οποίες ξεκίνησαν την επανάσταση της κρυπτογραφίας δημόσιου κλειδιού. Στην πρώτη θέτονται τα θεμέλια της κρυπτογραφίας δημόσιου κλειδιού και παρουσιάζεται το πρώτο πρωτόκολλο ανταλλαγής κλειδιών, το οποίο χρησιμοποιείται μέχρι και σήμερα και είναι γνωστό στην βιβλιογραφία με τα ονόματα των συγγραφέων της εργασίας, το Πρωτόκολλο Diffie-Hellman (DH) ή αλλιώς το Πρωτόκολλο Ανταλλαγής Κλει-

διών Diffie-Hellman (Diffie Hellman Key Exchange ή DHKE). Ωστόσο, στην εργασία αυτή δεν παρουσιάζεται κάποιο κρυπτογραφικό σχήμα δημοσίου κλειδιού με υποστήριξη ανταλλαγής μηνυμάτων ή ψηφιακών υπογραφών. Αυτό παρουσιάστηκε για πρώτη φορά στη δεύτερη εργασία και είναι το πρώτο ιδιαίτερα επιτυχημένο κρυπτοσύστημα δημόσιου κλειδιού, το RSA, που χρησιμοποιείται με παραλλαγές μέχρι και σήμερα. Το κρυπτοσύστημα αυτό αναπτύχθηκε παράλληλα, το 1973, από τον Clifford Cocks όταν δούλευε για το GCHQ, ωστόσο έγινε δημόσια διαθέσιμο (declassified) το 1997.

1.1.3 Η κρυπτογραφία σήμερα

Σήμερα, βρισκόμαστε στην εποχή της 4ης Βιομηχανικής Επανάστασης [9] [10], το διαδίκτυο πλέον αριθμεί αρκετά δισεκατομμύρια συσκευές και καθημερινά παράγονται και αποθηκεύονται αρκετά μεγαλύτερες τάξεις μεγέθους από bytes δεδομένων. Οι εφαρμογές τις κρυπτογραφίας έχουν πλέον ξεφύγει από τους καθαρά πολεμικούς σκοπούς του παρελθόντος, οι οποίοι συνεχίζουν ωστόσο να είναι τα μεγαλύτερα κίνητρα, και έχουν πάρει μια πιο κοινωνικοποιημένη μορφή. Βρίσκονται πλέον σιωπηρά στην καθημερινότητα κάθε ανθρώπου. Η κρυπτογραφία αποτελεί πλέον επιστημονικό κλάδο με πολύ ισχυρά θεμέλια. Το φάσμα της έχει απλωθεί σε πάρα πολλούς τομείς, όπως της ασφαλούς επικοινωνίας (τηλεπικοινωνίες, εφαρμογές μηνυμάτων κα.), της ασφαλούς πρόσβασης και συναλλαγών (τραπεζικές συναλλαγές, ανέπαφες συναλλαγές κα.), των ηλεκτρονικών ψηφοφοριών (κρατικές ψηφοφορίες κα.), ασφαλών αναθέσιμων υπολογισμών και αποθήκευσης (αναθέσιμοι ασφαλείς cloud υπολογισμοί και αποθήκευση κα.), ενσωματωμένων εφαρμογών (συστήματα ασφαλείας όπως συναγερμοί, έξυπνα αντικείμενα όπως συσκευές οικιακής χρήσης κα.) και αποκεντρωμένων εφαρμογών και νομισματικών συστημάτων, όπως αυτή του κρυπτονομίσματος Bitcoin και του Ethereum. Γίνεται άμεσα αντιληπτή η σύνδεση και η αξιοποίηση της από πολλούς επιστημονικούς και βιομηχανικούς κλάδους. Ωστόσο, αξίζει να αναφέρουμε ότι στα περισσότερα προαναφερθέντα συστήματα η κρυπτογραφία παίζει τον ρόλο του διαφανούς ενδιάμεσου επιπέδου (transparent middle layer).

1.2 Ασφαλής Υπολογισμός (Secure Computation)

1.2.1 Αναθέσιμος Υπολογισμός και Αποθήκευση (Outsource Computation and Storage)

Η έννοιες του Αναθέσιμου Υπολογισμού (Outsourced Computation) και της Αναθέσιμης Αποθήκευσης (Outsourced Storage), με την ευρύτερη ονομασία Νέφους (Cloud) ή και με τις ισοδύναμες τους, αυτές του Υπολογισμού σε Νέφος (Cloud Computing) και Αποθήκευσης σε

Νέφος (Cloud Storage) αντίστοιχα, γίνονται όλο και πιο διαδεδομένες στις μέρες μας. Πολλές φορές, η χρήση τους προέρχεται από επιτακτική ανάγκη, πέρα από επιλογή. Ας αναλύσουμε πως όμως προέκυψε η ανάγκη για τις δύο αυτές έννοιες Αναθεσιμότητας. Η ανάγκη αυτή προκύπτει, από το ότι ο αριθμός της υπολογιστικής ισχύς και αποθηκευτικής ικανότητας της μέσης συνδεδεμένης συσκευής ανά μονάδα χώρου αυξάνεται πολύ πιο αργά σε σχέση με τον αριθμό τον δεδομένων που παράγει και καλείται επεξεργαστεί. Πιο συγκεκριμένα, η αύξηση που παρατηρείται στην περίπτωση της υπολογιστικής και αποθηκευτικής ισχύος ανά μονάδα χώρου είναι το πολύ γραμμική στη μονάδα του χρόνου, ενώ αντίθετα η αύξηση των παραγομένων δεδομένων είναι τουλάχιστον πολυωνυμική στην ίδια μονάδα χρόνου. Αντιθέτως, ο ρυθμός αύξησης της μέσης ταχύτητας σύνδεσης μιας συσκευής στο διαδίκτυο παρατηρείται να είναι εκθετικός, γεγονός που κάνει τις έννοιες των Αναθέσιμου Υπολογισμού και Αποθήκευσης ιδιαίτερα ελκτικές [11] [12] [13]. Για να υποστηρίξουμε τους παραπάνω ισχυρισμούς, αρκεί να αναλογιστούμε πόσοι από εμάς "Δεν διαθέτουμε χώρο στο κινητό μας τηλέφωνο" και αναγκαζόμαστε να χρησιμοποιήσουμε κάποια εξωτερική υπηρεσία αποθήκευσης νέφους ώστε να αποθηκεύσουμε τα αρχεία μας, και πόσοι επιστημονικοί υπολογισμοί και πειράματα εκτελούνται σε υπηρεσίες Νέφους λόγω του ότι δε διαθέτουμε, ή είναι ασύμφορο να την αποκτήσουμε για τη χρήση μας, την κατάλληλη υλικοτεχνική υποδομή

1.2.2 Ασφαλής Αναθέσιμος Υπολογισμός και Αποθήκευση (Secure Outsourced Computation and Storage)

Θα εξετάσουμε ένα παράδειγμα, από το Διαδίκτυο των Πραγμάτων, για το πως προκύπτει η ανάγκη για εφαρμογή κρυπτογραφικών μεθόδων στη χρήση του Αναθέσιμου Υπολογισμού και Αποθήκευσης. Πλέον, οι αισθητήρες και οι έξυπνες συσκευές (smart devices) βρίσκονται όλο και περισσότερο μέσα στην καθημερινότητα μας, χωρίς απαραίτητα να μας γίνεται αντιληπτό. Το Διαδίκτυο των Πραγμάτων (Internet of Things ή IoT) αρχίζει να είναι όλο ένα και περισσότερο μέρος της πραγματικότητας [14] [15] . Έξυπνοι αισθητήρες, συσκευές και μηχανές είναι πλέον μέρος αυτής. Όμως, όπως είδαμε και προηγουμένως, το μεγαλύτερο ποσοστό αυτών των συσκευών είναι ανίκανο να επεξεργαστεί και να αποθηκεύσει τον όγκο των δεδομένων που παράγει. Έτσι οι περισσότερες συσκευές, όπως για παράδειγμα ένα έξυπνο ρολόι (smartwatch), πρέπει να στείλουν τα δεδομένα τους σε τρίτους με την κατάλληλη υπολογιστική ισχύ και αποθηκευτική ικανότητα ώστε να κάνουν τους υπολογισμούς και την αποθήκευση των δεδομένων. Έτσι γεννήθηκε το Cloud computing και storage. Τα δεδομένα όμως μπορεί να είναι προσωπικά ή ευαίσθητα ώστε να μην θέλουμε αυτός ο τρίτος να μπορεί να έχει πρόσβαση στο περιεχόμενο τους για να τα επεξεργαστεί, πόσο μάλλον για να τα αποθηκεύσει. Αυτή η κεντροποίηση (centralization) του υπολογισμού και της αποθήκευσης εναποθέτει την ασφάλεια των δεδομένων σε κάποιον Έμπιστο Τρίτο Μέρος (Trusted Third

Party ή TTP). Δυστυχώς, όσο έμπιστος και αν είναι αυτός ο τρίτος, στην πράξη δημιουργείται Μοναδικό Σημείο Αστοχίας (Single Point of Failure) αφού αυτός ο τρίτος καταλήγει να κατέχει πολύτιμη ποσότητα δεδομένων και να γίνεται στόχος κυβερνοεπιθέσεων, το οποίο επαληθεύεται από τις δεκάδες διαρροές δεδομένων (data breaches) που συμβαίνουν πλέον καθημερινά. Έτσι, προκύπτει η ανάγκη για εφαρμογή κρυπτογραφικών μεθόδων με απώτερο σκοπό να περιοριστεί η πρόσβαση στα δεδομένα των χρηστών στις πλέον απαραίτητες οντότητες. Μπορούμε να φανταστούμε ένα σενάριο στο οποίο, κρυπτογραφήσουμε τα δεδομένα πριν τα στείλουμε σε κάποιον τρίτο για επεξεργασία, αυτός να κάνει τον υπολογισμό πάνω στην κρυπτογραφημένη τους μορφή και να μας επιστρέφει τα αποτελέσματα, ή την ανάκτηση αυτών, είτε σε κρυπτογραφημένη μορφή που μόνο εμείς θα έχουμε πρόσβαση, είτε σε απλή (μη κρυπτογραφημένη μορφή), όπου στην περίπτωση αυτή θα είχε πρόσβαση και αυτός ο τρίτος, ανάλογα με το αν το αποτέλεσμα του υπολογισμού το θεωρούμε ευαίσθητο ή όχι. Αντίστοιχα, μπορούμε να σκεφτούμε και παρόμοια σενάρια σχετικά με την αποθήκευση δεδομένων.

Το πρόβλημα που σχετίζεται με την με την Ασφαλή Αποθήκευση (Secure Storage) των δεδομένων, είναι εκ φύσεως πιο εύκολο στην επίλυση του και έχει σε ένα βαθμό επιλυθεί στις μέρες μας, για αυτό και δεν θα αναφερθούμε περαιτέρω σε αυτό στα πλαίσια αυτής της εργασίας. Ο αναγνώστης που ενδιαφέρεται να διαβάσει περισσότερα για αυτό μπορεί να ανατρέξει στις εργασίες [16] [17].

Το πρόβλημα που σχετίζεται με την ασφαλή επεξεργασία των δεδομένων έρχεται να λύσει ο Ασφαλής Υπολογισμός (Secure Computation), που χρησιμοποιείται ως μια γενική έννοια που συμπεριλαμβάνει όλες τις μεθόδους που χρησιμοποιούνται για τον υπολογισμό σε κρυπτογραφημένα δεδομένα. Πέραν όμως του ασφαλούς υπολογισμού, μας ενδιαφέρει και η επαλήθευση των αποτελεσμάτων, δηλαδή η εξακρίβωση αν τα αποτελέσματα όντως προέκυψαν από τον επιθυμητό υπολογισμό, αυτό είναι το αντικείμενο του Επαληθέυσιμου Υπολογισμού (Verifiable Computation). Ο Ασφαλής και ο Επαληθεύσιμος υπολογισμός αποτελούν από τους πιο ενεργούς κλάδους της κρυπτογραφίας σήμερα. Υπάρχουν δύο κύριοι υποκλάδοι που συνδυάζουν και τα δύο αυτά χαρακτηριστικά, ο Ασφαλής Αναθέσιμος Υπολογισμός (Secure Outsourced Computation) και ο Ασφαλής Υπολογισμός Πολλών Μερών (Secure Multi-Party Computation ή (SMPC)[1]. Η κύρια διαφορά του Ασφαλής Υπολογισμού Πολλών Μερών και του Ασφαλή Αναθέσιμου Υπολογισμού είναι ότι ο πρώτος περιλαμβάνει αποκλειστικά μη διαδραστικά (non-interactive) πρωτόκολλα ενώ ο δεύτερος περιλαμβάνει αποκλειστικά διαδραστικά (interactive) πρωτόκολλα.

[1]Πρέπει να αναφέρουμε ότι στην βιβλιογραφία οι όροι Ασφαλής Υπολογισμός Πολλών Μερών (Secure Multi-Party Computation ή SMPC) και Υπολογισμός Πολλών Μερών (Multi-Party Computation ή MPC) χρησιμοποιούνται ως ταυτόσημοι, ωστόσο επειδή ο πρώτος είναι πιο περιγραφικός, στα πλαίσια αυτής της εργασίας θα χρησιμοποιήσουμε αυτόν.

1.2.2.1 Ομομορφική Κρυπτογράφηση (Homomorphic Encryption ή HE)

Η Ομομορφική Κρυπτογράφηση είναι ένα από τα βασικότερα εργαλεία που χρησιμοποιούνται στην επίτευξη Ασφαλούς Αναθέσιμου Υπολογισμού. Με τον όρο Ομομορφική Κρυπτογράφηση αναφερόμαστε σε μια κλάση μη διαδραστικών κρυπτογραφικών σχημάτων που επιτρέπουν ανάλογα με τη φύση τους, την εφαρμογή μίας ή περισσότερων πράξεων, για μία ή περισσότερες φορές σε ένα κρυπτοκείμενο. Η έννοια αυτή είναι πολύ στενά συνδεδεμένη με αυτήν της Ευπλαστότητας (Malleability). Παρακάτω ορίζουμε την Ομομορφική Κρυπτογράφηση και τις κύριες κατηγορίες σχημάτων που εντάσσονται σε αυτήν.

Definition 1.2.1. Ομομορφικό Σχήμα Κρυπτογράφησης (Homomorphic Encryption Scheme ή HE Scheme) : Ένα κρυπτογραφικό σχήμα είναι ομομορφικό σε μια πράξη $\star$ όταν υποστηρίζει την παρακάτω ισοδυναμία

$$E_k(m_1) \star E_k(m_2) = E_k(m_1 \star m_2), \forall m_1, m_2 \in M \tag{1.1}$$

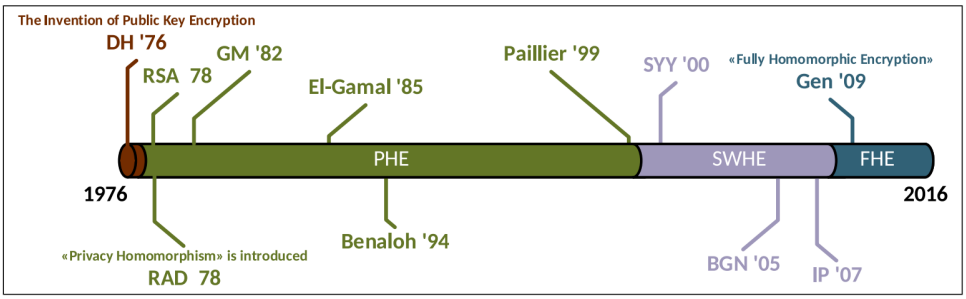
όπου E είναι ο αλγόριθμος κρυπτογράφησης και M ο χώρος των μηνυμάτων.

Διακρίνουμε τις εξής κατηγορίας Ομομορφικής Κρυπτογραφίας :

Definition 1.2.2. Κύριες Κατηγορίες Ομομορφικής Κρυπτογράφησης :

- **Μερικώς Ομομορφικό Σχήμα Κρυπτογράφησης (Partially Homomorphic Scheme)** : Επιτρέπει μόνο ένα είδος πράξης χωρίς περιορισμό στον αριθμό των φορών που μπορεί να εφαρμοστεί σε ένα κρυπτοκείμενο.
- **Κάπως Ομομορφικό Σχήμα Κρυπτογράφησης (Somewhat Homomorphic Encryption Scheme ή SWHE Scheme)** : Επιτρέπει περιορισμένο αριθμό πράξεων που μπορούν να εφαρμοστούν περιορισμένες φορές σε ένα κρυπτοκείμενο.
- **Πλήρες Ομομορφικό Σχήμα Κρυπτογράφησης (Fully Homomorphic Encryption Scheme ή FHE Scheme)** : Δεν έχει περιορισμό ούτε στο πλήθος των πράξεων ούτε στον αριθμό των φορών που μπορούν αυτές να εφαρμοστούν σε ένα κρυπτοκείμενο.

Το 2009 ο Gentry στη διατριβή του [18] δημιούργησε το πρώτο Πλήρες Ομομορφικό Κρυπτογραφικό Σχήμα, ένα ανοιχτό πρόβλημα για πάνω από 30 χρόνια στον κλάδο της κρυπτογραφίας. Ωστόσο, παρά την πολλή έντονη ερευνητική δραστηριότητα και τα πολλά κρυπτογραφικά σχήματα που έχουν προταθεί στη βιβλιογραφία, στον κλάδο της Πλήρους Ομομορφικής Κρυπτογραφίας, δεν έχουν βρεθεί ακόμα ιδιαίτερα πρακτικά σχήματα που να μπορούν να χρησιμοποιηθούν αποτελεσματικά στην πράξη. Συνήθως είτε έχουν πολύ μεγάλο μέγεθος κλειδιών είτε η διαδικασία εκκίνησης του πρωτοκόλλου είναι πολύ χρονοβόρα [19].



Σχήμα 1.1: Η ιστορική εξέλιξη των Ομομορφικών Σχημάτων μέχρι το πρώτο Πλήρες Ομομορφικό Κρυπτογραφικό Σχήμα [20].

Στο Σχήμα 1.1 βλέπουμε την Ιστορική εξέλιξη της Ομομορφικής Κρυπτογραφίας μέχρι την ανακάλυψη του πρώτου Πλήρες Ομομορφικού Κρυπτογραφικού Σχήματος από τον Gentry.

1.2.2.2 Ασφαλής Υπολογισμός Πολλών Μερών (Secure Multi Party Computation ή SMPC)

Με τον όρο Ασφαλής Υπολογισμών Πολλών Μερών, αναφερόμαστε σε μια κλάση διαδραστικών κρυπτογραφικών σχημάτων η οποία επιτρέπει σε δύο ή περισσότερους συμμετέχοντες που διαθέτουν αντίστοιχες εισόδους (ο κάθε συμμετέχον μπορεί να έχει παραπάνω από μία είσοδο, για χάριν απλότητας όμως μπορούμε να τις θεωρήσουμε ως μια, η οποία μπορεί να ενσωματώνει περισσότερες από μία μέσω κάποιας δομής όπως μια λίστα ή ένα διατεταγμένος ζεύγος), αλλά δεν εμπιστεύονται ο ένας τον άλλον, να υπολογίσουν τις εξόδους μιας συνάρτησης εισάγοντας δημόσια τις από εισόδους τους και εκτελώντας κάποιο κατανεμημένο, ή και μη, πρωτόκολλο. Ο στόχος ενός SMPC πρωτοκόλλου είναι να υπολογίσει σωστά τη συνάρτηση με βάση την είσοδο του κάθε συμμετέχον και να αποκρύψει από κάθε συμμετέχον οποιαδήποτε πληροφορία σχετικά με την είσοδο κάποιου άλλου συμμετέχοντος. Στην περίπτωσή των δύο συμμετεχόντων, αναφερόμαστε ως Ασφαλής Υπολογισμών Δύο Μερών (Secure Two Party Computation ή 2PC). Ο Ασφαλής Υπολογισμός Πολλών Μερών ορίζεται ως εξής :

Definition 1.2.3. [Ασφαλής Υπολογισμός Πολλών Μερών (Secure Multi Party Computation ή SMPC)]Ασφαλής Υπολογισμός Πολλών Μερών (Secure Multi Party Computation ή SMPC) : Είναι μια κλάση διαδραστικών κρυπτογραφικών σχημάτων που επιτρέπουν σε συμμετέχοντες $P_1, P_2, \dots, P_n$ με αντίστοιχες εισόδους $x_1, x_2, \dots, x_n$ να υπολογίσουν την συνάρτηση την έξοδο y της συνάρτησης f, ως $y = f(x_1, x_2, \dots, x_n)$ και ικανοποιούν τις παρακάτω ιδιότητες :

- **Ορθότητα** : Η έξοδος y είναι η σωστή έξοδος της συνάρτησης για τις δεδομένες εισόδους $x_1, x_2, \dots, x_n$

- **Ιδιωτικότητα** : Η έξοδος y είναι η μόνη πληροφορία που φανερώνει το πρωτόκολλο.

Το πρώτο πρωτόκολλο για SMPC παρουσιάστηκε από τον Yao το 1986 στις εργασίες [21] [22] και προτάθηκε ως πρωτόκολλο επίλυσης του Προβλήματος των Εκατομμυριούχων (Millionaires' Problem). Στην βιβλιογραφία σήμερα αναφέρεται ως Μπερδεμένα Δίκτυα Yao (Yao's Garbled Circuits ή Yao's-GC ή YaoGC). Θα αναλύσουμε το πρωτόκολλο αυτό καθώς και άλλα πρωτόκολλα για SMPC στην Ενότητα 4.

Κεφάλαιο 2

Κρυπτογραφία

Στο κεφάλαιο αυτό θα γίνει μια εισαγωγή στην Κρυπτογραφία. Αρχικά θα αναλύσουμε βασικούς ορισμούς/μοντέλα κρυπτογραφικών σχημάτων, ξεκινώντας από πιο ιστορικούς και καταλήγοντας σε πιο σύγχρονους. Πολλοί από αυτούς θα χρησιμοποιηθούν στο Κεφάλαιο 3 ως βασικά δομικά στοιχεία. Στη συνέχεια του κεφαλαίου θα αναλύσουμε κυρίως σύγχρονα κρυπτογραφικά εργαλεία και πρωτόκολλα που χρησιμοποιούνται ως συστατικά στοιχεία σε σύγχρονα πρωτόκολλα και πολλά από αυτά θα χρησιμοποιηθούν ως βασικά δομικά στοιχεία στα Κεφάλαια 4 και 5. Σε καμία περίπτωση αυτό το κεφάλαιο δεν αποτελεί μια πλήρη εισαγωγή στον κλάδο της κρυπτογραφίας. Ο αναγνώστης που ενδιαφέρεται για κάτι τέτοιο μπορεί να απευθυνθεί στα βιβλία [23], [24], [25], το τελευταίο εκ των οποίων είναι και στην ελληνική γλώσσα.

2.1 Βασικοί Ορισμοί

2.1.1 Πρωταρχικοί Ορισμοί

Αρχικά θα αναφερθούμε στους πρώτους ιστορικά τυποποιημένους ορισμούς κρυπτογραφικών σχημάτων που έγιναν από τον Shannon, οι οποίοι αναφέρονται σε συστήματα συμμετρικής κρυπτογραφίας, με απώτερο σκοπό να μπορέσουμε να παρουσιάσουμε με σαφήνεια τα μοντέλα των αντιπάλων, της ασφάλειας και των αποδείξεων τους στην Ενότητα 3. Αυτά στη συνέχεια μπορούν να γενικευθούν, να προσαρμοστούν και να χρησιμοποιηθούν σε όλα τα κρυπτογραφικά κατασκευάσματα, όπως αυτά της ασύμμετρης κρυπτογραφίας, των συναρτήσεων κατακερματισμού, των κρυπτογραφικών πρωτοκόλλων κ.α.

Ο Shannon στην εργασία [6] όρισε το παρακάτω γενικό μαθηματικό μοντέλο συμμετρικού κρυπτογραφικού σχήματος.

Definition 2.1.1. Κρυπτογραφικό Σχήμα Shannon (Shannon Cipher) : Ένα ζεύγος ντετερμινιστικών συναρτήσεων $\mathcal{E} = (\text{Enc}, \text{Dec})$ οι οποίες ορίζονται ως εξής:

- Enc : $\mathcal{K} \times \mathcal{M} \to \mathcal{C}$
- Dec : $\mathcal{K} \times \mathcal{C} \to \mathcal{M}$

όπου $\mathcal{M}, \mathcal{C}, \mathcal{K}$ είναι οι χώροι των μηνυμάτων, κρυπτοκειμένων και κλειδιών αντίστοιχα τους οποίους θα θεωρήσουμε πεπερασμένους. Στην περίπτωση αυτή λέμε ότι το κρυπτοσύστημα $\mathcal{E}$ ορίζεται πάνω στο $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. Οι συναρτήσεις Enc, Dec ορίζονται ως εξής:

- Η Enc είναι η συνάρτηση κρυπτογράφησης και παίρνει ως όρισμα ένα κλειδί $k \in \mathcal{K}$ και ένα μήνυμα $k \in \mathcal{M}$ και επιστρέφει ένα κρυπτοκείμενο $c \in \mathcal{C}$. Δηλαδή :

$$c = \text{Enc}_k(m) \text{ ή με ισοδύναμο συμβολισμό } c = \text{Enc}(k, m)$$

- Η Dec είναι η συνάρτηση αποκρυπτογράφησης, δηλαδή η αντίστροφη συνάρτηση της Enc, και παίρνει ως όρισμα ένα κλειδί $k \in \mathcal{K}$ και ένα κρυπτοκείμενο $c \in \mathcal{C}$ και επιστρέφει ένα μήνυμα $m \in \mathcal{M}$. Δηλαδή :

$$m = \text{Dec}_k(c) \text{ ή με ισοδύναμο συμβολισμό } m = \text{Dec}(k, c)$$

Όπου οι Enc, Dec ακολουθούν την παρακάτω Ιδιότητα Ορθότητας:

$$\forall m \in \mathcal{M}, \forall k \in \mathcal{K} : Pr[m = \text{Dec}_k(\text{Enc}_k(m))] = 1 \tag{2.1}$$

Ένα παράδειγμα Κρυπτογραφικού Σχήματος Shannon είναι αυτό του Σημειωματάριου Μιας Χρήσης (One Time Pad ή OTP) [26] με σταθερό ή και μεταβλητό μήκος m και c.

Δυστυχώς, ο Ορισμός 2.1.1 είναι πολύ γενικός και η κλάση των κρυπτογραφικών σχημάτων που ορίζει, συμπεριλαμβάνει προβλήματα με μη πρακτικές χρονικές και χωρικές πολυπλοκότητες. Έτσι, υπήρξε ανάγκη στην βιβλιογραφία να συμπεριληφθεί μια ακόμα κλάση κρυπτογραφικών σχημάτων που συμπεριλαμβάνει, αποκλειστικά, κρυπτογραφικά σχήματα που μπορούν να υλοποιηθούν αποδοτικά.

Definition 2.1.2. Υπολογιστικό Κρυπτογραφικό Σχήμα (Computational Cipher) [23] : Ένα ζεύγος συναρτήσεων $\mathcal{E} = (\text{Enc}, \text{Dec})$ όπως ορίζονται στο Κρυπτογραφικό Σχήμα Shannon υπό τον περιορισμό ότι οι αλγόριθμοι Enc, Dec είναι αποδοτικοί και πιθανοτικοί δηλαδή, $\text{Enc}, \text{Dec} \in PPT(1^n)$, όπου n παράμετρος ασφάλειας, διατηρώντας την Ιδιότητα Ορθότητας 2.1.

Σήμερα για την ασφαλή ανταλλαγή μηνυμάτων διακρίνονται δύο κύριες κατηγορίες Υπολογιστικών Κρυπτογραφικών Σχημάτων. Τα Συμμετρικά και Ασύμμετρα Κρυπτογραφικά σχήματα, με την λέξη "Υπολογιστικά" να παραλείπεται και να αναφέρεται μόνο όπου υπάρχει ανάγκη διάκρισης από άλλα σχήματα, αφού κυρίως μας ενδιαφέρουν πρακτικά σχήματα και σε αυτά θα επικεντρωθούμε και στα πλαίσια αυτής της εργασίας.

2.1.2 Συμμετρική Κρυπτογραφία

Η συμμετρική κρυπτογραφία παίρνει το όνομά της από την συμμετρική χρήση κλειδιών, δηλαδή το ίδιο κλειδί που θα χρησιμοποιηθεί για την κρυπτογράφηση θα πρέπει να χρησιμοποιηθεί και για την αποκρυπτογράφηση. Στην κατηγορία των συμμετρικών κρυπτογραφικών σχημάτων ανήκουν τα περισσότερα σχήματα και συστήματα της Κλασσικής Κρυπτογραφίας.

Definition 2.1.3. Συμμετρικό Κρυπτογραφικό Σχήμα (Symmetric Cryptographic Scheme) : Έστω κατάλληλα $\mathcal{M}$, $\mathcal{C}$, $\mathcal{K}$. Αποτελείτε από ένα σύνολο αποδοτικών αλγορίθμων $(\text{KGen}, \text{Enc}, \text{Dec})$ οι οποίοι ορίζονται παρόμοια με τον Ορισμό 2.1.2 και έχουν τις εξής λειτουργίες :

- $k \leftarrow \text{KGen}(1^n)$: Δημιουργεί ένα κρυπτογραφικό κλειδί k το οποίο χρησιμοποιείται τόσο για κρυπτογράφηση όσο και για αποκρυπτογράφηση.
- $c \leftarrow \text{Enc}_k(m)$: Κρυπτογραφεί ένα μήνυμα m χρησιμοποιώντας το κλειδί k.
- $m \leftarrow \text{Dec}_k(c)$: Αποκρυπτογραφεί ένα κρυπτοκείμενο c χρησιμοποιώντας το κλειδί k.

Οι παραπάνω αλγόριθμοι πρέπει να ακολουθούν την εξής Ιδιότητα Ορθότητας όπως ακριβώς και με τον Ορισμό 2.1.2.

Ένα από τα βασικότερα μειονεκτήματα των συμμετρικών κρυπτογραφικών σχημάτων είναι ότι στην περίπτωση που χρησιμοποιούνται για ασφαλή επικοινωνία για κάθε συμμετέχοντα που θέλει να συμμετάσχει στην επικοινωνία θα πρέπει ο κάθε συμμετέχοντας να διαθέτει για την επικοινωνία τους ένα μοναδικό κλειδί. Δηλαδή στην περίπτωση των n συμμετεχόντων θα χρειαστούμε n^2 κλειδιά, το πρόβλημα αυτό είναι γνωστό και ως **Πρόβλημα των Τετραγώνων στην βιβλιογραφία**. Έτσι τα σχήματα αυτά, από μόνα τους, καθίστανται ανίκανα για χρήση σε μεγάλα δίκτυα επικοινωνίας, όπως το Διαδίκτυο ή το Διαδίκτυο των Πραγμάτων. Ένα ακόμη μειονέκτημα είναι ότι οι συμμετέχοντες θα πρέπει να έχουν προαποφασίσει ένα συμμετρικό το συμμετρικό τους κλειδί. Τα προβλήματα αυτά έλυσε η εφεύρεση της ασύμμετρης κρυπτογραφίας. Τέλος σημαντικό είναι να αναφέρουμε ότι αυτά τα κρυπτογραφικά σχήματα λόγω του μεγάλου τους πλεονεκτήματος να έχουν πολύ γρήγορες υλοποιήσεις χρησιμοποιούνται κατά κόρον για μεταφορά μεγάλου όγκου δεδομένων.

2.1.3 Ασύμμετρη Κρυπτογραφία

Η ιστορία της Ασύμμετρης Κρυπτογραφίας αναφέρθηκε την Ενότητα 1. Ο ορισμός των αντίστοιχων κρυπτογραφικών σχημάτων είναι παρόμοιος με αυτόν των συμμετρικών, με την κύρια διαφορά ότι υπάρχει ένα ζεύγος κλειδιών, το δημόσιο και το ιδιωτικό. Ο αυστηρώς ορισμός της παρουσιάζεται παρακάτω :

Definition 2.1.4. Ασυμμετρικό Κρυπτογραφικό Σχήμα (Assymetric Cryptographic Scheme) : Ορίζεται παρόμοια με τον Ορισμό 2.1.3 με μοναδική διαφορά το ζέυγος κλειδιών και την ιδιότητα ορθότητας. Αποτελείτε από ένα σύνολο αλγορίθμων $(\text{KGen}, \text{Enc}, \text{Dec})$ με τις εξής λειτουργίες :

- $(pk, sk) \leftarrow \text{KGen}(1^n)$: Δημιουργεί ένα ζεύγος ασύμμετρων κρυπτογραφικά κλειδιών. Το Δημόσιο Κλειδί (Public Key) pk που χρησιμοποιείται στην κρυπτογράφηση και το αντίστοιχο Ιδιωτικό Κλειδί (Private Key) sk που χρησιμοποιείται στην αποκρυπτογράφηση.
- $c \leftarrow \text{Enc}_{pk}(m)$: Κρυπτογραφεί ένα μήνυμα m χρησιμοποιώντας το δημόσιο κλειδί pk.
- $m \leftarrow \text{Dec}_{sk}(c)$: Αποκρυπτογραφεί ένα κρυπτοκείμενο c χρησιμοποιώντας το ιδιωτικό κλειδί sk.

Όπως και στην περίπτωση της Συμμετρικής Κρυπτογραφίας οι παραπάνω αλγόριθμοι πρέπει να ακολουθούν την εξής ιδιότητα ορθότητας :

$$Pr(\text{Dec}_{sk}(\text{Enc}_{pk}(m))) = Pr(\text{Enc}_s k(\text{Enc}_p km)) = 1$$

Η τελευταία, ιδιότητα δηλαδή ότι $\text{Dec}_p k(m) = \text{Enc}_s^{-1} k(m)$ δίνει και την δυνατότητα για την υποστήριξη ψηφιακών υπογραφών και αποστολής μηνυμάτων με ένα ζεύγος κλειδιών ανά συμμετέχοντα.

Η Ασύμμετρη Κρυπτογραφία λύνει το πρόβλημα της διανομής κρυπτογραφικών κλειδιών που υπάρχει στην περίπτωση της Συμμετρικής, καθότι ο κάθε συμμετέχων χρειάζεται να διαθέτει μόνο ένα ζεύγος κλειδιών, από τα οποία το ένα μπορεί να το μοιραστεί δημόσια. Επίσης, μπορεί να χρησιμοποιηθεί ως συστατικό στοιχείο σε πολλά πρωτόκολλα με σκοπούς πέραν της απλής ανταλλαγής μηνυμάτων και των ψηφιακών υπογραφών, που θα ήταν αδύνατα χωρίς την ασύμμετρη κρυπτογραφία. Ωστόσο, αν και πρόκειται για σύγκριση ανόμοιων πραγμάτων, τα ασύμμετρα κρυπτογραφικά σχήματα είναι πιο κοστοβόρα (αρκετές δεκάδες φορές για μικρό μέγεθος μηνυμάτων και εώς και μια τάξη μεγέθους για μεγάλο μέγεθος μηνυμάτων) σε σχέση με τα συμμετρικά από άποψη υπολογισμών και πόρων στις υλοποιήσεις

τους. Ένα απλό παράδειγμα φαίνεται στο Σχήμα 2.1[1] στο οποίο συγκρίνονται οι χρόνοι (cpu time) κρυπτογράφησης και αποκρυπτογράφησης ενός μικρού μηνύματος για τα σχήματα RSA και AES. Έτσι, σήμερα σεαρκετά δημοφιλή κρυπτογραφικά πρωτόκολλα, όπως για παράδειγμα στο πρωτόκολλο TLS [27], που χρησιμοποιείται για την ασφαλή επικοινωνία στο Επίπεδο Μεταφοράς ενός δικτύου, στην αρχή της συνεδρίας χρησιμοποιούνται ασυμμετρικά σχήματα ψηφιακών υπογραφών και ανταλλαγής κλειδιών, όπως τα ECDSA και ECDHE αντίστοιχα, με σκοπό τη δημιουργία από κοινού συμμετρικών κλειδιών που χρησιμοποιούνται στην συνέχεια από συμμετρικούς αλγορίθμους για την κρυπτογράφηση των δεδομένων της επακόλουθης συνεδρίας. Ακόμα, ένα από τα κύρια χαρακτηριστικά των ασυμμετρικών σχημάτων είναι ότι πιο μαθηματικά δομημένα. Αυτό έχει ως αποτέλεσμα συνήθως να βασίζονται έμμεσα σε πιο πολύπλοκους μαθηματικούς υπολογισμούς, όπως η εύρεση μεγάλων πρώτων αριθμών σε σχέση με τους σχετικά απλούς υπολογισμούς που εκτελεί ένα συμμετρικό σχήμα. Μια ακόμα συνέπεια του χαρακτηριστικού τους ότι είναι πιο μαθηματικά δομημένα, είναι ότι η ασφάλεια τους κατά κόρον βασίζεται σε υποθέσεις πολυπλοκότητας, όπως αυτές που θα αναφερθούν στην Ενότητα 3, από τις οποίες πολλές από αυτές έχει αποδειχθεί ότι δεν ισχύουν σε μη κλασσικά μοντέλα υπολογισμού, όπως το κβαντικό, αλλά και κάνουν την απόδειξη της ασφάλειας τους πιο πολύπλοκη.

2.2 Κρυπτογραφικά Εργαλεία

Στην Ενότητα αυτή θα μελετήσουμε σύγχρονα κρυπτογραφικά εργαλεία που αποτελούν βασικά συστατικά στοιχεία πολλών Κρυπτογραφικών Προτοκόλλων με διάφορες εφαρμογές. Πολλά από αυτά παρατηρούνται συχνά σε πρωτόκολλα SMPC κάτι που θα το παρατηρήσουμε πρακτικά και στο Κεφάλαιο 4.

2.2.1 Σχήματα Δέσμευσης (Commitment Schemes)

Τα Σχήματα Δέσμευσης (Commitment Schemes) είναι το πρώτο εργαλείο που θα εξετάσουμε. Αυτά έχουν τη μορφή πρωτοκόλλων επικοινωνίας δύο φάσεων μεταξύ δύο (ή πιο σπάνια περισσότερων) συμμετεχόντων. Παρέχουν την δυνατότητα σε έναν συμμετέχον, S, να δεσμευτεί κάποια συγκεκριμένη τιμή σε κάποιον άλλον, R, χωρίς όμως αυτός να γνωρίζει ο S την συγκεκριμένη τιμή, δηλαδή η τιμή είναι σε κρυπτογραφημένη μορφή. Το πρωτόκολλο δίνει την δυνατότητα να αποκαλυφθεί στον R η τιμή στην οποία δεσμεύτηκε αρχικά ο V, και μόνο αυτή, σε επόμενη χρονική στιγμή την οποία επιλέγει ο συμμετέχον S. Αυτό αποτρέπει

[1] Είναι σημαντικό αναφέρουμε ότι σχεδόν όλοι οι σύγχρονοι CISC επεξεργαστές διαθέτουν εξειδικευμένα κυκλώματα καθώς και αντίστοιχες εντολές μηχανής και συμβολική γλώσσας για το κρυπτογραφικό σχήμα AES κάτι που σε καμία περίπτωση δεν ισχύει για τον RSA. Το τελευταίο οφείλεται και στο ότι η φύση τελευταίου δεν επιδέχεται μεγάλο βαθμό παραλληλοποίησης.

τον συμμετέχον S στο να μεταβάλλει την τιμή αυτή χωρίς να γίνει αντιληπτό από τον άλλο συμμετέχον. Τα σχήματα αυτά χρησιμοποιούνται ως δομικά στοιχεία σε σχήματα Αποδείξεων Μηδενικής Γνώσης αλλά και στον Ασφαλή Υπολογισμό. Το Σχήμα Δέσμευσης ορίζεται ως εξής :

Definition 2.2.1. Σχήματα Δέσμευσης : Έστω M ο χώρος των μηνυμάτων, C ο χώρος των δεσμευμένων τιμών, D ο χώρος των κλειδιών αποδέσμευσης και R μια ομοιόμορφη πιθανοτική κατανομή που περιέχει αρκετή εντροπία για οποιοδήποτε σχήμα δέσμευσης. Τα σχήματα δέσμευσης είναι πρωτόκολλα δύο συμμετεχόντων, S και R, στα οποία ο S δεσμέυετε μια τιμή στον R. Αποτελούνται από μια τριάδα αλγορίθμων $(\text{Setup}, \text{Commit}, \text{Verify})$ που ορίζονται ως εξής :

- $\text{Setup}(1^n)$:
- $\text{Commit}(m, r)$: Δέχεται ως όρισματα ένα μήνυμα m και μερικά τυχαία νομίσματα r, και δίνει ως εξόδους μια δεσμευμένη τιμή c και ένα κλειδί αποδέσμευσης d για την c.
- $\text{Open}(c, m, d)$: Δέχεται ως όρισματα μια δεσμευμένη τιμή c, το μήνυμα m και το κλειδί ανοίγματος d, και δίνει ως έξοδο μια δυαδική τιμή ανάλογα με το αν η τιμή που φανερώθηκε από την αποδέσμευση της c μέσω της d είναι η m.

Το σχήμα πρέπει να ικανοποιεί τις παρακάτω ιδιότητες :

- **Ορθότητα (Completeness)** : $\forall m \in M, r \leftarrow\$\ R : Pr[\text{Open}(\text{Commit}(m, r), m)] = 1$
- **Απόκρυψη (Hiding)** : $\forall$ distinct $m, m' \in M, r \leftarrow\$\ \text{R}, \forall A$:
  $|Pr[A(\text{Commit}(m, r))] - Pr[A(\text{Open(r, m')})]| \leq \varepsilon_h$
- **Δέσμευση (Binding)** : $\forall$ distinct m, m', $\forall$ distinct d, d', $\forall c$:
  $|Pr[\text{Open}(c, m, d)] - Pr[\text{Open}(c, m', d')]| \leq \varepsilon_b$

Οι τιμές ε_h και ε_b εξαρτόνται από το είδος της ιδιότητας που θέλουμε να έχουμε. Σε αντιστοιχία με τους ορισμούς ασφάλειας αλλά και με την ισχύ του αντιπάλου που θα ορίσουμε στο επόμενο κεφάλαιο, μπορούμε να έχουμε **Υπολογιστική**, **Στατιστική** ή **Τέλεια Απόκρυψη** και **Δέσμευση** και οι τιμές που λαμβάνουν οι ε_h και ε_b είναι 0, $negl(1^n)$ και $negl(1^n)$ αντίστοιχα. Τέλος, μπορεί να αποδειχθεί σχετικά εύκολα, ότι δεν μπορούμε να έχουμε σχήματα που να διαθέτουν ταυτόχρονα τέλεια απόκρυψη και τέλεια δέσμευση.

2.2.2 Αποδείξεις Μηδενικής Γνώσης

Το επόμενο εργαλείο που θα εξετάσουμε είναι αυτό τον Αποδείξεων Μηδενικής Γνώσης. Η Μηδενική Γνώση είναι μια ιδιότητα που μπορεί να αποκτήσει η κλάση IP των Διαδραστικών Αποδείξεων αν υποθέσουμε ότι υπάρχουν Συναρτήσεις Μονής Διαδρομής (One Way

Functions ή OWF), μια υπόθεση που θα δούμε στο επόμενο κεφάλαιο. Στο Κεφάλαιο 8 κάνουμε μια μικρή μελέτη στις διαδραστικές αποδείξεις που είναι άμεσα συσχετισμένες με τις Αποδείξεις Μηδενικής Γνώσης. Διαισθητικά σε ένα πρωτόκολλο με την ιδιότητα αυτή ο Αποδεικνύων P, μπορεί να αποδείξει στον Επαληθευτή V ότι γνωρίζει κάποιον μάρτυρα w που ανήκει σε μια προκαθορισμένη γλώσσα L που γνωρίζουν και οι δύο οντότητες, χωρίς ο V να μπορεί να αποκτήσει γνώση σχετικά με τον w. Μια πιο παραστατική αντίληψη του πως λειτουργούν οι αποδείξεις με την ιδιότητα αυτή παρουσιάζεται στο Σχήμα 2.2. Σχετικά με τον αυστηρό ορισμό της ιδιότητας της Μηδενικής Γνώσης, χρειάστηκαν αρκετά χρόνια ώστε να υπάρξει ένας καταληκτικός ορισμός στη βιβλιογραφίας. Αυτός παρουσιάζεται παρακάτω σε αντιστοιχία με τον ορισμό της κλάσης IP:

Definition 2.2.2. Κλάση Αποδείξεων Μηδενικής Γνώσης (Computational Zero Knowledge Proofs ή CZK) : Έστω Pντετερμινιστική συναρτήση TM που είναι απεριόριστης ισχύος. Η γλώσσα L λέμε ότι ανήκει στην κλάση IP, δηλαδή $L \in IP$ αν υπάρχει μη ντετερμινιστική TM V με $x, r, a_1, ..., a_i, V$ και χρονική πολυπλοκότητα σε $O(poly(|x|))$ που ικανοποιεί τις παρακάτω ιδιότητες :s

- **Πληρότητα (Completeness)** : $x \in L \Rightarrow \exists \Pr[\text{ out }_V\langle V, P\rangle(x) = 1] \geq (\varepsilon =)\frac{2}{3}$
- **Ορθότητα (Soundness)** : $x \notin L \Rightarrow \forall \Pr[\text{ out }_V\langle V, P\rangle(x) = 1] \leq 1 - \varepsilon$
- **Μηδενική Γνώση** : $\forall V, \exists S \in PPT : \forall x \in L \wedge w \in R_L$:

$$View_V = S(x, z)$$

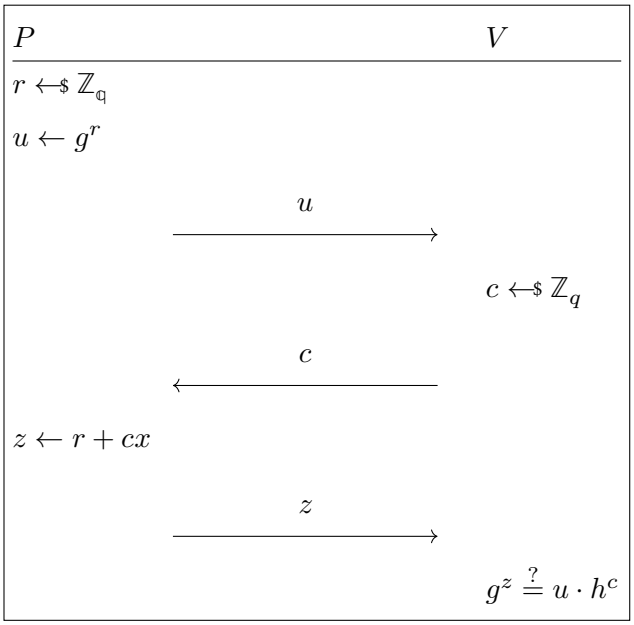
Προφανώς όπως ισχύει και για τις ιδιότητες της Πληρότητας και της Ορθότητας των Διαδραστικών Αποδείξεων έτσι και για την ιδιότητα της Μηδενικής Γνώσης μπορούμε να έχουμε από τους αντίστοιχους ορισμούς ασφάλειας που θα παρουσιαστούν στο επόμενο κεφάλαιο, **Τέλεια**, **Στατιστική** ή **Υπολογιστική Μηδενική Γνώση** ανάλογα με τον τύπο της δυσδιακριτότητας των κατανομών της μηδενικής γνώσης και οι κλάσεις που ορίζουν είναι οι PZK, SZK και CZK αντίστοιχα. Για τις κλάσεις αυτές γνωρίζουμε ότι PZK $\subseteq$ SZK $\subseteq$ CZK. Η ευρύτερη από αυτές τις κλάσεις είναι αυτή της CZK που παρουσιάστηκε και στον Ορισμό 2.2.2. Για την κλάση CZK γνωρίζουμε ότι IP $=$ CZK $=$ PSPACE και ότι NP $\subseteq$ CZK, προφανώς υπό την υπόθεση OWF [29]. Για τις κλάσεις SZK και PZK ωστόσο δεν έχει αποδειχθεί κάποια παρόμοια ισότητα αλλά γνωρίζουμε το εξής PZK $\subseteq$ SZK $\subseteq$ AM $\cap$ co-AM και επίσης εικάζεται ότι δεν ισχύει $NP \subseteq$ SZK καθώς τότε καταρρέει όλη η πολυωνυμική ιεραρχία της NP. Μια σύνοψη της γνώσης που έχουμε προς το παρόν για τις τρεις προηγούμενες κλάσεις, συμπεριλαμβανομένων και αρκετών σχέσεων που παραλήψαμε για λόγους συνοπτικότητας, μπορεί να βρεθεί στην εργασία [30]. Δεν μπορούμε να παραλείψουμε να αναφέρουμε δύο

σημαντικές κλάσεις που σχετίζονται άμεσα με την CZK, αυτή των Μη Διαδραστικών Αποδείξεων Μηδενικής Γνώσης, $NICZK$, και αυτή των Πιθανοτηκά Ελεγχόμενων Αποδείξεων PCP, ωστόσο η μελέτη τους εμπείπτει εκτός τον σκοπώς αυτής της εργασίας.

Παρακάτω παρουσιάζεται ένα από τα πιο γνωστά σχήματα μηδενικής γνώσης, το Πρωτόκολλο Schnorr που μπορεί να χρησιμοποιηθεί για να αποδειχθεί η γνώση μιας λύσης ενός στιγμιοτύπου του προβλήματος του Διακριτού Λογαρίθμου, δηλαδή ενός στοιχείου x τέτοιου ώστε $h = g^x$.

Definition 2.2.3. Πρωτόκολλο Schnorr : Απόδειξη Μηδενικής Γνώσης για τον Διακριτό Λογάριθμο [31] : Έστω μια πολλαπλασιαστική ομάδα G, πρώτης τάξης με $|G| = q$.

- **Εισόδοι** :
  - P: x, h
  - V: h
- **Πρωτόκολλο** :



- **Εξόδοι** :
  - P: $\perp$
  - V: 0 ή 1 ανάλογα με το αν ισχύει η τελευταία πράξη ισότητας που εκτέλεσαι

2.2.3 Σχήματα Ανυποψίαστης Μεταφοράς

Τα σχήματα Ανυποψίαστης Μεταφοράς (Oblivious Transfer ή OT) είναι το επόμενο κρυπτογραφικό εργαλείο που θα μελετήσουμε.Πρόκειται πρωτόκολλα δύο συμμετεχόντων, P_1, P_2, στα οποία ο ένας συμμετέχον έχει n τιμές και από τις οποίες ο άλλος συμμετέχον μπορεί να επιλέξει $t \leq n$ από αυτές δίχως ο πρώτος συμμετέχον να γνωρίζει ποιες επέλεξε ο δεύτερος. Ίσως να ακούγεται ανόητη η χρησιμότητα τους όμως έχει αποδειχθεί ότι έχουν πολύ μεγάλη σημασία και δίχως σχήματα OT πολλά σύνθετα δεν μπορούν να υλοποιηθούν. Για να αντιληφθούμε την σημασία τους, έχει αποδειχθεί πως με βάση μόνο σχήματα OT μπορεί να υπολογιστεί οποιαδήποτε SMPC συνάρτηση [32]. Το πρώτο πρωτόκολλο Ανυποψίαστης Μεταφοράς παρουσιάστηκε στην βιβλιογραφία το 1981 από τον Rabin, στην εργασία [33]. Μπορούμε να μοντελοποιήσουμε τα σχήματα αυτά ως εξης :

Definition 2.2.4. Πρωτόκολλο Ανυποψίαστης Μεταφοράς t από τα n (Oblivious Transfer ή OT t-out-of-n) :

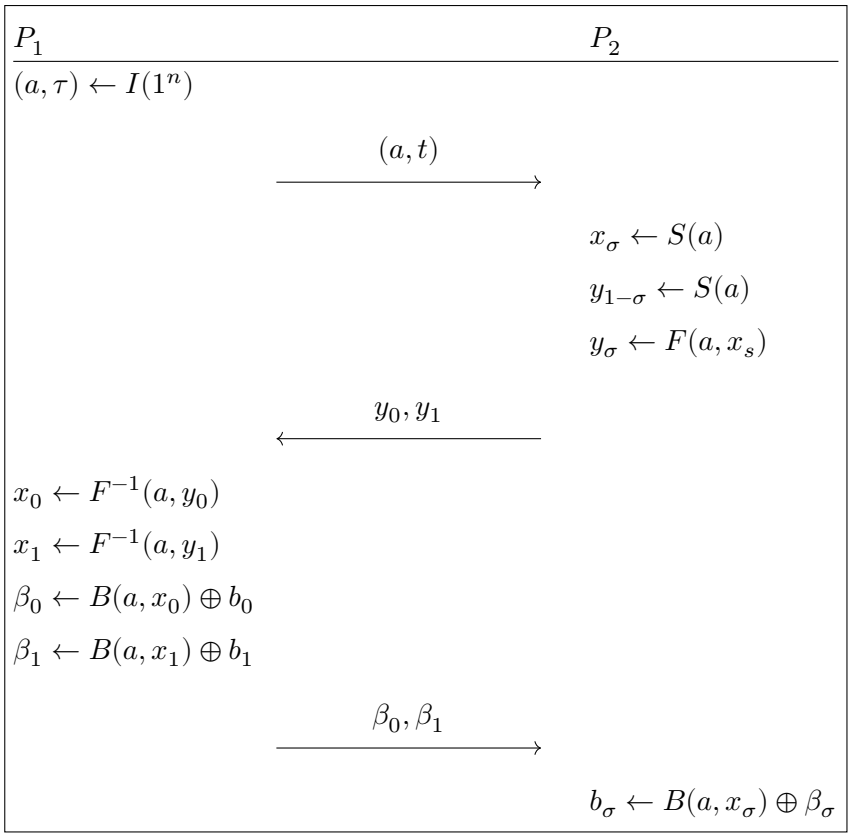
- **Εισόδοι** :
  - $P_1 : b_0, b_1, ... b_n \in \{0,1\}^l$
  - $P_2 : t_0, t_1, ... t_n \in \{0,1\}$
- **Εξόδοι** :
  - $P_1 : \perp$
  - $P_2 : \{\forall t_n : b_{\{t_n=1\}}\}$

Πολλές φορές στην βιβλιογραφία χρησιμοποιείται ο συμβολισμός OT t/n ή OT t-n, δηλαδή για παράδειγμα OT 1-2 ή OT 1/2. Παρακάτω παρουσιάζεται ένα πρωτόκολλο για OT 1-2 που είναι Σημασιολογικά Ασφαλές απέναντη σε Παθητικούς Αντιπάλους και παρουσιάστηκε στην εργασία [34].

Definition 2.2.5. Πρωτόκολλο Ανυποψίαστης Μεταφοράς 1/2 (Oblivious Transfer 1/2) [34] :

- **Εισόδοι** :
  - Κοινή είσοδος : Οικογένεια Trapdoor Permutations (TDP) : (I, S, F, F^{-1}), Hardcore Predicate : B
  - $P_1 : b_0, b_1 \in \{0,1\}$
  - $P_2 : \sigma \in \{0,1\}$.

- **Πρωτόκολλο** :



- **Εξόδοι** :
  - $P_1 : \perp$
  - $P_2 : b_\sigma$

2.2.4 Σχήματα Διαμοίρασης Μυστικών (Secret Sharing)

Το τελευταίο κρυπτογραφικό εργαλείο που θα μελετήσουμε είναι αυτό των σχημάτων Διαμοίρασης Μυστικών. Για να αντιληφθούμε τη χρησιμότητα των σχημάτων Διαμοίρασης Μυστικών ας εξετάσουμε πρώτα ένα απλό παράδειγμα. Ας υποθέσουμε ότι ο ιδιοκτήτης μιας επιχείρησης έχει στην κατοχή του κάποια πολύ σημαντικά έγγραφα τα οποία βρίσκονται σε κάποια θυρίδα που είναι ασφαλισμένη με κάποιο κωδικό. Σε αυτά τα έγγραφα θέλει να έχει πρόσβαση το Διοικητικό Συμβούλιο της εταιρίας ακόμα και στην περίπτωση που αυτός δεν είναι παρόν, π.χ. στην περίπτωση που αυτός ξαφνικά φύγει από την ζωή, υπό την προϋπόθεση όμως ότι η πλεοψηφία, ή κάποιο συγκεκριμένο ποσοστό που έχει επιλέξει αυτός,των συμβούλων συμφωνεί στο άνοιγμα της θυρίδας.

Το πρώτο κρυπτογραφικό σχήμα που λύνει το πρόβλημα αυτό παρουσιάστηκε από τον Shamir στην εργασία [35] το 1979 και σήμερα απαντάται στην βιβλιογραφία ως Πρωτόκολλο Διαμοίρασης Μυστικών Shamir. Βασίζεται στο γεγονός ότι οποιοδήποτε σύνολο $n+1$ σημείων σε ένα δυδιάστατο πεδίο $\mathbf{F}^2$ καθορίζει μοναδικά ένα πολυώνυμο n βαθμού. Δηλαδή αν $f : \mathbb{F} \to \mathbb{F}$ όπου $f(x) = \alpha_n x^n + \cdots + \alpha_1 x + \alpha_0$ τότε ένα σύνολο σημείων $(x_i, f(x_i))$ όπου $x \in [0, n]$ είναι επαρκές για να ανακτήσουμε την συνάρτηση f. Αυτό μπορεί πολύ εύκολα να συμβεί λύνοντας το σύστημα εξισώσεων μεγέθους $(n+1)\cdot(n+1)$ που προκύπτει αν αντικαταστήσουμε με κάθε σημείο την γενική εξίσωση πολυωνύμου $f(x_i) = a_n x_i^n + \cdots + a_1 x_i + a_0$. Ένας πιο εύκολος τρόπος να επιτευχθεί αυτό είναι μέσω των πολλαπλασιαστών Lagrange που αναλύθηκαν στην Ενότητα 8. Παρακάτω παρουσιάζονται οι δύο αλγόριθμοι του Πρωτοκόλλου Διαμοίρασης Μυστικών του Shamir, ένας για την διαμοίραση του μυστικού και ένας για την ανακατασκεύη του.

Definition 2.2.6. Σχήμα Διαμοίρασης Μυστικών του Shamir (Shamir's Secret Sharing ή SSS) : Έστω ένα πεδίο $\mathbb{F}$, n παίκτες $P_0, P_1, P_2, \ldots, P_n$ που συμμετέχουν το πρωτόκολλο και μια τιμή $s \in \mathbb{F}$ που θα θεωρήσουμε ότι είναι το μυστικό που θέλει να διαμοιράσει ο παίκτης P_0, τον οποίο αποκαλούμε κάτοχο του μυστικού. Το πρωτόκολλο αυτό χρησιμοποιείται για την διαμοίραση μυστικού σε n συμμετέχοντες με τρόπο έτσι ώστε να χρειάζεται τουλάχιστον t, όπου $t \leq n$, να συνεργαστούν ώστε να γίνει η αποκάλυψη του μυστικού.

- **Πρωτόκολλο Διαμοίρασης** : Το πρωτόκολλο αυτό κτελείται μεταξύ όλων των συμμετεχόντων.
  - **Εισόδοι** :
    - Κοινή είσοδος : Πεδίο $\mathbb{F}$
    - Κάτοχος του μυστικού P_0 : Μυστικό $s \in \mathbb{F}$
  - **Πρωτόκολλο** :

P_0	P_i
$f(x) \leftarrow\$\ \mathbb{F}[x] : deg(f_i(x)) = t+1 \wedge f_i(0) = s$	
$\forall i \in \{1, 2, ...\} : p_i \leftarrow\$\ f(x)_{x \neq 0}$	
	$\xrightarrow[\text{to player } P_i]{p_i}$

  - **Εξόδοι** :
    - $P_i : p_i$ (**Μετοχή του συμμετέχοντα** i)

Το πρωτόκολλο ανακατασκευής εκτελείται μόνο μεταξύ των συμμετεχόντων που προσπαθούν να ανακατασκευάσουν το μυστικό s. Υποθέτουμε ότι μέσω κάποιου καναλιού επικοινωνίας τουλάχιστον $t+1$ από τους συμμετέχοντες συμμετέχοντες έχουν ανταλλάξει τα στοιχεία τους p_i μεταξύ τους.

- **Πρωτόκολλο Ανακατασκευής** :
  - **Εισόδοι** :
    - Κοινή Είσοδος (των συμμετεχόντων που εκτελούν την ανακατασκευή): $S = \{\text{Σύνολο μετοχών } p_i\}$, όπου $t \leq |S| \leq n$
  - **Πρωτόκολλο** :

P_i
$L(x) := \sum_{p_i \in S} p_i l_j(x)$, όπου $l_j(x)$ είναι οι συντελεστές Lagrange
$s \leftarrow L(0)$

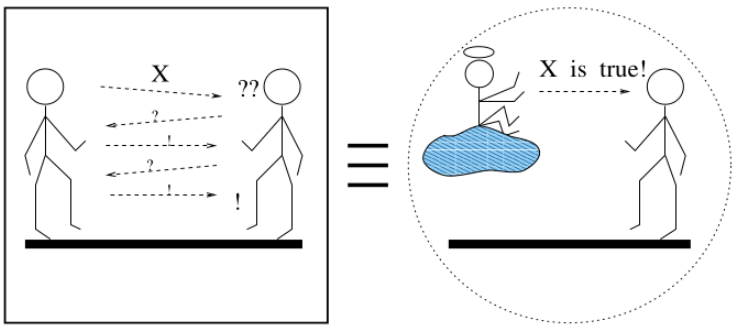
  - **Εξόδοι** :
    - Κοινή Έξοδος : s

Αξίζει να σημειωθεί ότι στην βιβλιογραφία έχουν προταθεί αρκετά πρωτόκολλα για Διαμοίραση Μυστικών. Όπως του Blakley [36], του Mignotte και των Asmut-Bloom. Το πρώτο βασίζεται στο γεγονός ότι αν πάρουμε n n-διάστατα υπερεπιπέδα με την προϋπόθεση ότι ανά δύο δεν είναι παράλληλα τότε όλα έχουν ένα κοινό μονοδιάστατο κοινό σημείο τομής. Έτσι αν θέλουμε να πάρουμε ένα σχήμα Διαμοίρασης Μυστικών με n συμμετέχοντες και κατόφλι t, τότε αρκεί να δώσουμε ένα υπερεπίπεδο t-οστού βαθμού σε κάθε συμμετέχοντα, δηλαδή συνολικά n υπερεπίπεδα υπό τον περιορισμό ότι όλα τα υπερεπίπεδα τέμνονται σε ένα σημείο το οποίο το επιλέγουμε εξ'αρχής και είναι ο μυστικό μας. Τέλος μια πολύ χρήσιμη ιδιότητα του σχήματος SSS αλλά και αυτού του Blakley είναι ότι διαθέτουν τέλεια ασφάλεια, που εξηγούμε τι σημαίνει στο επόμενο κεφάλαιο.

```
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP
from Crypto.Util.Padding import pad, unpad

from time import process_time


data = "CRYPTOGRAPHY".encode("utf-8")

key = RSA.generate(2048)
private_key = key
public_key = key.publickey()

t1_start = process_time()

enc_rsa = PKCS1_OAEP.new(public_key)
dec_rsa = PKCS1_OAEP.new(private_key)
enc_data = enc_rsa.encrypt(data)
dec_data = dec_rsa.decrypt(enc_data)

t1_stop = process_time()

assert(dec_data == data)
print("Elapsed time for RSA encryption and decryption in milliseconds:",
                                "{:.2f}".format((t1_stop-t1_start) * 1000))

key = get_random_bytes(16)

t1_start = process_time()

cipher_aes = AES.new(key, AES.MODE_ECB)
enc_data = cipher_aes.encrypt(pad(data, 32))
dec_data = unpad(cipher_aes.decrypt(enc_data), 32)

t1_stop = process_time()

assert(dec_data == data)
print("Elapsed time for AES encryption and decryption in milliseconds:",
                                "{:.2f}".format((t1_stop-t1_start) * 1000))
```

```
Elapsed time for RSA encryption and decryption in milliseconds: 2.28
Elapsed time for AES encryption and decryption in milliseconds: 0.17
```

Σχήμα 2.1: Μικρή σύγκριση των επιδόσεων για την κρυπτογράφηση και την αποκρυπτογράφηση ενός απλού μηνύματος μεταξύ συμμετρικών και ασύμμετρων κρυπτογραφικών σχημάτων (AES, RSA).



Σχήμα 2.2: Κόμικ που αναπαρηστά πως λειτουργούν οι Αποδείξεις Μηδενικής Γνώσης [28].

Κεφάλαιο 3

Ασφάλεια

Η κρυπτογραφία αναπτύσσεται επιδιώκοντας την ασφάλεια. Η επιφάνεια του αντικειμένου της ασφάλειας ενός κρυπτογραφικού σχήματος είναι πολύ ευρεία, ξεκινάει από την στιγμή που σχεδιάζεται αυτό και τελειώνει, ίσως, όταν αυτό αρχίσει να χρησιμοποιείται στην πράξη. Στο κεφάλαιο αυτό θα μελετήσουμε, από τις μόνες μορφές αποδείξιμης ασφάλειας, την θεωρητική ασφάλεια ενός σχήματος. Επειδή, στα πλαίσια αυτής της εργασίας πραγματευόμαστε αποκλειστικά την θεωρητική ασφάλεια, θα αναφερόμαστε σε αυτήν απλά ως ασφάλεια. Προφανώς, ακόμα και η έννοια της θεωρητικής ασφάλειας ίσως να ακούγεται πολύ γενική. Παρατηρούμε ότι ανάλογα με την κατηγορία του σχήματος ή του εκάστοτε αλγορίθμου που εξετάζουμε επιθυμούμε να έχει και διαφορετικές ιδιότητες. Για παράδειγμα, με διαφορετικό τρόπο ορίζουμε την ασφάλεια για ένα SMPC πρωτόκολλο και με διαφορετικό για ένα σχήμα δημόσιου κλειδιού, αφού διαθέτουν ποιοτική διαφορά. Επίσης, αν λάβουμε υπόψιν και την ισχυρότατα της ασφάλειας που επιθυμούμε να αποδείξουμε, μοντελοποιούμε και τον αντίπαλο του σχήματος με διαφορετικό τρόπο και άρα προκύπτει και μια ποσοτική διαφορά των ιδιοτήτων ασφάλειας. Αυτό συνήθως συμπεριλαμβάνει τους υπολογιστικούς πόρους και τις προθέσεις (π.χ. να γίνει αντιληπτός) που μπορεί να έχει κάποιος επιτιθέμενος για να σπάσει την ιδιότητα της ασφάλειας αλλά και το κέρδος που μπορεί να έχει από αυτό. Έτσι, για παράδειγμα, διαφορετικό τύπο αντιπάλων θα υποθέταμε στην μοντελοποίηση της ασφάλειας ενός κρυπτοσυστήματος ανταλλαγής κρατικών μυστικών και διαφορετικό τύπου αντιπάλων για ένα ανταλλαγής δεδομένων μεταξύ αισθητήρων που μετράνε το οξυγόνο σε ένα δάσος, υποθέτοντας ότι η πρόσβαση σε αυτά τα δεδομένα δεν δίνει κάποιο ιδιαίτερο κέρδος σε έναν αντίπαλο. Έτσι, ένα τυπικό παράδειγμα ισχυρισμού για την ασφάλεια ενός κρυπτοσυστήματος είναι το εξής : "Σημασιολογικά ασφαλές για υπολογιστικά περιορισμένους αντιπάλους". Στο παράδειγμα αυτό, "Σημασιολογικά ασφαλές" είναι η ιδιότητα που επιθυμούμε να έχει το κρυπτοσύστημα μας και "Υπολογιστικά περιορισμένοι αντίπαλοι", είναι ως προς ποιο μοντέλο αντιπάλων επιθυμούμε να ισχύει αυτή η ιδιότητα. Όμως δεν αρκεί απλά να ισχυριστούμε ότι ένας αλγόριθμος είναι ασφαλείς πρέπει και να το αποδείξουμε. Στην

συνέχεια του κεφαλαίου θα παρουσιαστούν και θα μελετηθούν βασικοί ορισμοί μοντέλων αντιπάλων, βασικοί ορισμοί της ασφάλειας, τυπικών ισχυρισμών αυτής, μεθόδων απόδειξης και βασικών μοντέλων/υποθέσεων στα οποία ανάγουμε αυτές. Πριν ξεκινήσουμε την μελέτη μας πρέπει πρώτα να δώσουμε έναν γενικό ορισμό ενός αντιπάλου τον οποίο θα κάνουμε πιο συγκεκριμένο στην συνέχεια του κεφαλαίου. Αυτός είναι ο εξής :

Definition 3.0.1. Αντίπαλος κρυπτογραφικού σχήματος : Είναι μια οντότητα ο οποία μπορεί να "χειρίζεται" έναν ή και περισσότερους συμμετέχοντες (ανάλογα με το πόσοι συμμετέχουν στο κρυπτογραφικό σχήμα) σε σχήμα, δηλαδή μπορεί να διαφθείρει έναν ή περισσότερους συμμετέχοντες με σκοπό να λειτουργήσουν σύμφωνα με το δικό του σκοπό. Στην πράξη μοντελοποιείται με έναν αλγόριθμο ή που ελέγχει όλους τους διεφθαρμένους συμμετέχοντες. Προφανώς, δεν έχει νόημα να θεωρήσουμε έναν αντίπαλο που ελέγχει όλους τους συμμετέχοντες ενός σχήματος.

Στο σημείο αυτό πρέπει να αναφέρουμε ότι για πολλούς από τους ορισμούς που θα μελετήσουμε έχουν χρησιμοποιηθεί ως βιβλιογραφικές πηγές οι [4], [24], [32].

3.1 Βασικοί Ορισμοί

Στην ενότητα αυτή θα γίνει μια εισαγωγή στην βασική σημειολογία της θεωρητικής ασφάλειας, στα βασικά μοντέλα ασφάλειας και στις βασικές κρυπτογραφικές υποθέσεις που χρησιμοποιούμε για να καταφέρουμε να αποδείξουμε την ασφάλεια ενός κρυπτογραφικού σχήματος.

3.1.1 Βασική Σημειολογία

Θα ξεκινήσουμε με μια εισαγωγή στην σημειολογία που χρησιμοποιείται στην θεωρητική ασφάλεια. Δεν μπορούμε να ξεκινήσουμε παρά ορίζοντας μια συνάρτηση που είναι πανταχού παρόν στην κρυπτογραφία και κυρίως στις αποδείξεις υπολογιστικής ασφάλειας. Αυτή της μηδαμινής συνάρτησης, που ορίζεται ως εξής :

Definition 3.1.1. Μηδαμινή συνάρτηση (Negligible function) : Κάθε συνάρτηση $negl$: $\mathbb{N} \to \mathbb{R}$ που συναρτήσει της εισόδου της N, τείνει ως προς το 0 ασυμπτωτικά γρηγορότερα από κάθε αντίστροφη θετική πολυωνυμική συνάρτηση.

Συνήθως στις αποδείξεις ασφάλειας η μηδαμινή συνάρτηση παίρνει ως είσοδο κάποια παράμετρο ασφάλειας, οπότε στη βιβλιογραφία πολύ συχνά συναντάμε συμβολισμούς όπως ο $negl(1^n)$. Πολύ βασική είναι επίσης και η έννοια της δυσδιακριτότητας μεταξύ δύο κατανομών, διακρίνουμε δύο τύπους δυσδιακριτότητας ανάλογα με τους υπολογιστικούς πόρους το αντιπάλου, την υπολογιστική και την στατιστική, που ορίζονται ως εξής :

Έστω ότι έχουμε δύο πιθανοτικούς αλγορίθμους, που η έξοδος του ακολουθεί κατανομή D_1 και D_2 αντίστοιχα.

Definition 3.1.2. Στατιστική δυσδιακριτότητα (Statistical Indistinguishability) : Οι κατανομές D_1 και D_2 έχουν Στατιστική δυσδιακριτότητα (Indistinguishable distributions), αν για οποιονδήποτε αλγόριθμο (ακόμα και με απεριόριστους υπολογιστικούς πόρους) A ισχύει:

$$\Pr\left[A(D_1(n)) = 1\right] - \Pr\left[A(D_2(n)) = 1\right] \leq negl(1^n)$$

όπου Δ είναι η στατιστική απόσταση των δύο κατανομών, όπου n μια παράμετρος ασφάλειας.

Definition 3.1.3. Υπολογιστική Δυσδιακριτότητα (Computational Indistinguishability) : Οι κατανομές D_1 και D_2 έχουν Υπολογιστική δυσδιακριτότητα (Computational indistinguishability), αν για οποιονδήποτε μη ομοιόμορφο πιθανοτικό πολυωνυμικό αλγόριθμο (non-uniform PPT) A ισχύει:

$$\Pr\left[A(D_1(n)) = 1\right] - \Pr\left[A(D_2(n)) = 1\right] \leq negl(1^n)$$

όπου n μια παράμετρος ασφάλειας.

Οι έννοιες τις δυσδιακριτότητας είναι ιδιαίτερα χρήσιμες στις αποδείξεις ασφάλειας καθώς ο στόχος κάθε απόδειξης είναι να δείξουμε ότι η κατανομή των δεδομένων που γνωρίζει ένας αντίπαλος μέσω κάποιου κρυπτογραφικού σχήματος είναι δυσδιάκριτη από κάποια τυχαία κατανομή. Οι παράμετροι ασφάλειας (security parameters) εμφανίζονται στις αποδείξεις ασφάλειας κρυπτογραφικών σχημάτων ως ένας τρόπος ποσοτικοποίησης της ασφάλειας, δηλαδή της δυσκολίας του αντιπάλου να σπάσει την ασφάλεια τους. Ουσιαστικά οι παραμέτροι ασφάλειας είναι ένα μέτρο ποσοτικοποίησης του πλεονεκτήματος του αντιπάλου σε ένα σύστημα. Όπως θα δούμε και στη συνέχεια, είναι άμεσα συσχετισμένο με τις έννοιες τις δυσδιακριτότητας. Στην βιβλιογραφία απαντάμε δύο κύριες παραμέτρους ασφάλειας, αυτή της στατιστικής και αυτή της υπολογιστικής.

Η παράμετρος της υπολογιστικής ασφάλειας προφανώς σχετίζεται με την Υπολογιστική Ασφάλεια και άρα με την ποσοτικοποίηση της δυσκολία επίλυσης κάποιου υπολογιστικού προβλήματος από έναν υπολογιστικά περιορισμένο αντίπαλο. Ας δούμε ένα πρακτικό παράδειγμα. Σε ένα πρωτόκολλο RSA για παράδειγμα η παράμετρος υπολογιστικής ασφάλειας είναι είσοδος της συνάρτησης KGen, αφού πρόκειται και πρακτικά είναι το μήκος n σε bit των πρώτων αριθμών p,q που επιλέγουμε τέτοιοι ώστε $N = p \cdot q$. Είναι σύνηθες να απαντάμε αυτή την παράμετρο ως το μήκος κλειδιού που χρησιμοποιείται σε ένα σχήμα. Την ορίζουμε ως εξής :

Definition 3.1.4. Παράμετρος υπολογιστικής ασφάλειας (Computational security parameter) n : Είναι η παράμετρος που σχετίζεται με την δυσκολία σπασίματος προβλημάτων

από τον αντίπαλο μέσω της υπολογιστικής ισχύς του. Όταν δίνεται ως όρισμα σε μια συνάρτηση συμβολίζεται με τον μοναδιαίο συμβολισμό 1^n.

Αντίστοιχα, η παράμετρος της στατιστικής ασφάλειας σχετίζεται με την Στατιστική Ασφάλεια, δηλαδή με την ποσοτικοποίηση της δυσκολίας επίλυσης κάποιου προβλήματος, με την γενικότερη έννοια, που δεν σχετίζεται με την υπολογιστική ισχύ του αντιπάλου, αφού στην περίπτωση της Στατιστικής Ασφάλειας υποθέτουμε ότι ο αντίπαλος διαθέτει άπειρους υπολογιστικούς πόρους. Για παράδειγμα, σε ένα διαδραστικό πρωτόκολλο ο αντίπαλος μπορεί να διαθέτει μια μοναδική ευκαιρία να παραβιάσει την ασφάλεια του πρωτοκόλλου αν καταφέρει να προβλέψει μια τυχαία τιμή που θα επιλέξει ο κάποιος συμμετέχον στον επόμενο γύρο. Είναι προφανές ότι πρόβλεψη αυτής της τιμής είναι ανεξάρτητη από τους υπολογιστικούς πόρους του αντιπάλου. Την πιθανότητα να προβλέψει αυτή την τιμή την ποσοτικοποιούμε με την παράμετρο αυτή. Η παράμετρος ορίζεται ως εξής :

Definition 3.1.5. Παράμετρος στατιστικής ασφάλειας (Statistical security parameter) σ : Είναι η παράμετρος που σχετίζεται με την δυσκολία σπασίματος προβλημάτων που δεν σχετίζονται με την υπολογιστική του ισχύ αντιπάλου. Όταν δίνεται ως όρισμα σε μια συνάρτηση συμβολίζεται επίσης με τον μοναδιαίο συμβολισμό 1^σ.

Πρέπει να αναφέρουμε πως εκ των ονομάτων των παραμέτρων αυτών, είναι απολύτως λογικό να περίμενε κάποιος σε ένα κρυπτογραφικό σχήμα να υπάρχει μια παράμετρος ασφαλείας η οποία ανάλογα με το τι είδος ασφάλειας θέλουμε να αποδείξουμε να ονομάζεται στατιστική ή υπολογιστική παράμετρος ασφάλειας αντίστοιχα. Η αλήθεια είναι ότι σε πιο σύνθετα σχήματα, όπως για παράδειγμα το πρωτόκολλο Cut-and-Choose που θα δούμε στο Κεφάλαιο 4, μπορούν να περιέχουν και τις δύο παραμέτρους ασφάλειας. Στην περίπτωση αυτή ο σωστός τρόπος να ερμηνεύσουμε την ασφάλεια του συστήματος είναι ως : $2^{-\sigma} + negl(1^n)$. Είναι προφανές ότι στην περίπτωση που υπάρχουν και οι δύο παράμετροι ασφαλείας το μέγιστο επίπεδο ασφάλειας που μπορεί να επιτύχει ένα σχήμα είναι αυτό της Υπολογιστικής Ασφάλειας.

3.1.2 Βασικά Κρυπτογραφικά Μοντέλα

Θα συνεχίσουμε με τα βασικά κρυπτογραφικά μοντέλα ή αφαιρέσεις που χρησιμοποιούνται στην θεωρητική ασφάλεια. Η απόδειξη της θεωρητικής ασφάλειας ενός κρυπτογραφικού σχήματος δεν είναι καθόλου εύκολη υπόθεση. Πολλές φορές για να γίνει εφικτή η απόδειξη αυτή χρειάζεται να μοντελοποιήσουμε ή να εξιδανικεύσουμε αρκετά στοιχεία του σχήματος που εξετάζουμε, όπως για παράδειγμα στοιχεία που σχετίζονται με την υλοποίηση ή στοιχεία που σχετίζονται με κρυπτογραφικά εργαλεία που χρησιμοποιεί το σχήμα τα οποία τα θεωρούμε ως ιδανικά ώστε να διαχωρίσουμε τις αποδείξεις ασφάλειας του. Για

παράδειγμα, μια συνάρτηση κατακερματισμού μπορούμε να την μοντελοποιήσουμε ως ένα Τυχαίο Μαντείο όταν χρησιμοποιείται ως συστατικό στοιχεία σε ένα σχήμα. Η εξέταση του κατά πόσο η συνάρτηση αυτή προσεγγίζει το μοντέλο αυτό αποτελεί αντικείμενο κάποιας άλλης απόδειξης. Ας ξεκινήσουμε με το πιο απλό αλλά και ταυτόχρονα το πιο ισχυρό μοντέλο ασφάλειας. Στο μοντέλο αυτό θεωρείται εξαιρετικά δύσκολο να αποδειχθεί κάποιο σχήμα ότι είναι ασφαλές.

Definition 3.1.6. Κανονικό Μοντέλο (Standard Model ή SM) : Στο μοντέλο αυτό ο μόνος περιορισμός ενός επιτιθέμενου είναι ότι διαθέτει υπολογιστικά περιορισμένους πόρους και ότι ισχύουν οι συνήθεις εικασίες πολυπλοκότητας συγκεκριμένων προβλημάτων και κλάσεων προβλημάτων, όπως για παράδειγμα για το DLP ή για την παραγοντοποίηση σε πρώτους παράγοντες.

Στο μοντέλο αυτό, δεν είναι καθόλου πρακτικό καθώς κάνει τις ελάχιστες δυνατές υποθέσεις. Πολλά πρωτόκολλα που έχουν προταθεί ως ασφαλή στο μοντέλο αυτό εν τέλει έχουν απορριφθεί λόγω του ότι βρέθηκαν λάθη στην απόδειξη της ασφάλειας τους.

Το επόμενο μοντέλο που θα μελετήσουμε κάνει μια σχετικά απλή υπόθεση, μια εξιδανίκευση της υλοποίησης. Ας το δούμε πρώτα μέσα από ένα παράδειγμα. Ένα κρυπτογραφικό σχήμα, όπως το Diffie-Hellman γνωρίζουμε ότι χρησιμοποιεί μια πεπερασμένη πολλαπλασιαστική ομάδα μεγάλης πρώτης τάξης. Η υλοποιητική αναπαράσταση της ομάδας αυτής μπορεί να φανερώνει πληροφορίες σχετικά με την εσωτερική δομή της, γεγονός που να επιτρέπει την χρήση πιο εξειδικευμένων και γρήγορων αλγορίθμων, σε σχέση με γενικευμένους αλγόριθμους (π.χ. γενικευμένους αλγορίθμους αναζήτησης όπως ο Baby step-Giant), για την επίλυση του DLP ή του CDH προβλήματος. Επιθέσεις που σχετίζονται με την εκάστοτε υλοποίηση κάποιου σχήματος δεν μπορούμε να της λάβουμε υπόψη, αφού αντικείμενο της θεωρητικής ασφάλειας δεν είναι η υλοποίηση ενός αλγορίθμου παρά μόνο ο αλγόριθμος ενός σχήματος. Έτσι το 1997 προτάθηκε από τον Shoup στην εργασία [37] ένα μοντέλο για την επίλυση αυτού του προβλήματος, δηλαδή την μοντελοποίηση της υλοποίησης μιας αλγεβρικής ομάδας.

Definition 3.1.7. Μοντέλο Γενικευμένης Ομάδας (Generic Group Model ή GGM) [38] : Έστω μια πολλαπλασιαστική ομάδα $\langle G, \cdot \rangle$, με $ord(G) = q$ και μια συνάρτηση κωδικοποίησης $\sigma : \mathcal{Z} \rightarrow 0, 1*$. Στο μοντέλο αυτό οποιαδήποτε αλληλεπίδραση με στοιχεία της ομάδας γίνεται μόνο μέσω ενός μαντείου $\mathcal{O}$. Το μαντείο αυτό υποστηρίζει δύο είδους ερωτήματα :

- $\mathcal{O}(i)$, με $i \in [0, q]$: Επιστρέφει $\sigma(i)$. Αν δεν έχει ξανά δεχτεί το ίδιο ερώτημα i επιστρέφει μια ομοιόμορφα τυχαία τιμή διαφορετικά επιστρέφει την ίδια τιμή $\sigma(i)$ που επέστρεψε και στο παρελθόν.

- $\mathcal{O}(r, s, \sigma(i), \sigma(j))$, με $r, s \in [0, q]$: Επιστρέφει $\sigma(ri + sj \mod q)$. Ψάχνει τα $\sigma(i)$ και $\sigma(j)$ στον κατάλογο του, βρίσκει τα αντίστοιχα i, j και υπολογίζει την τιμή $k = ri + sjmodq$. Στην συνέχεια εκτελεί το $\mathcal{O}(k)$.

Μπορεί παρόμοια να οριστεί και Γενικευμένο Μοντέλο Δακτυλίων. Το παραπάνω μοντέλο θεωρείται αρκετά εξιδανικευμένο αφού για παράδειγμα σε πολλές υλοποιήσεις το ταυτοτικό στοιχείο αναπαρίσταται με το 1, το οποίο έρχεται σε πλήρη αντίθεση με το μοντέλο GGM. Το μοντέλο αυτό έχει δεχτεί αρκετή κριτική από την βιβλιογραφία κυρίως γιατί έχουν υπάρξει παραδείγματα αποδείξεων για κρυπτογραφικά σχήματα, όπως η [39] για το σχήμα RSA-OAEP, οι οποίες βασίζονταν το GGM και εν τέλη αποδείχθηκε ότι διέθεταν λάθος στην απόδειξη, όπως η [40] για το RSA-OAEP. Ωστόσο, συνήθως τα προβλήματα στις αποδείξεις αυτές όπως αναλύεται στην εργασία [38] δεν σχετίζονται άμεσα με το GGM αλλά με το ότι μπορεί να απλοποιήσει σημαντικά τις διαδικασίες απόδειξης με αποτέλεσμα να παραληφθούν άλλες σημαντικές επιφάνειες επίθεσης στην ασφάλεια ενός σχήματος.

Ένα ακόμα μοντέλο που χρησιμοποιείται ευρέως είναι αυτό του Τυχαίου Μαντείου. Συνήθως χρησιμοποιείται ως αφαίρεση για μια κρυπτογραφική συνάρτηση κατακερματισμού που μπορεί να χρησιμοποιεί κάποιο σχήμα. Όπως και το GGM, στην βιβλιογραφία θεωρείται ως αρκετά ουτοπικό καθώς αρκετές συναρτήσεις κατακερματισμού που χρησιμοποιούνταν σε κρυπτογραφικές υλοποιήσεις έως και πριν μερικά χρόνια, όπως η SHA-1 ή η MD5, δεν μπορούν να μοντελοποιηθούν σωστά από αυτό το το μοντέλο αυτό, αφού είναι γνωστό πως είναι ευάλωτες σε Επιθέσεις Επέκτασης Μήκους. Το μοντέλο αυτό ορίζεται ως εξής :

Definition 3.1.8. Μοντέλο Τυχαίου Μαντείου (Random Oracle Model ή ROM) : Πρόκειται για ένα μαντείο $\mathcal{O} \to 0, 1^* \times 0, 1^*$ που δέχεται ερωτήματα με μια παράμετρο, δηλαδή $\mathcal{O}(i)$ και επιστρέφει μια τιμή. Αν η τιμή i δεν έχει επαναληφθεί σε προηγούμενο ερώτημα τότε η τιμή που επιστρέφεται είναι μια ομοιόμορφα τυχαία τιμή. Διαφορετικά αν η τιμή i έχει επαναληφθεί επιστρέφεται τιμή που είχε επιστραφεί στο τελευταίο ερώτημα για την τιμή i

Έχει αποδειχθεί ότι δεν μπορούμε να κατασκευάσουμε συναρτήσεις που να έχουν τις ιδιότητες ενός Τυχαίου Μαντείου και ταυτόχρονα να είναι υπολογιστικά αποδοτικές. Αυτό έχει ως αποτέλεσμα πολλά κρυπτογραφικά σχήματα να χρησιμοποιούν κρυπτογραφικές συναρτήσεις, οι οποίες είναι αποτελεσματικές με τα όποια όμως μειονεκτήματα υπάρχουν σε αυτές και στην συνέχεια να αφήνεται στην επιστημονική κοινότητα να καταλήξει στο αν αυτά τα μειονεκτήματα μπορούν να χρησιμοποιηθούν αποτελεσματικά για την παραβίαση της ασφάλειας του σχήματος.

Σε πολλά μη διαδραστικά σχήματα, όπως οι Μη Διαδραστικές Αποδείξεις Μηδενικής Γνώσης που θα εξετάσουμε σε επόμενη ενότητα, προκειμένου να απλοποιηθεί η απόδειξη του σχήματος χρειάζεται να υποθέσουμε κάποιο κοινό σημείο εκκίνησης, κάποια κοινή γνώση

μεταξύ των συμμετεχόντων. Η πληροφορία αυτή συνήθως διατυπώνεται με την μορφή συμβολοσειρών αλλά μπορεί να έχει και οποιαδήποτε άλλη μορφή, δηλαδή ότι οι συμμετέχοντες έχουν όλοι στην κατοχή τους την ίδια συμβολοσειρά. Αυτή την συμβολοσειρά μπορεί να έχουν χρησιμοποιήσει κάποιο άλλο πρωτόκολλο για να την αποκτήσουν είτε απλά να βασιστούν σε έναν έμπιστο τρίτο άτομο να την διαμοιράσει σε αυτούς. Έτσι προκύπτει το παρακάτω μοντέλο.

Definition 3.1.9. Μοντέλο Συμβολοσειρών Κοινής Αναφοράς (Common Reference String Model ή CRSM) : Το μοντέλο μοντέλο αυτό υποθέτει ότι όλες οι οντότητες ενός κρυπτογραφικού σχήματος έχουν πρόσβαση σε μια συμβολοσειρά crs που ανήκει σε μια προκαθορισμένη κατανομή D.

3.1.3 Βασικές Κρυπτογραφικές Υποθέσεις

Τελειώνοντας με την μελέτη της σημειολογίας στις αποδείξεις θεωρητικής ασφάλειας, δεν μπορούμε να παραλείψουμε τις βασικές κρυπτογραφικές υποθέσεις ή καλύτερα βασικές υποθέσεις της υπολογιστικής πολυπλοκότητας που έχουν εφαρμογή στην κρυπτογραφία. Αυτές πρόκειται για προτάσεις που δεν έχουμε καταφέρει να αποδείξουμε ούτε ότι ισχύουν αλλά ούτε το αντίθετο. Ωστόσο, τα περισσότερα κρυπτογραφικά σχήματα που έχουν αναφερθεί στην κρυπτογραφία βασίζουν την ασφάλεια τους σε αυτές τις υποθέσεις. Για παράδειγμα, ο κλάδος της κρυπτογραφίας δεν έχει καταφέρει να αποδείξει αν το πρωτόκολλο ανταλλαγής κλειδιών Diffie-Hellman είναι υπολογιστικά ασφαλές στο κλασσικό μοντέλο υπολογισμού (στο κβαντικό μοντέλο υπολογισμού γνωρίζουμε ότι δεν είναι), ωστόσο επειδή έχουν περάσει σχεδόν 50 χρόνια από τότε που προτάθηκε εικάζεται στην βιβλιογραφία ότι είναι πολύ πιθανό αυτό να συμβαίνει. Έτσι επειδή και άλλα σχήματα βασίζονται στο ίδιο πρόβλημα με αυτό του Diffie-Hellman, έχουμε δημιουργήσει ένα πρόβλημα για αυτό, αλλά και μια συσχετιζόμενη υπόθεση με το πρόβλημα αυτό. Το πρόβλημα Diffie-Hellman το απαντάμε σε δύο παραλλαγές αυτή του Υπολογιστικού και αυτή του Αποφαστιστικού Προβλήματος Diffie Hellman. Προφανώς υπάρχουν και οι αντίστοιχες υποθέσεις. Αυτές είναι οι παρακάτω.

Definition 3.1.10. Υπόθεση Αποφαστιστικού Diffie-Hellman (Decisional Diffie-Hellman ή DDH Assumption) : Έστω μια πολλαπλασιαστική ομάδα πρώτης τάξης G. Η DDH υπόθεση εκφράζει ότι οι πιθανοτικές κατανομές (g^a, g^b, g^{ab}) και (g^a, g^b, g^c), όπου $a, b, c \in G$ είναι τυχαία και ομοιόμορφα επιλεγμένα, είναι υπολογιστικά δυσδιάκριτες. Ισοδύναμα, το πλεονέκτημα οποιοδήποτε μη ντετερμινιστικού αντιπάλου με υπολογιστικά περιορισμένους πόρους που προσπαθεί να τις διακρίνει είναι το πολύ μηδαμινό σε συνάρτηση της παραμέτρου ασφάλειας η οποία είναι ο αριθμός των ψηφίων της τάξης της ομάδας G. Πιο αυστηρά

το ορίζουμε το εξής :

$$\forall A \in PPT : Pr[A(g^a, g^b, g^a b) = 1] - Pr[A(g^a, g^b, g^c)] \leq negl(1^n) = \varepsilon_{DDH}$$

όπου $a, b, c \leftarrow\!\!\$\ G$ Το πλεονέκτημα το συμβολίζουμε με ε_{DDH}.

Definition 3.1.11. Υπόθεση Υπολογιστικού Diffie-Hellman (Computational Diffie-Hellman ή CDH Assumption) : Έστω μια πολλαπλασιαστική ομάδα πρώτης τάξης G. Η CDH υπόθεση εκφράζει ότι οποιοσδήποτε πολυωνυμικός αλγόριθμος με είσοδο την τριάδα (g^a, g^b, g) όπου $a, b, g \in G$ είναι τυχαία και ομοιόμορφα επιλεγμένα έχει το πολύ μηδαμινή πιθανότητα εύρεσης του g^{ab}.

Είναι πολύ εύκολο να αποδειχθεί ότι η υπόθεση DDH είναι πιο ισχυρή υπόθεση από την CDH. Για λόγους πληρότητας αναφέρουμε πως το DHKE βασίζεται στην CDH υπόθεση, ωστόσο η DDH ονομάστηκε έτσι γιατί είναι συγγενικά παρόμοια με αυτή της CDH. Μια ακόμα πολύ γνωστή υπόθεση που σχετίζεται με αυτές είναι η Υπόθεση Διακριτού Λογαρίθμου η οποία είναι πιο χαλαρή υπόθεση ακόμα και από το CDH.

Definition 3.1.12. Υπόθεση Διακριτού Λογαρίθμου (Discrete Logarithm Assumption) : Έστω μια κυκλική πολλαπλασιαστική ομάδα $\langle G, \cdot \rangle$ με $G = \langle g \rangle$ όπου g ένας γεννήτορας της G (π.χ. η $\langle \mathbb{Z}_p, \cdot \rangle$ όπου p πρώτος αριθμός). Η υπόθεση αυτή εκφράζει ότι, οποιοσδήποτε πολυωνυμικός στον αριθμό ψηφίων της τάξης της ομάδας αλγόριθμος, για ένα τυχαίο στοιχείο $a \in G$ έχει μηδαμινή πιθανότητα επίλυσης του Προβλήματος Διακριτού Λογαρίθμου, δηλαδή την εύρεση $x \in G$ τέτοιου ώστε $log_g(a) = x$.

Definition 3.1.13. Υπόθεση Συνάρτηση Μονής Διαδρομής (One Way Function ή OWF) : Μια συνάρτηση $f : \{0,1\}^* \rightarrow \{0,1\}^*$ η οποία είναι εύκολο να υπολογιστεί από έναν πολυωνυμικό στο χρόνο αλγόριθμο αλλά η αντίστροφη της, f^{-1}, δεν μπορεί να υπολογιστεί επιτυχώς με μη μηδαμινή πιθανότητα από οποιονδήποτε πολυωνυμικό αλγόριθμο.

3.2 Μοντέλα αντιπάλων

Αφού είδαμε βασικούς ορισμούς της θεωρητικής ασφάλειας βασιζόμενοι στον γενικό ορισμό ενός κρυπτογραφικού αντιπάλου τώρα θα προχωρήσουμε στην εξειδίκευση αυτού του ορισμού. Είναι προφανές ότι για να δημιουργήσουμε την απόδειξη ασφάλειας ενός κρυπτογραφικού σχήματος είναι απαραίτητο να διαθέτουμε ένα μοντέλο του αντιπάλου που θα κληθεί αυτό να αντιμετωπίσει όταν υλοποιηθεί στην πράξη. Είναι πολύ σημαντική η επιλογή κατάλληλου αντιπάλου που να προσεγγίζει όσο πιο ρεαλιστικά γίνεται μια οντότητα που θα προσπαθήσει να παραβιάσει την ασφάλεια του σχήματος μας και πολλές φορές καθορίζει την πρακτικότητα του σχήματος μας. Στην βιβλιογραφία υπάρχουν διάφορες κατηγορίες

με μοντέλα αντιπάλων τα οποία μπορούμε να χρησιμοποιήσουμε και να συνδυάσουμε ανάλογα με την φύση του εκάστοτε κρυπτογραφικού σχήματος και των ιδιοτήτων ασφάλειας που επιθυμούμε να αποδείξουμε. Παρακάτω αναφέρουμε τις βασικές κατηγορίες αντιπάλων που απαντώνται στη βιβλιογραφία και σε ποια χαρακτηριστικά βασίζεται η διαφοροποίηση τους:

Definition 3.2.1. Κατηγορίες αντιπάλων:

- Αντίπαλοι διαφοροποιημένοι ως προς την υπολογιστική τους ισχύ.
- Αντίπαλοι διαφοροποιημένοι ως προς την δυνατότητα διαφθοράς των συμμετεχόντων.
- Αντίπαλοι διαφοροποιημένοι ως προς το πότε επιλέγονται οι διεφθαρμένοι συμμετέχοντες.

Στην συνέχεια, αναλύουμε τις παραπάνω κατηγορίες. Προφανώς, ένας αντίπαλος μπορεί να συνδυάζει περισσότερα από ένα χαρακτηριστικά, τα οποία όμως ανήκουν σε διαφορετικές κατηγορίες.

3.2.1 Αντίπαλοι διαφοροποιημένοι ως προς την υπολογιστική ισχύ

Η διαφοροποίηση των αντιπάλων ως προς την υπολογιστική του ισχύ είναι αντίστοιχη με αυτήν την ύπαρξης δύο διαφορετικών παραμέτρων ασφαλείας που αναλύσαμε σε προηγούμενη ενότητα. Ως επί το πλείστον βέβαια τα κρυπτογραφικά σχήματα σήμερα προκειμένου να είναι πρακτικά είναι ασφαλή μόνο ενάντια σε υπολογιστικά φραγμένους αντιπάλους.

Definition 3.2.2. Αντίπαλος με υπολογιστικά άπειρους πόρους (Computationally unbounded resources adversary) είναι ένας αλγόριθμος, πιθανοκρατικός ή μη, ο οποίος έχει άπειρη υπολογιστική ισχύ.

Definition 3.2.3. Αντίπαλος με υπολογιστικά φραγμένους πόρους (Computationally bounded resources adversary) είναι ένας πολυωνυμικός αλγόριθμος, πιθανοκρατικός ή μη, που τρέχει σε πολυωνυμικό χρόνο ως προς την παράμετρο υπολογιστική ασφάλειας n.

3.2.2 Αντίπαλοι διαφοροποιημένοι ως προς την δυνατότητα διαφθοράς των συμμετεχόντων

Η διαφοροποίηση αυτή τον αντίπαλων είναι αρκετά κρίσιμη και καθορίζει άμεσα τη χρησιμότητα του κρυπτογραφικού σχήματος αλλά και την δυσκολία απόδειξης της ασφάλειας του. Είναι προφανές ότι η μεγαλύτερη ισοδυναμεί με δυσκολότερη ή και αδύνατη την απόδειξη της ασφάλειας ενός κρυπτογραφικού σχήματος ενάντια σε έναν τέτοιο αντίπαλο. Οι

κύριες κατηγορίες αντιπάλων ως προς την δυνατότητα διαφθοράς των συμμετεχόντων σε ένα κρυπτογραφικών σχήμα είναι οι παρακάτω :

Definition 3.2.4. Παθητικός αντίπαλος (Passive adversary) ή Περίεργος αντίπαλος (Honest-but-curious adversary) είναι ένας αλγόριθμος, πιθανοκρατικός ο οποίος δεν επεμβαίνει παρεμβατικά στο κρυπτοσύστημα με τρόπο ώστε να αλλάξει τον τρόπο εκτέλεσης του. Εκτελεί το κρυπτοσύστημα σωστά, ωστόσο από τις πληροφορίες που συλλέγει, από τους συμμετέχοντες που έχει διαφθείρει, μπορεί να εκτελέσει επιπλέον υπολογισμούς με σκοπό την εξαγωγή επιπλέον πληροφοριών που οδηγούν στο σπάσιμο της ασφάλειας του συστήματος. Η παρουσία του δεν μπορεί να γίνει αντιληπτή από τους μη διεφθαρμένους συμμετέχοντες.

Definition 3.2.5. Ενεργητικός αντίπαλος (Reactive adversary) είναι ένας αλγόριθμος, πιθανοκρατικός ή μη, ο οποίος είναι ένα υπερσύνολο του Παθητικού αντιπάλου, με την έννοια ότι πέρα τον επιπλέον υπολογισμών που μπορεί να εκτελέσει, μπορεί επίσης να εκτελέσει και να απέχει αυθαίρετα από το κρυπτοσύστημα με σκοπό το σπάσιμο της ασφάλειας του. Η παρουσία του γίνεται αντιληπτή από τους μη διεφθαρμένους συμμετέχοντες.

Definition 3.2.6. Συγκαλυμμένος αντίπαλος (Covert adversary) είναι ένας αλγόριθμος, πιθανοκρατικός ή μη, ο οποίος ενεργεί όπως ο Ενεργητικός αντίπαλος, με τον περιορισμό ότι η παρουσία του δεν γίνεται αντιληπτή από τους μη διεφθαρμένους συμμετέχοντες.

Definition 3.2.7. Ημι-κακόβουλος αντίπαλος (Semi-malicious adversary) είναι ένας αλγόριθμος, πιθανοκρατικός ή μη, ο οποίος ακολουθεί το πρωτόκολλο όμως επιλέγει αυθαίρετα την είσοδο του και το αποτέλεσμα τυχαίων πειραμάτων στην περίπτωση που είναι πιθανοκρατικός.

Ουσιαστικά η κλάση των αλγορίθμων των Παθητικών αντίπαλων είναι ένα υποσύνολο της κλάσης Συγκαλημένων αντιπάλων και η κλάση των Συγκαλυμμένων αντίπαλων είναι ένα υποσύνολο της κλάσης των Ενεργητικών αντιπάλων.

3.2.3 Αντίπαλοι διαφοροποιημένοι ως προς το πότε επιλέγονται οι διεφθαρμένοι συμμετέχοντες

Μια ακόμα παράμετρος διαφοροποίησης των αντιπάλων που έχει εφαρμογή κυρίως σε κρυπτογραφικά σχήματα με πολλούς συμμετέχοντες είναι αυτή του συνόλου των διεφθαρμένων συμμετεχόντων που ελέγχει ο αντίπαλος. Το σύνολο αυτό μπορεί να είναι είτε στατικό είτε δυναμικό. Οι ακριβείς ορισμοί είναι οι παρακάτω :

Definition 3.2.8. Στατικός αντίπαλος (Static adversary) είναι ένας αλγόριθμος, πιθανοκρατικός ή μη, ο οποίος επιλέγει στατικά τους διεφθαρμένους συμμετέχοντες στην αρχή, δηλαδή δεν έχει την δυνατότητα να αλλάξει τις επιλογές του στην συνέχεια.

Definition 3.2.9. Προσαρμοστικός αντίπαλος (Adaptive adversary) είναι ένας αλγόριθμος, πιθανοκρατικός ή μη, οποίος επιλέγει δυναμικά τους διεφθαρμένους συμμετέχοντες κατά την διάρκεια εκτέλεσης του κρυπτοσυστήματος.

3.3 Είδη Ασφάλειας (Security Notions)

Στην ενότητα αυτή θα μελετήσουμε κάποια είδη ασφάλειας της ασφάλειας ξεκινώντας από τους ιστορικά πιο παλιούς και καταλήγοντας σε αυτούς που χρησιμοποιούνται σήμερα. Ο Shannon, το 1945, στην εργασία [6] πέραν του Ορισμού 2.1.1 όρισε και την έννοια της Απόλυτης Ασφάλειας για το Κρυπτογραφικό Σχήμα Shannon.

Definition 3.3.1. Απόλυτη ασφάλεια : Έστω $\mathcal{E} = (\text{Enc}, \text{Dec})$ ένα Κρυπτογραφικό Σχήμα Shannon ορισμένο στα $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ και έστω ένας αντίπαλος $\mathcal{A}$ με υπολογιστικά άπειρους πόρους. Αν το $k \sim U(\mathcal{K})$, το Κρυπτογραφικό Σχήμα είναι Απόλυτα Ασφαλές απέναντι στον $\mathcal{A}$ αν, και μόνο αν,

$$\forall m_0, m_1 \in \mathcal{M}, \forall c \in \mathcal{C} : Pr[\text{Enc}(\mathbf{k}, m_0) = c] = Pr[\text{Enc}(\mathbf{k}, m_1) = c] \tag{3.1}$$

$$\Leftrightarrow \forall m_0, m_1, \text{Enc}(\mathbf{k}, m_0) \equiv \text{Enc}(\mathbf{k}, m_1) \tag{3.2}$$

Η Σχέση 3.1 ίσως να γίνει πιο κατανοητή από τον αναγνώστη αν λάβει υπόψιν του και την παρακάτω Σχέση :

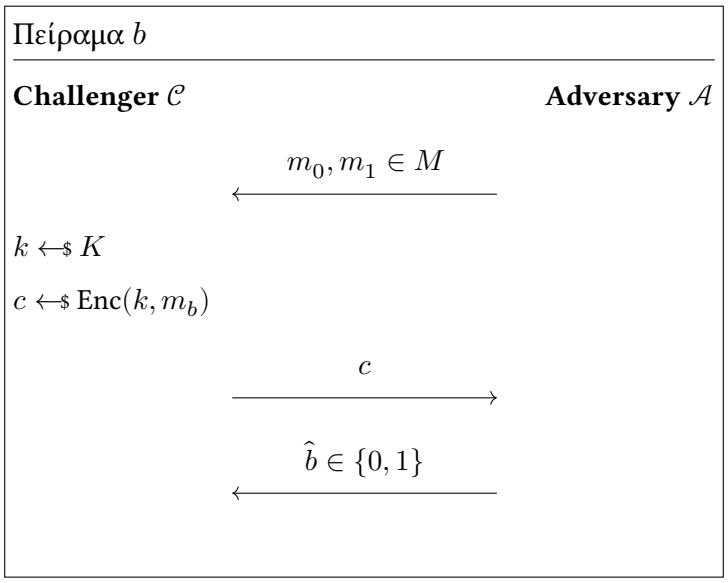
$$\text{Για σταθερά } m, c \ : Pr[\text{Enc}(\mathbf{k}, m) = c] = \frac{\#(k \in \mathcal{K} : \text{Enc}(k, m) = c)}{|\mathcal{K}|} \tag{3.3}$$

Πρακτικά αυτό σημαίνει, αν η επιλογή των κλειδιών k είναι ομοιόμορφη τότε για ένα συγκεκριμένο μήνυμα m και ένα συγκεκριμένο κρυπτοκείμενο c, το πλήθος των κλειδιών k που μπορούν να δώσουν ως αποτέλεσμα $\text{Enc}(k, m) = c$ είναι το ίδιο για κάθε πιθανό ζεύγος m και c. Με άλλα λόγια, ο αντίπαλος έχοντας πρόσβαση στο κρυπτοκείμενο δεν κερδίζει καμία πληροφορία σχετικά με το μήνυμα σε σχέση με αυτήν που είχε χωρίς να έχει πρόσβαση σε αυτό.

Ένας ακόμα τρόπος να ορίσουμε την Ασφάλεια ενός κρυπτογραφικού σχήματος, είναι αυτός του Παιχνιδιού της Δυσδιακριτότητας - Επίθεσης Επιλεγμένου Μηνύματος (Indistinguishability - Chosen Plaintext Attack ή IND-CPA), ο οποίος χρησιμοποιήθηκε αρχικά από τους Goldwasser και Micali στην εργασία [41] για τον ορισμό της Σημασιολογικής Ασφάλειας (Semantic Security), την οποία και θα ορίσουμε στην συνέχεια, ωστόσο ο τρόπος απόδειξης μέσω παιχνιδιού είχε μεγάλη απήχηση, αποκόπηκε από την έννοια της Σημασιολογικής Ασφάλειας και εν συνέχεια χρησιμοποιήθηκε και για τον ορισμό και άλλων τύπων

ασφάλειας. Θα αναφερθούμε πιο αναλυτικά στις αποδείξεις ασφάλειας μέσω παιχνιδιού σε επόμενη ενότητα, ωστόσο προς το παρόν μπορούμε να σταθούμε σε αυτόν τον απλό ορισμό.

Definition 3.3.2. Παιχνίδι Δυσδιακριτότητας - Επίθεσης Επιλεγμένου Μηνύματος ή IND-CPA-**Game** : Για δεδομένο κρυπτογραφικό σχήμα $\mathcal{E} = (\text{Enc}, \text{Dec})$ ορισμένο στο $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ και αντίπαλο a με υπολογιστικά άπειρους πόρους, ορίζουμε δύο πειράματα, Πείραμα 0 και Πείραμα 1, για $b = 0, 1$ ως εξής :



Έστω S_b το γεγονός ότι $b = \hat{b}$, δηλαδή ο αντίπαλος $\mathcal{A}$ να μάντεψε σωστά ποιο κείμενο κρυπτογραφήσε ο προκαλών $\mathcal{C}$ στο Πείραμα b. Τότε το πλεονέκτημά του $\mathcal{A}$ μπορεί να οριστεί ως την πόση μεγαλύτερη πιθανότητα έχει να μαντέψει σωστά στο Πείραμα 1 σε σχέση με το Πείραμα 2. Δηλαδή, πιο τυπικά ως εξής :

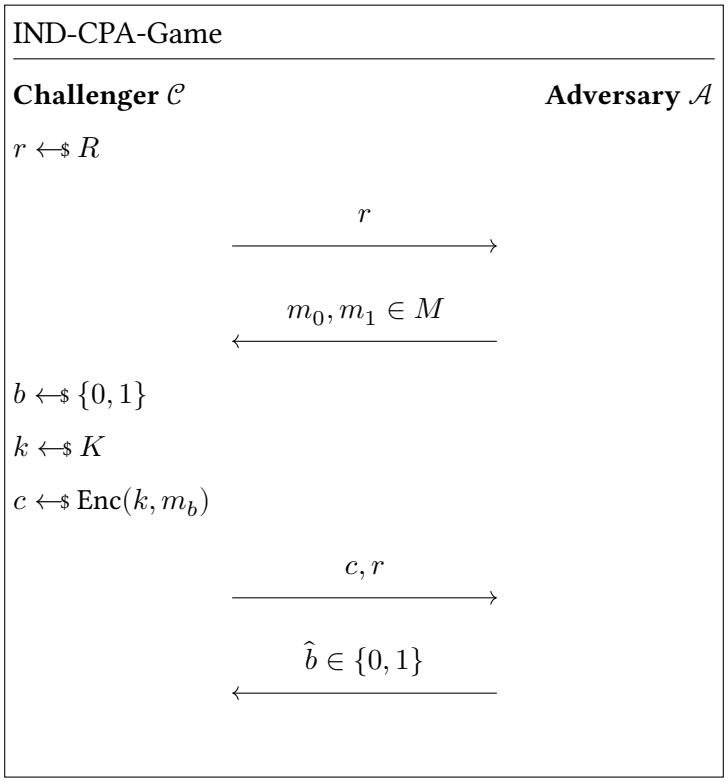
$$\text{IND-CPA-}Advantage[\mathcal{A}, \mathcal{C}] = |Pr[S_0] - Pr[S_1]| \tag{3.4}$$

Τώρα η απόλυτη ασφάλεια μπορεί να οριστεί ισοδύναμα ως εξής :

Definition 3.3.3. Απόλυτη ασφάλεια (Ισοδύναμος ορισμός) : Ένα κρυπτογραφικό σχήμα $\mathcal{E}$ είναι απόλυτα ασφαλές αν, και μόνο αν, IND-CPA-$Advantage[\mathcal{A}, \mathcal{C}] = |Pr[S_0] - Pr[S_1]| = 0$. Δηλαδή, αν οποιοδήποτε ζεύγος κατανομών που παράγει το σύστημα είναι ισοδύναμες.

Ένας επίσης ισοδύναμος ορισμός της Σημασιολογικής Ασφάλειας που συναντάται συχνά στη βιβλιογραφία, είναι το να θεωρήσουμε ότι αντί δύο πειραμάτων έχουμε ένα στο οποίο η μεταβλητή b δεν θεωρείται παράμετρος. Τότε αλλάζει γεγονός που χρησιμοποιούμε για να ορίσουμε την ασφάλεια του σχήματος και επίσης αλλάζει και το πως παρέχεται η τυχαιότητα στο παιχνίδι.

Definition 3.3.4. Παιχνίδι Δυσδιακριτότητας - Επίθεση Επιλεγμένου Μηνύματος ή IND-CPA**-Game (Ισοδύναμος Ορισμός)** :



Έστω S το γεγονός ότι $b = \hat{b}$, δηλαδή ο αντίπαλος $\mathcal{A}$ να μάντεψε σωστά ποιο κείμενο κρυπτογράφησε ο προκαλών $\mathcal{C}$. Τότε το πλεονέκτημά του $\mathcal{A}$ μπορεί να οριστεί ως την πόση μεγαλύτερη πιθανότητα έχει να μαντέψει σωστά σε σχέση με το αν μάντευε στην τύχη. Δηλαδή, πιο τυπικά ως εξής:

$$\text{IND-CPA-}Advantage[\mathcal{A}, \mathcal{C}] = |Pr[S] - \frac{1}{2}| \tag{3.5}$$

Definition 3.3.5. Απόλυτη ασφάλεια (Ισοδύναμος ορισμός) : Ένα κρυπτογραφικό σχήμα $\mathcal{E}$ είναι απόλυτα ασφαλές αν, και μόνο αν,

$$\text{IND-CPA-}Advantage[\mathcal{A}, \mathcal{C}] = 0$$

Μια βασική διαφορά των δύο ορισμών είναι ότι στον Ισοδύναμο Ορισμό 1, οι αλγόριθμοι του προκαλούντος και του αντιπάλου, έχουν ενσωματωμένη την πηγή τυχαιότητας άρα είναι τυχαιοκρατικοί, ενώ στην περίπτωση του Ισοδύναμου Ορισμού 2 του δίνεται ως όρισμα και άρα είναι ντετερμινιστικός. Στα πλαίσια της εργασίας αυτής θα χρησιμοποιήσουμε τον Ορισμό 3.3.5, γιατί είναι πιο συνοπτικός και επιτρέπει με πιο κατανοητό τρόπο τη χρήση της

μεθόδου Μεταπήδησης μεταξύ Παιχνιδιών που θα αναλύσουμε στη συνέχεια της ενότητας.

Δυστυχώς, στην ίδια εργασία [6] ο Shannon διατύπωσε το περίφημο θεώρημα του σχετικά με Απόλυτη Ασφάλεια ενός κρυπτογραφικού σχήματος, σύμφωνα με το οποίο το πιο αποδοτικό απόλυτα ασφαλές σχήμα είναι το Σημειωματάριο Μιας Χρήσης (OTP) το οποίο δεν είναι πρακτικό λόγω του μεγέθους που απαιτείται να έχει το κλειδί, δηλαδή να είναι ίδιου μήκους με το μήνυμα.

Definition 3.3.6. Θεώρημα Shannon : Έστω ένα κρυπτογραφικό σχήμα Shannon $\mathcal{E} = (\text{Enc}, \text{Dec})$ ορισμένο στο $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ με $|\text{K}| = L$, $|M| = N$. Δεν μπορεί να υπάρξει απόλυτα ασφαλές κρυπτογραφικό σχήμα με $L < N$. Δηλαδή, δεν μπορεί να υπάρξει απόλυτα ασφαλές κρυπτογραφικό σχήμα με μέγεθος κλειδιού μικρότερο από το μέγεθος του μηνύματος.

Το παραπάνω Θεώρημα αποτελεί ένα πολύ περιοριστικό αποτέλεσμα το οποίο οφείλεται στο ότι ορίζεται πάνω σε πολύ ισχυρές υποθέσεις, αυτή του αντίπαλου με απεριόριστους υπολογιστικούς πόρους και αυτή του μηδενικού πλεονεκτήματος του αντιπάλου. Για να προσπεράσουμε το παραπάνω ιδιαίτερα περιοριστικό θεώρημα, πρέπει να χαλαρώσουμε τις υποθέσεις. Έτσι στην βιβλιογραφία έχουν προκύψει οι παρακάτω ορισμοί ασφάλειας. Στην περίπτωση που χαλαρώσουμε τον περιορισμό του μηδενικού πλεονεκτήματος προκύπτει ο ορισμός της Στατιστικής Ασφάλειας ενώ αν χαλαρώσουμε επίσης και τον περιορισμό του αντιπάλου με άπειρους υπολογιστικούς πόρους προκύπτει η Υπολογιστική Ασφάλεια ή αλλιώς Σημασιολογική Ασφάλεια.

Definition 3.3.7. Στατιστική Ασφάλεια : Ένα κρυπτογραφικό σχήμα $\mathcal{E}$ είναι Στατιστικά Ασφαλές απέναντι σε αντιπάλους $\mathcal{A}$ με υπολογιστικά άπειρους πόρους αν, και μόνο αν,

$$\text{IND-CPA-}Advantage[\mathcal{A}, \mathcal{C}] = |Pr[S] - \frac{1}{2}| = negl(n)$$

Definition 3.3.8. Υπολογιστική Ασφάλεια (Computational Security) ή Σημασιολογική Ασφάλεια (Semantic Security) Ένα κρυπτογραφικό σχήμα $\mathcal{E}$ είναι Υπολογιστικά Ασφαλές απέναντι σε αντιπάλους $\mathcal{A}$ με υπολογιστικά περιορισμένους πόρους αν, και μόνο αν,

$$\text{IND-CPA-}Advantage[\mathcal{A}, \mathcal{C}] = |Pr[S] - \frac{1}{2}| = negl(n)$$

Σε αυτό το σημείο, πρέπει να αναφερθεί ότι η Σημασιολογική Ασφάλεια όπως ορίστηκε αρχικά στην εργασία [41] δεν ήταν συνδεδεμένη με το παιχνίδι IND-CPA. Ωστόσο, ο ορισμός που της είχε δοθεί δεν ήταν πολύ πρακτικός και όπως αποδείχθηκε ήταν ισοδύναμος με τον ορισμό της Υπολογιστικής ασφάλειας μέσω IND-CPA όπως και ορίζεται σήμερα. Στη συνέχεια το παιχνίδι IND-CPA χρησιμοποιήθηκε και για τον ορισμό της Στατιστικής, της Απόλυτης Ασφάλειας καθώς και άλλως τύπων ασφάλειας. Οι παραπάνω Ορισμοί συνοψίζονται στον Πίνακα 3.1.

IND-CPA-$Adv(\mathcal{A}, \mathcal{C})$	Μη Φραγμένος Αντίπαλος	Φραγμένος Αντίπαλος
Απόλυτη Ασφάλεια	0	
Φραγμένη Ασφάλεια	$negl(1^n)$	
Υπολογιστική Ασφάλεια		$negl(1^n)$

Πίνακας 3.1: Σύνοψη ορισμών ασφάλειας

3.4 Αποδείξεις ασφάλειας

Αφού καλύψαμε τη βασική προαπαιτούμενη σημειολογία και ορισμούς της θεωρητικής ασφάλειας στη κρυπτογραφία που μας επιτρέπει να περάσουμε στο κυρίως θέμα του κεφαλαίου, τις αποδείξεις ασφάλειας. Θα εξετάσουμε αρχικά την γενική βασική μεθοδολογία που μπορεί να ακολουθήσει κάποιος που προσπαθεί να συντάξει την απόδειξη ασφάλειας ενός κρυπτογραφικού συστήματος. Στην συνέχεια θα αναφερθούμε στις δύο βασικές μεθόδους που χρησιμοποιούνται σήμερα στις κρυπτογραφικές αποδείξεις και τέλος θα συγκρίνουμε τα πλεονεκτήματα και τα μειονεκτήματα των μεθόδων αυτών.

3.4.1 Βασική Μεθοδολογία

Αν υποθέσουμε ότι διαθέτουμε ένα οποιοδήποτε κρυπτογραφικό σχήμα, το οποίο θέλουμε να αποδείξουμε ότι είναι ασφαλές ενάντια σε κάποιο μοντέλο αντιπάλων, τότε η γενική μεθοδολογία που ακολουθούμε στην απόδειξη της ασφάλειας του είναι παρόμοια, και αποτελείται από τα παρακάτω βήματα :

1. **Μοντελοποίηση των συστατικών στοιχείων του κρυπτογραφικού σχήματος**
   Στο βήμα αυτό αναπαριστούμε το κρυπτοσύστημα μας με ένα μοντέλο αυτού. Αυτό ουσιαστικά σημαίνει ότι σε κάθε κρυπτογραφικό συστατικό στοιχείο που χρησιμοποιείται το αντικαθιστούμε με μια μοντελοποίηση αυτού, δηλαδή μια αφαιρετική μαθηματική αναπαράσταση αυτού. Για παράδειγμα, αν χρησιμοποιείται μια κρυπτογραφική συνάρτηση κατακερματισμού μπορούμε να την αντικαταστήσουμε με το Μοντέλο Τυχαίου Μαντείου (Random Oracle Model) και σε αυτήν την περίπτωση η ασφάλεια του συστήματος μας θα βασίζεται στην Υπόθεση Τυχαίου Μαντείου (Random Oracle Assumption) [42]. Είναι πολύ σημαντικό οι υποθέσεις που γίνονται, να αντικατοπτρίζουν κατά το μέγιστο δυνατό την πραγματικότητα, διαφορετικά η πρακτική ασφάλεια του κρυπτοσυστήματος μας μπορεί να απέχει πολύ από αυτήν.

2. **Μοντελοποίηση δρώντων**
   Αναπαριστούμε τους δρώντες του συστήματος μας, με μοντέλα αυτών. Οι δρώντες, αποτελούνται τους καλοπροαίρετους συμμετέχοντες στο κρυπτοσύστημα αλλά και

τους κακόβουλους/διεφθαρμένους συμμετέχοντες που τους χειρίζεται ο αντίπαλος. Για παράδειγμα, οι κακόβουλοι συμμετέχοντες μπορεί να είναι παθητικοί ή ενεργητικοί, μπορεί επίσης να έχουν πρόσβαση σε περιορισμένους ή απεριόριστους υπολογιστικούς πόρους ανάλογα με το είδος του αντιπάλου. Αντίστοιχα και οι καλοπροαίρετοι μπορεί να έχουν πρόσβαση σε περιορισμένους ή απεριόριστους πόρους. Στην πράξη συνήθως και τα δύο είδη συμμετεχόντων έχουν πρόσβαση σε περιορισμένους πόρους. Ουσιαστικά στο βήμα αυτό αποφασίζουμε τα χαρακτηριστικά που θα θεωρήσουμε ότι θα έχει ο αντίπαλος, τα οποία τα αναλύσαμε σε προηγούμενη ενότητα και τα χαρακτηριστικά των καλοπροαίρετων συμμετεχόντων, εκεί ωστόσο υπάρχει μικρότερη γκάμα επιλογής χαρακτηριστικών.

3. **Μοντελοποίηση καναλιών επικοινωνίας**
   Αναπαριστούμε τα κανάλια επικοινωνίας του συστήματος, με μοντέλα αυτών. Για παράδειγμα, μπορούμε να θεωρήσουμε την ύπαρξη ενός ασφαλούς καναλιού επικοινωνίας το όποιο έχει εγκαθιδρυθεί με κάποιον εξωτερικό τρόπο ως προς το σύστημα μας.

4. **Μοντελοποίηση λοιπών συστατικών στοιχείων**

Improve this!

5. **Επιλογή κρυπτογραφικών ιδιοτήτων προς απόδειξη**
   Επιλέγουμε τις κρυπτογραφικές ιδιότητες που θα επιθυμούσαμε να έχει το μοντέλο του συστήματος που δημιουργήσαμε προηγουμένως.

6. **Απόδειξη κρυπτογραφικών ιδιοτήτων**
   Σε αυτό το τελευταίο βήμα αποδεικνύουμε τις κρυπτογραφικές ιδιότητες που επιλέξαμε στο προηγούμενο βήμα. Συνήθως στις αποδείξεις βασιζόμαστε σε αναγωγές σε προβλήματα που έχουν αποδειχθεί δυσεπίλυτα ή πιστεύεται ότι είναι δυσεπίλυτα στην βιβλιογραφία. Στις αποδείξεις αυτές συνήθως χρησιμοποιούμε κάποια μέθοδο απόδειξης όπως αυτές που θα αναλύσουμε παρακάτω.

3.4.2 Απόδειξη ασφάλειας κρυπτοσυστήματος μέσω παιχνιδιού (Game based security proof)

Η απόδειξης της ασφάλειας ενός κρυπτογραφικού σχήματος μέσω παιχνιδιού εισήχθη στην βιβλιογραφία από τους Goldwasser και Micali, το 1982, στην πολύ γνωστή εργασία τους [41], που εισήγαγε αρκετές καινοτομίες στον τομέα της Κρυπτογραφίας, όπως η πρόταση του πρώτου (πιθανοτικού) πρωτοκόλλου δημοσίου κλειδιού το οποίο είναι ασφαλές στο Κανονικό Μοντέλο. Στην συνέχεια η τεχνική αυτή ορίστηκε με πιο αυστηρό τρόπο στην εργασία [43] από τους Bellaware και Rogaway. Η μέθοδος αυτή είναι η πιο βασική μέθοδος που

χρησιμοποιείται στη βιβλιογραφία, και θεωρείται πιο απλή κατά την σύνταξη των αποδείξεων σε σχέση με αυτή της προσομοίωσης που θα εξετάσουμε στην επόμενη ενότητα. Ωστόσο δεν είναι ιδιαίτερα ισχυρή καθότι για κάθε διαφορετική ιδιότητα ασφάλειας που επιθυμούμε να εξασφαλίσουμε στο σύστημα μας πρέπει να παρέχουμε και την αντίστοιχη απόδειξη. Συνήθως προτιμάται για πιο απλά κρυπτογραφικά σχήματα με λίγους συμμετέχοντες και για λίγες επιθυμητές ιδιότητες ασφάλειας, διότι διαφορετικά είναι οι αποδείξεις γίνονται αρκετά πολύπλοκες. Στα πλαίσια της ενότητας αυτής η κύρια βιβλιογραφική μας αναφορά είναι η εισαγωγική εργασία [44] του Shoup, αφού δυστυχώς υπάρχει ελάχιστη μελέτη τους σε βιβλία.

Πριν μπούμε στις λεπτομέρειές και τη μεθοδολογία απόδειξης ας σταθούμε στην έννοια του "παιχνιδιού". Η έννοια αυτή στον κλάδο της Ασφάλειας έχει πολλές ερμηνείες. Συγκεκριμένα ο όρος "παιχνίδι", πέρα από τον κλάδο της κρυπτογραφίας χρησιμοποιείται στον κλάδο της Ποσοτικής Ασφάλειας (Quantitative Security) και είναι άμεσα συσχετισμένος με τον αντίστοιχο όρο της Θεωρίας Παιγνίων. Στη συγκεκριμένη περίπτωση που θα αναφερθούμε, πρόκειται για ένα κρυπτογραφικό "παιχνίδι" και είναι ασυσχέτιστο με την έννοια "παιχνίδι" της Ποσοτικής Ασφάλειας και της Θεωρίας Παιγνίων [45] Είναι μια θεωρητική έννοια που εκφράζεται ως ένας αλγόριθμος, αλλά θα μπορούσε να έχει τη φυσική μορφή μιας δημόσιας διαμάχης (public debate). Στο παιχνίδι συμμετέχουν δύο παίχτες, ένας προκαλών (challenger) $\mathcal{C}$και ένας αντίπαλος (adversary) $\mathcal{A}$. Άρα αν θέλαμε να το ορίσουμε πιο αυστηρά, ένα παιχνίδι πρόκειται για έναν κατανεμημένο αλγόριθμο με ένα πρωτόκολλο επικοινωνίας που χρησιμοποιούν οι παίκτες. Ο αλγόριθμος που ακολουθεί ο προκαλών είναι αυστηρά ορισμένος ανάλογα με την ιδιότητα της ασφάλειας που θέλουμε να αποδείξουμε (π.χ. Σημασιολογική Ασφάλεια ενάντια σε Επιθέσεις Επιλεγμένων Κρυπτοκειμένων κτλ.). Αυτό εξασφαλίζει ότι η πρόκληση που ο προκαλών θα δημιουργήσει για τον αντίπαλο είναι σωστά δημιουργημένη. Παρόμοια, το πρωτόκολλο με το οποίο αλληλεπιδρούν οι παίκτες στα πλαίσια του παιχνιδιού είναι και αυτό αυστηρά ορισμένο και καθορίζεται επίσης από την ιδιότητα ασφάλειας που θέλουμε να αποδείξουμε. Αντιθέτως, ο αντίπαλος ο αντίπαλος έχει απόλυτη ελευθερία σχετικά με τους υπολογισμούς που θα εκτελέσει, αρκεί μόνο να συμβαδίζει το μοντέλο αντιπάλου που έχουμε υποθέσει (π.χ. Αντίπαλος με υπολογιστικά φραγμένους πόρους, Ενεργητικός ή Παθητικός αντίπαλος κτλ.). Έτσι, συνήθως τον αντίπαλο σχετικά με τον εσωτερικό αλγόριθμο που ακολουθεί, τον αντιμετωπίζουμε σαν μαύρο κουτί (black box) το οποίο ακολουθεί το πρωτόκολλο του παιχνιδιού για την επικοινωνία με τον προκαλών. Ανάλογα με την ιδιότητα ασφάλειας που θέλουμε να αποδείξουμε στα πλαίσια του παιχνιδιού εκτελούνται διάφοροι γύροι υπολογισμού και επικοινωνίας μεταξύ των δύο παικτών, ωστόσο οι τελευταίοι γύροι είναι κοινοί σε όλες τις περιπτώσεις. Ο προκαλών δημιουργεί μια πρόκληση που καλείται ο αντίπαλος να απαντήσει. Στο τελευταίο και πιο σημαντικό βήμα, προσπαθούμε να αποδείξουμε ότι η τυχαία μεταβλητή της απάντησης του αντιπάλου ακολουθεί μια επιθυμητή

κατανομή που ορίζεται αυστηρά από την ιδιότητα της ασφάλειας που επιθυμούμε να αποδείξουμε και από το είδος του παιχνιδιού, γιατί μπορεί για μια ιδιότητα να υπάρχουν ισοδύναμα παιχνίδια. Τέλος, πρέπει να είναι ξεκάθαρο στον αναγνώστη ότι όταν αναφέρουμε "αυστηρά ορισμένο" σητν συγκεκριμένη περίπτωση, αναφερόμαστε στον ορισμό ή σε ισοδύναμους του, που υπάρχουν στη βιβλιογραφία για την συγκεκριμένη ιδιότητα ασφάλειας.

3.4.2.1 Απόδειξη Σημασιολογικής Αφάλειας Hashed ElGamal με την Μέθοδο Μεταπήδησης μεταξύ Παιχνιδιών

Για να κατανοήσουμε τα παραπάνω, θα αναλύσουμε την απόδειξη της ασφάλειας ενός κρυπτοσυστήματος μέσω παιχνιδιού με τη βοήθεια ενός παραδείγματος. Ας πάρουμε για παράδειγμα το κρυπτοσύστημα δημόσιου κλειδιού Hashed ElGamal [44], το οποίο είναι μια παραλλαγή του γνωστού κρυπτοσυστήματος δημοσίου (ασύμμετρου) κλειδιού ElGamal η οποία αντιμετωπίζει τα μηνύματα ως συμβολοσειρές από bit (bitstrings) αντί για στοιχεία ομάδας. Θα αποδείξουμε την ασφάλεια του κρυπτογραφικού σχήματος Hashed ElGamal μέσω παιχνιδιού. Το κρυπτογραφικό σχήμα Hashed ElGamal ορίζεται όπως στο Σχήμα 3.4.1.

Definition 3.4.1. Ασύμμετρο κρυπτογραφικό σχήμα Hashed ElGamal : Έστω μια πολλαπλασιαστική ομάδα G πρώτης τάξης q, την συμβολίζουμε με $\mathbb{Z}_q$ αφού γνωρίζουμε ότι όλες οι ομάδες πολλαπλασιαστικές ομάδες πρώτης τάξης είναι ισομορφικές σε αυτή. Οι κατάλληλοι χώροι (K, M, C) από τον Ορισμό 2.1.4 είναι οι $(Z_q, \{0,1\}^l, \{0,1\}^l)$. Έστω επίσης μια οικογένεια κρυπτογραφικών συναρτήσεων κατακερματισμού με κλειδί $H := \{H_k\}_{k \in K}$ συνάρτηση κατακερματισμού $H : G \to \{0,1\}^l$. Οι αλγόριθμοι $(\text{KGen}, \text{Enc}, \text{Dec})$ ορίζονται ως εξής :

$\text{KGen}(1^n)$	$\text{Enc}_{pk}(m)$	$\text{Dec}_{sk}(c)$
1 : $x \leftarrow\$ \mathbb{Z}_q$	1 : $y \leftarrow\$ \mathbb{Z}_q$	1 : $m \leftarrow H_k(c_1^x) \oplus c_2$
2 : $k \leftarrow\$ K$	2 : $s \leftarrow a^y$	2 : **return** m
3 : $a \leftarrow g^x$	3 : $h \leftarrow H_k(s)$	
4 : $pk \leftarrow (x, h)$	4 : $c_1 \leftarrow g^y$	
5 : $sk \leftarrow (x, k)$	5 : $c_2 \leftarrow h \oplus m$	
6 : **return** (pk, sk)	6 : $c \leftarrow (c_1, c_2)$	
	7 : **return** c	

Σκοπός μας είναι να αποδείξουμε ότι το σχήμα Hashed ElGamal είναι Σημασιολογικά Ασφαλές (Semantically Secure) υπό την υπόθεση του Αποφασιστικού Diffie Hellman (Decisional Diffie Hellman ή DDH) και υπό την υπόθεση ότι η $H := \{H_k\}_{k \in K}$ είναι μια οικογένια κρυπτογραφικών συναρτήσεων κατακερματισμού που διαθέτει την ιδιότητα της Εξομάλνσης Εντροπίας (Entropy Smoothing). Η ιδιότητα αυτή ορίζεται εξής :

Definition 3.4.2. Υπόθεση Εξομάλυνσης Εντροπίας (Entropy Smoothing Assumption) για Κρυπτογραφικές Συναρτήσεις Κατακερματισμού : Έστω μια ομάδα G και $H := \{H_k\}_{k\in K}$ όπου $H_k : G \to \{0,1\}^l$, μια οικογένια κρυπτογραφικών συναρτήσεων κατακερματισμού με κλειδί. Η H διαθέτει την ιδιότητα αυτή αν, και μόνο αν :

$$Pr[k \leftarrow\$ K, s \leftarrow\$ G : D(k, H_k(s)) = 1]- \tag{3.6}$$

$$Pr[k \leftarrow\$ K, h \leftarrow\$ bin^l : D(k,h) = 1] \tag{3.7}$$

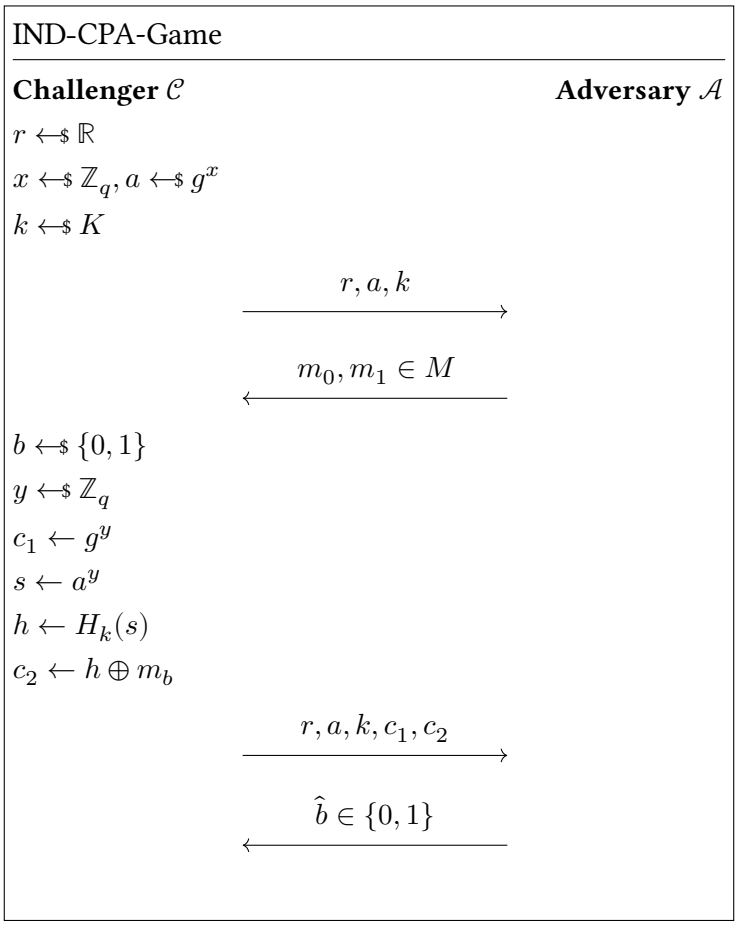
$$\leq negl(1^n) = \varepsilon_{ES} \tag{3.8}$$

Στην παρακάτω απόδειξη γίνεται χρήση της Μεθόδου Μεταπήδησης μεταξύ Παιχνιδιών, την οποία θα αναλύσουμε λεπτομερώς στην επόμενη ενότητα, ωστόσο πιστεύουμε ότι το παράδειγμα αυτό θα βοηθήσει στην πιο διαισθητική αντίληψη της ώστε να μπορέσουμε στην συνέχεια να της ορίσουμε πιο αυστηρά. Θα προσπαθήσουμε ξεκινώντας από τον αρχικό ορισμό του Παιχνιδιού της Δυσδιακριτότητας για το Hashed ElGamal και χρησιμοποιώντας τη Μεθόδο της Μεταπήδησης μεταξύ Παιχνιδιών να καταλήξουμε σε ένα παιχνίδι του οποίου το πλεονέκτημα ενός αντιπάλου είναι ισοδύναμο με αυτό ενός αντιπάλου στο Πρόβλημα DDH.

Ας ξεκινήσουμε με τον ορισμό του παιχνιδιού της του Παιχνιδιού της Δυσδιακριτότητας στην περίπτωση του Hashed ElGamal κρυπτογραφικού σχήματος, οποίος φαίνεται στο Σχήμα 3.1. Έστω S_0 το γεγονός $b = \hat{b}$ στο Παιχνίδι 0. Παρατηρούμε ότι το $h = H_k(g^{xy})$, το οποίο μπορεί να κατασκευαστεί από κάποιον που γνωρίζει το ιδιωτικό κλειδί αφού x και το κρυπτοκείμενο περιέχει το $c_1 = g^y$ και άρα μπορεί να υπολογίσει το h. Ορίζουμε τώρα μια παραλλαγή του Παιχνιδιού 0, στο οποίο $h = H_k(g^z)$, όπου το $z \leftarrow\$ \mathbb{Z}_q$. Το Παιχνίδι 1 φαίνεται στο Σχήμα 3.2, στο οποίο είναι σκιασμένες οι διαφορές του από το προηγούμενο παιχνίδι.

Έστω S_1 το γεγονός $b = \hat{b}$ στο Παιχνίδι 1. Θέλουμε να δείξουμε ότι $Pr[S_0] - Pr[S_1] = \varepsilon_{DDH}$, όπου $\varepsilon_D DH$ είναι το πλεονέκτημα που έχει οποιοσδήποτε πολυωνυμικός αντίπαλος στο πρόβλημα DDH. Δημιουργούμε το Υβριδικό Παιχνίδι 1-2, το οποίο φαίνεται στο Σχήμα 3.3 και συνδυάζει τα Παιχνίδια 1 και 2, για αυτό το αποκαλούμε **Υβριδικό Παιχνίδι**. Το Υβριδικό Παιχνίδι 0-1 φαίνεται στο Σχήμα 3.4 και γενικότερα η Μέθοδος Υβριδικού Παιχνιδιού ορίζεται ως εξής :

Definition 3.4.3. Μέθοδος Υβριδικού Παιχνιδιού Είναι ένας αλγόριθμος που χρησιμοποιείται για να συνδυάσει δύο παιχνίδια, έστω το Παιχνίδι i και Παιχνίδι j. Το υβριδικό παιχνίδι έχει την δυνατότητα να εκτελεί είτε το Παιχνίδι i είτε το Παιχνίδι j ανάλογα με τις εισόδους που του δίνονται. Συνήθως χρησιμοποιείτε για να δείξει με συνοπτικό και κατανοητό τρόπο ότι τα πλεονεκτήματα των αντιπάλων των δύο παιχνιδιών ικανοποιούν κάποια ιδιότητα.

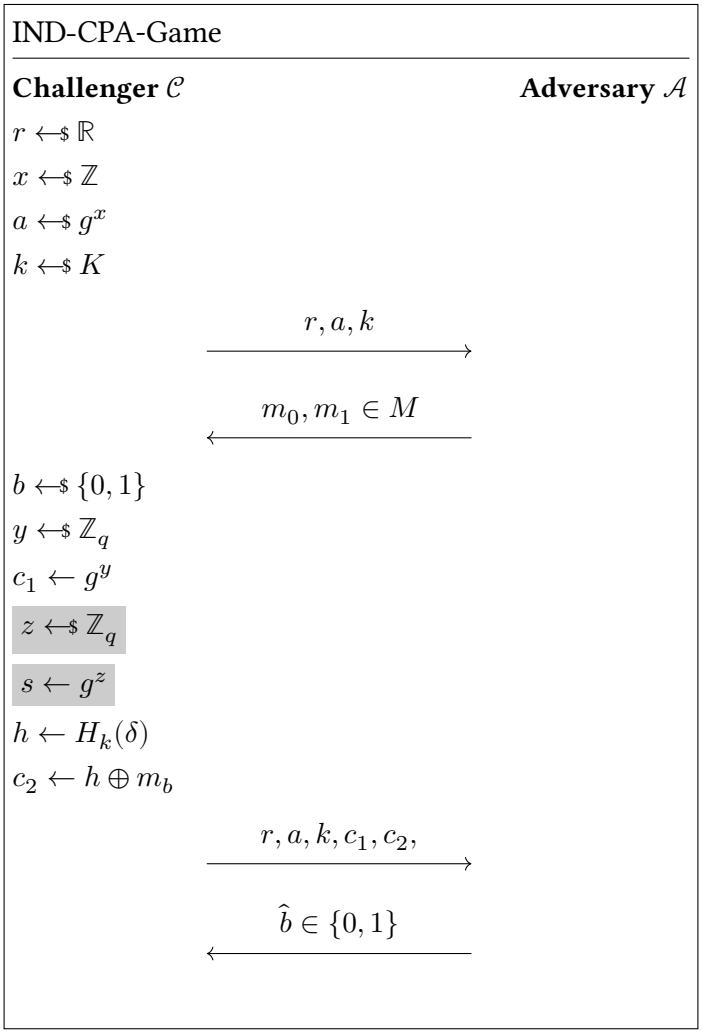


Σχήμα 3.1: Hashed ElGamal : Παιχνίδι 0

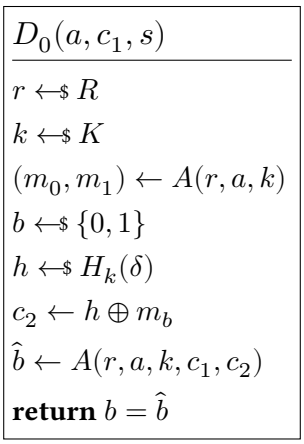
Για τον αλγόριθμο D_0 του Υβριδικού Παιχνιδιού 0-1, είναι προφανές ότι στην περίπτωση όπου $a \leftarrow g^x, c_1 \leftarrow g^y, s \leftarrow g^{xy}$ τότε εκτελείτε το Παιχνίδι 1, ενώ όταν $a \leftarrow \gamma^x, c_1 \leftarrow g^y, s \leftarrow g^z$ εκτελείται το Παιχνίδι 2. Ουσιαστικά αυτό που θέλουμε να δείξουμε μέσω του υβριδικού παιχνιδιού είναι ότι ο προκαλών $\mathcal{C}$ στα Παιχνίδια 1 και 2 μπορεί να αντικατασταθεί από τον υβριδικό αλγόριθμο και η πιθανότητα να γίνει αντιληπτός από τον αντίπαλο $\mathcal{A}$ είναι ίδια με αυτή του προβλήματος DDH, δηλαδή το πλεονέκτημα του αντιπάλου στο Παιχνίδι 2 σε σχέση με το Παιχνίδι 1 είναι ισοδύναμο με την πιθανότητα επίλυσης του προβλήματος DDH. Έτσι αποδεικνύουμε το εξής :

$$|Pr(S_0) - Pr(S_1)| = \varepsilon_{DDH} \tag{3.9}$$

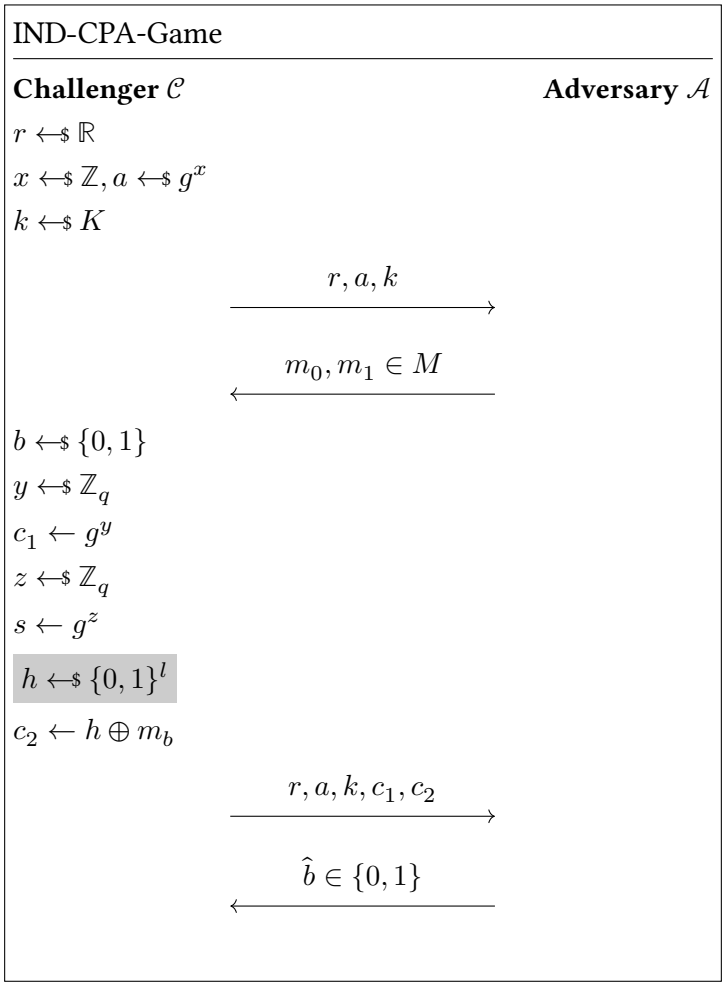
Ορίζουμε τώρα μια παραλλαγή του Παιχνιδιού 1, στο οποίο $h = H_k(g^z)$, όπου το z προέρχεται από ομοιόμορφη δειγματοληψία του $\mathbb{Z}_q$, με το Παιχνίδι 2, που φαίνεται στο Σχήμα 3.4 στο οποίο αντικαθηστούμε την h με μια ομοιόμορφα τυχαία τιμή από το πεδίο τιμών της συνάρτησης H_k. Σε αντιστοιχία με τα προηγούμενα παιχνίδια ορίζουμε S_2 το γεγονός $b = \hat{b}$



Σχήμα 3.2: Hashed ElGamal : Παιχνίδι 1



Σχήμα 3.3: Hash ElGamal : Υβριδικό Παιχνίδι 0-1



Σχήμα 3.4: Hashed ElGamal : Παιχνίδι 2

στο Παιχνίδι 2. Είναι προφανές ότι η h παίζει τον ρόλο του OTP για την τιμή και άρα λόγω της απόλυτης ασφάλειας που μας παρέχει ένα OTP σχήμα έχουμε το εξής :

$$Pr[S_2] = 1/2 \tag{3.10}$$

Ο σκοπός μας είναι να αποδείξουμε ότι :

$$|Pr(S_1) - Pr(S_2)| = \varepsilon_{ES} \tag{3.11}$$

Για να το αποδείξουμε αυτό δημιουργούμε το Υβριδικό Παιχνίδι 1-2 που φαίνεται στο Σχήμα 3.5. Είναι προφανές ότι ο αλγόριθμος D_1 παρεμβάλεται μεταξύ των πιθανοτικών κατανομών του Ορισμού 3.4.2 και άρα από τον ορισμό αυτόν αποδεικνύεται η σχέση αυτή.

Συνδυάζοντας τις σχέσεις 3.9, 3.11 και 3.10 μέσω της τριγωνικής ανισώτητας καταλήγουμε σε αυτό που θέλαμε αρχικά να αποδείξουμε, δηλαδή ότι :

$D_1(k, h)$

$r \leftarrow\$ R$
$a \leftarrow g^x$
$k \leftarrow\$ K$
$(m_0, m_1) \leftarrow A(r, a, k)$
$b \leftarrow\$ 0, 1$
$y \leftarrow\$ \mathbb{Z}_q c_1 \leftarrow g^y$
$c_2 \leftarrow h \oplus m_b$
$\hat{b} A(r, a, k, c_1, c_2)$
return $b = \hat{b}$

Σχήμα 3.5: Hashed ElGamal : Υβριδικό Παιχνίδι 1-2

$$|Pr[S_0] - 1/2| \leq \varepsilon_{DDH} + \varepsilon_{ES} = negl1^n \tag{3.12}$$

3.4.2.2 Μέθοδος Μεταπήδησης μεταξύ Παιχνιδιών

Στην απόδειξη του Hashed ElGamal στην παραπάνω ενότητα παρατηρούμε ότι για κάθε νέο παιχνίδι που ορίζαμε είναι υπολογιστικά δυσδιάκριτο από το προηγούμενο, με απότερο σκοπό να δείξουμε ότι το πρώτο και το τελευταίο παιχνίδι είναι υπολογιστικά δυσδιάκριτα. Η Μέθοδος Μεταπήδησης μεταξύ Παιχνιδιών [44] χρησιμοποιείται για να απλοποιήσει την πολυπλοκότητα των αποδείξεων στην περίπτωση που έχουμε αρκετά σύνθετα πρωτόκολλα και μας επιτρέπει να κάνουμε πιο σύνθετες μεταβάσεις από το ένα παιχνίδι στο άλλο χωρίς να παραβιάζουμε την ασφάλεια. Πιο συγκεκριμένα στην [44] αναφέρονται τρία είδη μεταπήδησης μεταξύ παιχνιδιών, ωστόσο η περαιτέρω ανάλυση των ειδών μεταπήδησης, πλην της μεταπήδησης βασισμένης στην δυσδιακριτότητα που είδη αναλύσαμε, εμπίπτει εκτός του σκοπού αυτής της εργασίας:

Είδη Μεταπήδησης μεταξύ παιχνιδιών :

- Μεταπήδηση βασισμένη στην δυσδιακριτότητα (Hopping based on indistinguishability)
- Μεταπήδηση βασιμένη σε γεγονότα αποτυχίας (Hopping based on failure events)
- Μεταπήδηση βασισμένη σε βήματα γεφύρωσης (Hopping based on bridging steps)

3.4.3 Απόδειξη ασφάλειας μέσω προσομοίωσης (Simulation based security proof)

Η απόδειξη ασφάλειας ενός παιχνιδιού προσεγγίζει τον ορισμό της ασφάλειας απο διαφορετική σκοπιά. Για να κατανοήσουμε αυτή τη σκοπία πρέπει να επανέλθουμε στον ορισμό της ασφάλειας. Ας πάρουμε για παράδειγμα ένα αυθαίρετο συμμετρικό κρυπτογραφικό σχήμα. Θα λέγαμε ότι ένα τέτοιο σχήμα είναι ασφαλές αν όλες οι δημόσιες πληροφορίες, στην προκείμενη περίπτωση το κρυπτοκείμενο και πιθανόν το μέγεθος του μπλοκ[1], δεν δίνουν καμία γνώση παραπάνω σε έναν αντίπαλο σε σύγκριση με αυτήν που είδη κατέχει, την οποία μπορεί να έχει αποκτήσει από άλλα μέσα. Δηλαδή η a priori πληφορία του δεν μεταβάλεται αφού γνωρίσει τις δημόσιες πληροφορίες του σχήματος. Το παραπάνω μπορούμε να το σκεφτούμε ως εξής. Για να αποδείξουμε ότι ένα κρυπτογραφικό σχήμα είναι ασφαλές με αυτή την μέθοδο θεωρούμε δύο κόσμους, έναν Πραγματικό και έναν Ιδανικό. Στους δύο αυτούς κόσμους όλοι οι συμμετέχοντες του κρυπτογραφικού σχήματος είναι ίδιοι εκτός από αυτούς που ελέγχει ο αντίπαλος. Στον Ιδανικό κόσμο θεωρούμε πως εκτελείται μια εξιδανικευμένη μορφή του κρυπτογραφικού σχήματος και άρα ο αντίπαλος δεν λαμβάνει καμία επιπλέον γνώση παρά μόνο τις παραμέτρους του συστήματος, ενώ στον Πραγματικό κόσμο θεωρούμε ότι εκτελείται αυτούσιο το κρυπτογραφικό σχήμα και άρα ο αντίπαλος λαμβάνει επιπλέον γνώση, όπως το κρυπτοκείμενο και το μήκος του μπλοκ. Στην περίπτωση του συμμετρικού σχήματος που αναφέραμε, το πρωτόκολλο στον ιδανικό κόσμο θα μπορούσε να ήταν το OTP ενώ στην περίπτωση ενός σύστηματος ψηφοφορίας θα μπορούσε να ήταν, θα ήταν μια εξειδανικευμένη μορφή του συστήματος αυτού θα ήταν να έχουμε ένα Έμπιστο Τρίτο Άτομο (Trusted Third Party ή TTP) στο οποίο κάθε συμμετέχον στέλνει την ψήφο του, ο TTP τις καταμετράει και ανακυρήσει τον νικητή της ψηφοφορίας σε όλους τους συμμετέχοντες. Για να αποδειχθεί ένα σύστημα ασφαλές με αυτή την μέθοδο θα πρέπει για κάθε πιθανό αντίπαλο του πραγματικού κόσμου να υπάρχει ένας αντίπαλος στον ιδανικό κόσμο για τους οποίους αν συγκρίνουμε όλη την γνώση που κατέχουν να είναι δυσδιάκριτες. Για τις αποδείξεις ασφάλειας μέσω προσωμοίωσης η βασική μας βιβλιογραφική αναφορά είναι η εργασία [46], διότι δυστυχώς παρότι είναι ευρέως διαδεδομένη στον κλάδο της κρυπτογραφίας δεν καταφέραμε να βρούμε σημαντικές μελέτες και αναλύσεις τις σε βιβλία ή σε εκτενέστερες εργασίες. Σε προηγούμενη ενότητα δώσαμε τον ορισμό της σημασιολογικής ασφάλειας μέσω παιχνιδιού. Ενώ αυτός εκείνος ο ορισμός μπορούσε να χρησιμοποιηθεί για ασύμμετρα αλλά και συμμετρικά σχήματα, δεν ισχύει το ίδιο με την περίπτωση της προσωμοίωσης, καθότι το κάθε σχήμα διαθέτει διαφορετική φύση και διαφορετικές πληροφορίες γνωρίζει σε κάθε περίπτωση ο αντίπαλος. Για παράδειγμα ο αντίστοιχος ορισμός της υπολογιστικής ασφάλειας για ένα συμμετρικό κρυπτογραφικό σχήμα είναι ο παρακάτω :

[1]Πρακτικά συνήθως λαμβάνει πολύ περισσότερες πληροφορίες, ωστόσο θεωρούμε ένα απλό παράδειγμα

Definition 3.4.4. Υπολογιστική Ασφάλειας μέσω Προσωμοίωσης για Συμμετρικά Κρυπτογραφικά σχήματα : Έστω συμμετρικό κρυπτογραφικό σχήμα (KGen, Enc, Dec). Το σχήμα αυτό είναι ασφαλές υπό τον ορισμό αυτό αν για κάθε μη-ομοιόμορφο αλγόριθμο $\mathcal{A} \in PPT$, υπάρχει $\mathcal{A}' \in PPT$, τέτοιος ώστε για κάθε πιθανοτική ακολουθία $\{X\}_{n\in\mathbb{N}}$ με $|X_n| \leq poly(1^n)$ και για κάθε ζεύγος (υπολογιστικά) πολυωνιμικά φραγμένων συναρτήσεων f, h : $\{0,1\}^* \to \{0,1\}^*$:

$$|\Pr_{k\leftarrow \text{KGen}(1^n)}[\mathcal{A}(1^n, E_k(X_n), 1^{|X_n|}, h(1^n, X_n)) = f(1^n, X_n)] - \Pr[\mathcal{A}'(1^n, 1^{|X_n|}, h(1^n, X_n)) = f(1^n, X_n)]| \leq negl(1^n)$$

Στο παραπάνω ορισμό, ο αντίπαλος $\mathcal{A}$ είναι αυτός του πραγματικού κόσμου ενώ ο $\mathcal{A}'$ είναι αυτός του ιδανικού κόσμου. Η διαφοροποίηση τους ως προς την πληροφορία είναι σκιασμένη και είναι το κρυπτοκείμενο $E_k(X_n)$. Σχετικά με τις υπόλοιπες παραμέτρους η παράμετρος 1^n είναι η παράμετρος ασφάλειας, η $1^{|X_n|}$ είναι το μήκος του μηνύματος, η συνάρτηση $h(1^n, X_n)$ είναι μια πολυωνυμική συνάρτηση που δίνει ως έξοδο την a priori γνώση, ωστόσο και οι δύο αντίπαλοι έχουν πρόσβαση μόνο στο αποτέλεσμα της συνάρτησης αυτής όχι και στα ορίσματα της, και η συνάρτηση $f(1^n, X_n)$ είναι η συνάρτηση τελική πληφοροία που γνωρίζει ο κάθε αντίπαλος.

Ένας περίεργος αναγνώστης μπορεί να αναρωτιώται από που προκύπτει το όνομα "Προσομοίωση" αφού δεν έχει αναφερθεί μέχρι στιγμής και ούτε ο παραπάνω ορισμός παραπέμπει σε κάτι τέτοι. Ας απομονώσουμε την εξής φράση από τον παραπάνω ορισμό, "για κάθε $\mathcal{A}$, υπάρχει $\mathcal{A}'$". Αυτό σημαίνει ότι για οποιονδήποτε ρεαλιστικό αντίπαλο πρέπει να μπορούμε να κατασκευάσουμε έναν ιδανικό αντίπαλο. Ωστόσο, παρατηρούμε ότι ο $\mathcal{A}'$ δεν γνωρίζει τι εκτελεί εσωτερικά ο $\mathcal{A}$. Για παράδειγμα μπορεί ο $\mathcal{A}$ να δίνει ως έξοδο μια συγκεκριμένη τιμή, π.χ. 0. Χωρίς κάποια παραπάνω πληροφορία σχετικά με το τι εκτελεί ο $\mathcal{A}$, είναι αδύνατο να μπορούμε να κατασκευάσουμε $\mathcal{A}'$ για κάθε $\mathcal{A}$. Για να λύσουμε το πρόβλημα αυτό, δίνουμε στον αντίπαλο του ιδανικού κόσμου πρόσβαση μαντείου σε αυτόν του πραγματικού κόσμου. Μπορεί πλέον να γίνει κατανοητή η ονομασία της "Προσομοίωσης" και για τον λόγο συνήθως αποκαλούμε τον $\mathcal{A}'$ **Προσομοιωτή**. Παρακάτω παρουσιάζεται ένας κατασκευαστικός αλγοριθμος για τον $\mathcal{A}'$.

$\mathcal{A}'(1^n, \text{E}_k(X_n), 1^{\lvert X_n\rvert}, h(1^n, X_n))$	
1 :	$k \leftarrow \text{KGen}(1^n)$
2 :	$c \leftarrow \text{Enc}_k(0^{\lvert X\rvert_n})$
3 :	**return** $\mathcal{A}(1^n, c, 1^{\lvert X\rvert_n}, h)$

3.4.3.1 Απόδειξη ασφάλειας στο Πρωτόκολλο Ανυποψίαστης Μεταφοράς 1/2 (Oblivious Transfer 1/2)

Στην ενότητα αυτή θα παρουσιάσουμε την απόδειξη της ασφάλειας υπό προσωμοίωση, του πρωτοκόλλου ανυποψίαστης μεταφοράς [34], OT-1/2 που παρουσιάζεται στον Ορισμό 2.2.5 στο Κεφάλαιο 2. Ο αναγνώστης που δεν έχει επίγνωση του πρωτοκόλλου αυτού, προκειμένου να αντιληφθεί την επόμενη απόδειξη, μπορεί να ανατρέξει απευθείας στην ενότητα αυτή του προφαναφερθείς κεφαλαίουτο παρουσιάζουμε. Στην συνέχεια θα παρουσιαστεί η απόδειξη απόδειξη ασφάλειας μέσω προσομοίωσης ενάντια σε παθητικούς υπολογιστικά περιορισμένους αντιπάλους του πρωτοκόλλου αυτού.

Όπως προαναφέρθηκε ο ορισμός της ασφάλειας μέσω προσωμοίωσης, δηλαδή οι σχέσεις που πρέπει να ικανοποιούνται όπως στον Ορισμό 3.4.4, είναι διαφορετικός για κάθε κρυπτογραφικό σχήμα που εξετάζουμε διότι πρέπει να λάβουμε υπόψη όλη τη δημόσια γνώση που μπορεί να αποκτήσει κάποιος αντίπαλος. Έτσι για ένα πρωτόκολλο δύο συμμετεχόντων ο ορισμός της ασφάλειας μέσω προσομοίωσης ενάντια σε παθητικούς υπολογιστικά περιορισμένους αντιπάλους είναι ο παρακάτω :

Definition 3.4.5. Ασφάλεια μέσω προσομοίωσης ενάντια σε παθητικούς αντιπάλους για πρωτόκολλο δύο συμμετεχόντων : Υποθέτουμε τα εξής :

- Έστω δύο συμμετέχοντες P_1 και P_2 όπου ο καθένας υπολογίζει μια συνάρτηση f_1 και f_2 αντίστοιχα με ιδιωτικές είσόδους $x, y \in \{0,1\}^*$. Έστω η τελική συνάρτηση που υπολογίζει το πρωτόκολλο είναι η $f = (f_1(x,y), f_2(x,y))$ και έστω ένα πρωτόκολλο π που την υπολογίζει την f.

- Έστω $\text{view}_i^\pi(x,y,n)$ η ανεπεξέργαστη πληροφορία που έχει είδε κατά την εκτέλεση του ο i συμμετέχοντας για το πρωτόκολλο π εκτελούμενο με εισόδους x, y και υπολογιστική παράμετρο n. Η πληροφορία αυτή αποτελείται από την ιδιωτική του είσοδο w, την εσωτερική του μνήμη r_i και τα μηνύματα που έχει λάβει $\{m_1^i, \dots, m_t^i\}$ (υποθέτωντας ότι όσα έχει στείλει είναι αποθηκευμένα σητν εσωτερική του μνήμη), δηλαδή $\text{view}_i^\pi(x,y,n) = (w, r^i; m_1^i, \dots, m_t^i)$.

- Έστω $\text{output}_i^\pi(x,y,n)$ η έξοδος του συμμετέχον i και

$$\text{output}^\pi(x,y,n) = (\text{output}_1^\pi(x,y,n), \text{output}_2^\pi(x,y,n))$$

  η έξοδος του πρωτοκόλλου που για να είναι ορθό το πρωτόκολλο θα πρέπει να ισούται με $(f_1(x,y), f_2(x,y))$.

Λέμε ότι ένα ορθό πρωτόκολλο π για την f είναι ασφαλές υπό προσομοίωση ενάντια σε υπολογοσιτικά περιορισμένους παθητικούς αντιπάλους αν, και μόνο αν, υπάρχουν $S_1, S_2 \in$

PPT τέτοιοι ώστε:

$$\{(\mathcal{S}_1(1^n, x, f_1(x,y)), f(x,y))\}_{x,y,n} \stackrel{c}{=} \{(\text{view}_1^{\pi}(x,y,n),\ \text{output}^{\pi}(x,y,n))\}_{x,y,n}, \text{ και} \tag{3.13}$$

$$\{(\mathcal{S}_2(1^n, y, f_2(x,y)), f(x,y))\}_{x,y,n} \stackrel{c}{=} \{(\text{view}_2^{\pi}(x,y,n),\ \text{output}^{\pi}(x,y,n))\}_{x,y,n} \tag{3.14}$$

, όπου $x, y \in \{0,1\}^*$ με $|x| = |y|$ και $n \in \mathbb{N}$

Σκοπός μας είναι να αποδείξουμε τις Σχέσεις 3.14 υπό το μοντέλο ασφάλειας που ορίσαμε προηγουμένως. Για να το επιτύχουμε αυτό πρέπει να εξετάσουμε δύο περιπτώσεις, αυτή που ο P_1 είναι διεφαρμένος και αυτή που είναι ο P_2 διευθαρμένος. Προφανώς δεν έχει κανένα νόημα να εξετάσουμε την περίπτωση που και οι δύο είναι διευθαρμένοι και ελέγχονται από τον αντίπαλο.

Θα ξεκινήσουμε με την πιο απλή περίπτωση, αυτή που ο P_1 είναι διευθαρμένος, αφού ο P_1 δεν διαθέτει κάποια έξοδο και άρα ο S_1 αρκεί να μπορεί να προσομοιώσει το view_1^{π}. Κατασκευάζουμε τον προσομοιωτή S_1 ως εξής :

$\text{S}_1((b_0,b_1), 1^n, r)$

1 : $(a,t) \leftarrow I(1^n, r)$

2 : $r_1, r_2 \leftarrow\!\!\$\ r$

3 : $y_0 \leftarrow S(a, r_1), y_1 \leftarrow S(a, r_2)$

4 : **return** $((b_0,b_1), r, (y_0,y_1))$

Στην περίπτωση του P_1 έχουμε $\text{view}_1^{\pi} = ((b_0,b_1), r, (y_0,y_1))$ όπου ισχύει $(y_0,y_1) = (S(a), F(a,S(a)))$ ή $(y_0,y_1) = (F(a,S(a)), S(a))$ ανάλογα με το σ που επιλέγει ο P_1. Προφανώς από τον ορισμό της Επαυξημένης Μετάθεσης Κερκόπορτας γνωρίζουμε ότι :

$$F(a,S(a))s \equiv S(a)$$

και άρα αποδεικνύεται ότι στην περίπτωση του διευθαρμένου P_1 το σχήμα μας είναι ασφαλές.

Ας εξετάσουμε τώρα την περίπτωση που ο P_2 είναι διαυθαρμένος. Όπως προαναφέραμε, ο προσομοιωτής S_2 πρέπει να έχει πρόσβαση στις εισόδους και στις εξόδους του P_2, δηλαδή να τις παίρνει ως ορίσματα. Κατασκευάζουμε τον προσομοιωτή S_2 ως εξής :

$S_2(\sigma, b_\sigma, 1^n, r)$

1 : $(a,t) \leftarrow I(1^n, r)$

2 : $x_\sigma \leftarrow S(a, r_1), x_{1-\sigma} \leftarrow F^{-1}(t, S(a, r_2))$

3 : $\beta_\sigma \leftarrow B(a, x_\sigma) \oplus b_\sigma$

4 : $\beta_{1-\sigma} \leftarrow B(a, x_{1-\sigma})$

5 : **return** $(\sigma, r_0, r_1; a, (\beta_0, \beta_1))$

Απλά παρατηρώντας το πρωτόκολλο μπορούμε να δούμε πως η όψη του σχήματος που βλέπει ο P_2 είναι η εξής :

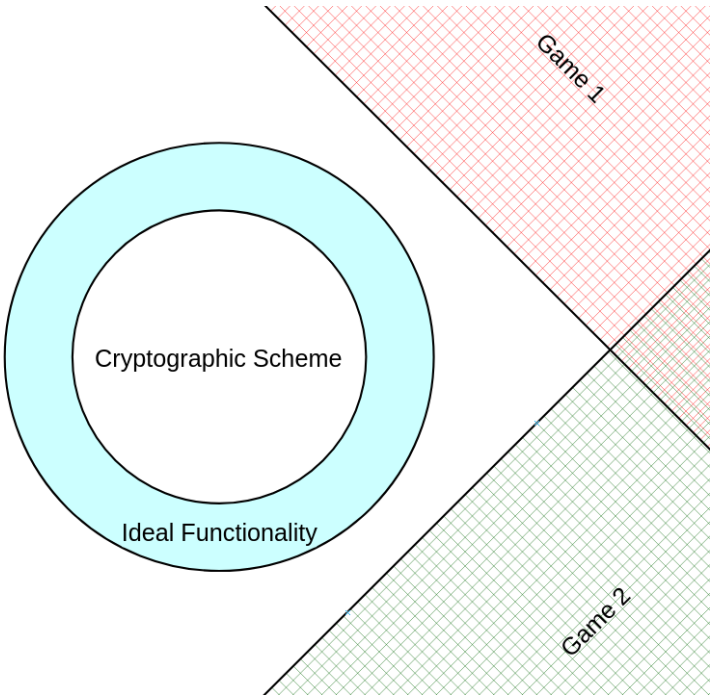
$$\text{view}_2^\pi\left((b_0, b_1), \sigma\right) = (\sigma, r_0, r_1; \alpha, (B(\alpha, x_\sigma) \oplus b_\sigma, B(\alpha, x_{1-\sigma}) \oplus b_{1-\sigma}))$$

ενώ του S_2 είναι η εξής :

$$\mathcal{S}_2\left(1^n, \sigma, b_\sigma\right) = (\sigma, r_0, r_1; \alpha, (B(\alpha, x_\sigma) \oplus b_\sigma, B(\alpha, x_{1-\sigma})))$$

3.4.4 Σύγκριση μεθόδων απόδειξης ασφάλειας

Οι δύο κυριότερες μεθόδοι αποδείξεων θεωρητικής ασφάλειας μελετήθηκαν στις δύο προηγούμενες ενότητες, αυτή της Απόδειξης μέσω Παιχνιδιών και αυτή της Απόδειξης μέσω Προσομοίωσης. Η δεύτερη είναι πολύ πιο ισχυρή από την πρώτη και αυτό γίνεται πιο διαισθητικά αντιληπτό από το Σχήμα 3.6. Ωστόσο, όπως έγινε αντιληπτό και από τα παραδείγματα που μελετήσαμε, η απόδειξη μέσω προσομοίωσης είναι πολύ πιο σύνθετη για απλά κρυπτογραφικά σχήματα σε σχέση με την απόδειξη μέσω παιχνιδιών, ωστόσο τα πλεονεκτήματα της εκτιμούνται περισσότερο σε πιο σύνθετα σχήματα με παραπάνω από δύο συμμετέχοντες, όπως για παράδειγμα σύνθετα πρωτόκολλα επικοινωνίας, αποδείξης μηδενικής γνώσης κτλ. [47]



Σχήμα 3.6: Σχηματική αναπαράσταση Απόδειξης ασφάλειας μέσω παιχνιδιού και μέσω προσομοίωσης

Κεφάλαιο 4

Ασφαλής Υπολογισμός Πολλών Μερών

Στην ενότητα αυτή θα γίνει μια εισαγωγή στον Ασφαλή Υπολογισμό Πολλών Μερών (Secure Multi Party Computation ή (SMPC)) και στη συνέχεια θα περιγραφούν και θα αναλυθούν ορισμένα βασικά πρωτόκολλα που χρησιμοποιούνται μέχρι και σήμερα. Σχετικά με την ανάλυση των πρωτοκόλλων θα ξεκινήσουμε από τα πιο απλά και ιστορικά που είναι ασφαλή ενάντια σε παθητικούς αντιπάλους και θα συνεχίσουμε προς τα πιο σύνθετα και σύγχρονα που είναι ασφαλή ενάντια σε ενεργητικούς αντιπάλους. Δυστυχώς, παρότι σε κάποια πρωτόκολλα ίσως είναι προφανής η ιδιότητα της ασφάλεια τους για συγκεκριμένα μοντέλα αντιπάλων, η πλήρης τυποποιημένη απόδειξη της ασφάλειας αυτής για όλα τα πρωτόκολλα που θα αναλύσουμε θα χρειαζόταν τουλάχιστον κάποιες δεκάδες σελίδες σελίδες παραπάνω και θα ξέφευγε από τα πλαίσια της εργασίας αυτής. Ωστόσο, η μεθοδολογία που περιγράψαμε στο Κεφάλαιο 3 καθώς και η παρατιθέμενη βιβλιογραφία δίνει ένα καλό εναρκτήριο σημείο για τον αναγνώστη που θέλει να εντρυφήσει στις αποδείξεις αυτές. Τα πρωτόκολλα που θα περιγράψουμε στο κεφάλαιο αυτό είναι τα Μπερδεμένα Δίκτυα Yao, BMR, GMW και SPDZ. Τα πρώτα τρία πρωτόκολλα υποστηρίζονται από τη βιβλιοθήκη ABY που χρησιμοποιούμε στο υλοποιητικό κομμάτι αυτής της εργασίας που βρίσκεται στο Κεφάλαιο 5. Είναι και τα τρία ασφαλή ενάντια σε παθητικούς αντιπάλους. Το τελευταίο πρωτόκολλο είναι ένα από τα πιο σύγχρονα και επιτυχημένα πρωτόκολλα που έχουν προταθεί στη βιβλιογραφία και παρότι δε σχετίζεται με την υλοποίηση επιλέξαμε να το αναλύσουμε για να υπάρχει σφαιρικότερη κάλυψη της βιβλιογραφίας, αφού είναι ασφαλές ενάντια σε ενεργητικούς αντιπάλους.

Στο Κεφάλαιο 1 κάναμε μια πολύ μικρή εισαγωγή στον κλάδο SMPC. Τα SMPC είναι διαδραστικά πρωτόκολλα στα οποία συμμετέχουν δύο ή περισσότεροι συμμετέχοντες, οι οποίοι διαθέτουν καμία, μία ή περισσότερες ιδιωτικές εισόδους. Η συνάρτηση που πρέπει να αποτιμηθεί για τις εισόδους αυτές, πρέπει να είναι από πριν γνωστή στους συμμετέχοντες ώστε σε πρωτόκολλα που το υποστηρίζουν να μπορούν να επαληθεύσουν αν αυτή που πρόκειται να αποτιμηθεί είναι η προσυμφωνηθήσα συνάρτηση. Σχετικά με την ασφάλεια του Υπολογισμού Πολλών Μερών είναι σημαντικό να αναφέρουμε ότι τα πρωτόκολλα SMPC μεριμνούν

αποκλειστικά και μόνο για την ασφαλή αποτίμηση του υπολογισμού. Αυτό σημαίνει ότι το αποτέλεσμα της αποτίμησης του υπολογισμού μπορεί να αποκαλύπτει σημαντικές πληροφορίες για την είσοδο κάποιου άλλου συμμετέχον, όπως για παράδειγμα στην περίπτωση μιας πύλης AND με δύο εισόδους όπου ελέγχονται από δύο συμμετέχοντες αντίστοιχα. Ας αναλογιστούμε το παράδειγμα της Ασφάλειας μέσω Προσομοίωσης, η οποία χρησιμοποιείται κατά κόρον στον κλάδο του SMPC. Στην περίπτωση αυτή, ένα πρωτόκολλο SMPC που έχει αποδειχθεί ασφαλές με αυτήν τη μέθοδο μας διασφαλίζει ότι οποιοσδήποτε αντίπαλος που ελέγχει κάποιους διεφθαρμένους συμμετέχοντες που συμμετέχουν στο πρωτόκολλο για την αποτίμηση του υπολογισμού δε μαθαίνει καμία πληροφορία παραπάνω από τη διαδικασία εκτέλεσης του πρωτοκόλλου, σε σχέση με την πληροφορία που θα μάθαινε αν συμμετείχε σε ένα πρωτόκολλο ιδανικού κόσμου. Προφανώς, ο αριθμός των διεφθαρμένων συμμετεχόντων που ελέγχονται από τον αντίπαλο, η υπολογιστική ισχύς του αντιπάλου καθώς και ο τρόπος διαφθοράς των συμμετεχόντων εξαρτάται από το μοντέλο αντιπάλου που έχουμε επιλέξει. Είναι αντιληπτό ότι τη γνώση που θα αποκτούσε ένας αντίπαλος στην περίπτωση μιας πύλης AND θα την αποκτούσε σε οποιοδήποτε, ιδανικό ή μη, πρωτόκολλο αποτίμησης της.

Στην ενότητα αυτή βασικές βιβλιογραφικές πηγές αποτέλεσαν τα βιβλία [32] και [48]. Επίσης, χρήσιμες εργασίες εισαγωγής και σύνοψης του κλάδου, αποτελούν οι [49] και [50]. Στην συνέχεια θα αναλύσουμε τα βασικά χαρακτηριστικά με βάση τα οποία μπορούμε να κατηγοριοποιήσουμε τα πρωτόκολλα SMPC.

4.1 Κατηγορίες πρωτοκόλλων ως προς το μοντέλο αναπαράσταση του υπολογισμού

Οποιοδήποτε πρωτόκολλο SMPC, για να εκτελέσει τον υπολογισμό μιας συνάρτησης f, χρειάζεται να λάβει την συνάρτηση σε μια συγκεκριμένη μορφή που εξαρτάται από το τύπο του πρωτοκόλλου. Δηλαδή, αν θέλουμε να χρησιμοποιήσουμε το πρωτόκολλο αυτό καλούμαστε να αναπαραστήσουμε την f στο κατάλληλο υπολογιστικό μοντέλο για το οποίο έχει φτιαχτεί το πρωτόκολλο. Αυτό συμβαίνει, με οποιαδήποτε μηχανή υπολογισμού, σε έναν Επεξεργαστή (CPU) ή σε μια Εικονική Μηχανή για παράδειγμα χρειάζεται να αναπαραστήσουμε την f στην κατάλληλη συμβολική γλώσσα ενώ σε μια Μηχανή Turing χρειάζεται να την αναπαραστήσουμε με την κατάλληλη αλγεβρική δομή. Στην περίπτωση των πρωτοκόλλων SMPC οι κύριες κατηγορίες που τα διακρίνουμε ως προς την αναπαράσταση της f είναι οι παρακάτω :

Definition 4.1.1. Κύρια Υπολογιστικά Μοντέλα των SMPC πρωτοκόλλων :

- Αναπαράσταση ως Boolean δίκτυο/κύκλωμα Συνδυαστικής Λογικής

- Αναπαράσταση ως Αριθμητικό δίκτυο/κύκλωμα
- Πιο σύνθετες αναπαραστάσεις, όπως ως Μηχανή Turing ή RAM (Random Access Machine)

Στα πλαίσια αυτής της εργασίας θα αναφερθούμε μόνο σε πρωτόκολλα που ανήκουν στις πρώτες δύο κατηγορίες. Τα πρωτόκολλα που ανήκουν στην τελευταία κατηγορία αποτελούν ιδιαίτερα ενεργό ερευνητικό κλάδο την τελευταία δεκαετία με αρκετά σημαντικές εργασίες όπως οι [51] και [52].

Προφανώς, οι παραπάνω κατηγορίες Boolean και Αριθμητικών δίκτυο σχετίζονται μεταξύ τους καθώς μπορούμε εύκολα να χρησιμοποιήσουμε ένα Αριθμητικό δίκτυο για τον υπολογισμό ενός Boolean κυκλώματος. Αν πάρουμε για παράδειγμα μια πύλη AND μπορούμε να τη μετατρέψουμε σε αριθμητική ως εξής :

$$\begin{aligned} NAND(x,y) &= AND(OR(x,y), OR(x, NOT(y)), OR(NOT(x), y)) = \\ &= OR(x,y) + OR(x, NOT(y)) + OR(NOT(x), y) = \\ &= x \cdot y + x \cdot NOT(y) + NOT(x) \cdot y = \\ &= x \cdot y + x \cdot (1-y) + (1-x) \cdot y \end{aligned}$$

Στην συγκεκριμένη μαθηματική έκφραση κάνουμε διαχωρισμό μεταξύ των λογικών πυλών τις οποίες σημειώνουμε με τα ονόματα τους και των συμβόλων τις πρόσθεσης, της αφαίρεσης και του πολλαπλασιασμού οι οποίες αναφέρονται στο μαθηματικό δακτύλιο ή πεδίο στο οποίο ανήκουν τα στοιχεία x, y.

Σε αυτό το σημείο πρέπει να αναφερθεί πως ορισμένα πρωτόκολλα SMPC υποστηρίζουν μόνο πεπερασμένο υπολογισμό, δηλαδή πεπερασμένους σε χρόνο αλγορίθμους, και άρα δεν μπορεί να περιέχει άπειρους βρόγχους. Δηλαδή, απαιτείται να γίνει Ξετύλιγμα των βρόγχων (Loop unrolling) στην περίπτωση που υπάρχουν. Αυτό συνήθως συμβαίνει σε πρωτόκολλα στα οποία το κύκλωμα πρέπει να αρχικοποιηθεί (bootstrap) ολόκληρο πριν ξεκινήσει ο υπολογισμός. Στα πρωτόκολλα που θα εξετάσουμε δε συμβαίνει αυτό στην περίπτωση της ασφάλειας ενάντια σε παθητικούς αντιπάλους. Ωστόσο, είμαστε αντιμέτωποι με το εμπόδιο αυτό αν επιθυμήσουμε να μετατρέψουμε τα πρωτόκολλα αυτά σε ασφαλή ενάντια σε ενεργητικούς αντιπάλους μέσω κάποιων ειδικών τεχνικών που θα εξετάσουμε στη συνέχεια του κεφαλαίου.

4.2 Κατηγορίες πρωτοκόλλων ως προς τις ιδιότητες ασφάλειας

Όπως προαναφέραμε, το μοντέλο αναπαράστασης του υπολογισμού που χρησιμοποιεί ένα πρωτόκολλο είναι σύμφυτο με αυτό. Ένα ακόμα σύμφυτο χαρακτηριστικό ενός πρωτο-

κόλλου είναι η ασφάλεια του. Γενικότερα στα πρωτόκολλα SMPC διακρίνουμε τις παρακάτω κατηγορίες ασφάλειας πρωτοκόλλων :

- Πρωτόκολλα ασφαλή ενάντια σε Παθητικούς Αντιπάλους
  - Ασφαλή ενάντια σε διεφθαρμένη μειοψηφία
  - Ασφαλή ενάντια σε διεφθαρμένη πλειοψηφία
- Πρωτόκολλα ασφαλή ενάντια σε Ενεργητικούς Αντιπάλους
  - Ασφαλή ενάντια σε διεφθαρμένη μειοψηφία
  - Ασφαλή ενάντια σε διεφθαρμένη πλειοψηφία

Προφανώς ένα πρωτόκολλο που είναι ασφαλές ενάντια σε ενεργητικούς αντιπάλους είναι ασφαλές και ενάντια σε παθητικούς. Το ίδιο ισχύει και στην περίπτωση της μειοψηφίας και πλειοψηφίας των διεφθαρμένων συμμετεχόντων. Στο Σχήμα 4.1 παραθέτουμε των πίνακα υποστηριζόμενων πρωτοκόλλων και των μοντέλων ασφάλειας στα οποία είναι ασφαλή, της βιβλιοθήκης MP-SPDZ [53]. Η βιβλιοθήκη αυτή έχει ίσως την εκτενέστερη υποστήριξη πρωτοκόλλων στην βιβλιογραφία.

		Αριθμητικό δίκτυο		Boolean κύκλωμα	
	Υπολογιστικό Μοντέλο / Μοντέλο Ασφάλειας	$modp$ ή $GF(2^n)$	$mod2^n$	Δυαδική Διαμοίραση Μυστικών	Μπερδεμένα Δίκτυα
Ενεργητικός	Διεφθαρμένη Πλειοψηφία	MASCOT / LowGear / HighGear	SPDZ2k	Tiny / Tinier	BMR
	Έμπιστή Πλειοψηφία	Shamir / Rep3 / PS / SY	Brain / Rep3 / PS / SY	Rep3 / CCD / PS	BMR
	Έμπιστη Υπερ-πλειοψηφία	Rep4	Rep4	Rep4	N/A
Συγκαλλημένος	Διεφθαρμένη Πλειοψηφία	CowGear / ChaiGear	N/A	N/A	N/A
Παθητικός	Διεφθαρμένη Πλειοψηφία	Semi / Hemi / Temi / Soho	Semi2k	SemiBin	Yao's GC / BMR
	Έμπιστη Πλειοψηφία	Shamir / ATLAS / Rep3	Rep3	Rep3 / CCD	BMR
	Dealer	Dealer	Dealer	Dealer	N/A

Πίνακας 4.1: Πίνακας υποστηριζόμενων πρωτοκόλλων της βιβλιοθήκης MP-SPDZ [53]. Ο παρόν είναι εμπνευσμένος από έναν παρόμοιο, αλλά δυσανάγνωστο, πίνακα που υπάρχει στα εγχειρίδια χρήσης της.

4.3 Ασφαλής Υπολογισμός Πολλών Μερών ενάντια σε Παθητικούς Αντιπάλους

Στη ενότητα αυτή θα εξετάσουμε τρία πρωτόκολλα ιστορικής σημασίας τα οποία χρησιμοποιούνται μέχρι και σήμερα λόγω της ευκολίας υλοποίησης τους και της απόδοσης τους. Τα πρωτόκολλα αυτά είναι τα Μπερδεμένα Δίκτυα Yao, το BMR και το GMW.

4.3.1 Πρωτόκολλο Μπερδεμένων Δικτύων Yao (Yao's Garbled Circuits Protocol)

Το πρωτόκολλο αυτό όπως αναφέραμε και στην Ενότητα 1 προτάθηκε από τον Yao το 1986 στις εργασίες [21] [22]. Πιο συγκεκριμένα, προτάθηκε ως πρωτόκολλο επίλυσης του Προβλήματος των Εκατομμυριούχων, δηλαδή για την περίπτωση των δύο συμμετεχόντων (2PC) αλλά μπορεί να γενικευθεί και σε περισσότερους συμμετέχοντες (MPC). Το συγκεκριμένο πρωτόκολλο απαιτεί την μετατροπή του αλγορίθμου σε μια Boolean συνάρτηση ή αντίστοιχα σε ένα Boolean κύκλωμα Συνδυαστικής Λογικής. Παρακάτω, θα αναλύσουμε την περίπτωση των δύο συμμετεχόντων για μια αυθαίρετη Boolean συνάρτηση και μετά θα αναφερθούμε στο πως μπορεί να επιταχυνθεί ο υπολογισμός με τη χρήση Boolean δικτύου και πως μπορεί να γενικευθεί το πρωτόκολλο για περισσότερους από έναν συμμετέχοντες. Το πρωτόκολλο αυτό είναι ασφαλές απέναντι σε Παθητικούς Αντιπάλους.

4.3.1.1 Υπολογισμός με χρήση του Πίνακα Αναζήτησης της συνάρτησης f

Έστω δύο συμμετέχοντες P_1 και P_2 που θέλουν να εκτελέσουν μια Boolean συνάρτηση $(\mathbf{out}) = f(\mathbf{x}, \mathbf{y})$ με x και y τα διανύσματα των εισόδων των P_1 και P_2 αντίστοιχα. Έστω επίσης x_i^j, το i-οστο bit του διανύσματος x για την τιμή j του bit αυτού. Γνωρίζουμε πως μια Boolean συνάρτηση μπορεί να περιγραφεί πλήρως από τον Πίνακα Αναζήτησης της (Look-up table). Το πρωτόκολλο αυτό απαιτεί κάποιον συμμετέχον να κάνει την προετοιμασία (bootstraping) ο οποίος πρέπει να γνωρίζει τον πλήρη Πίνακα Αναζήτησης της συνάρτησης f, ας υποθέσουμε ότι ο P_1 εκτελεί αυτή τη διαδικασία. Για κάθε γραμμή του πίνακα ο P_1 κρυπτογραφεί την αντίστοιχη έξοδο της συνάρτησης με $2 \cdot (|\mathbf{x}| + |\mathbf{y}|)$ διαφορετικά και ομοιόμορφα τυχαία επιλεγμένα κλειδιά χρησιμοποιώντας ένα συμμετρικό κρυπτογραφικό σχήμα με τον εξής τρόπο. Σε κάθε bit εισόδου του πίνακα αντιστοιχεί και ένα ζεύγος κλειδιών, ένα για κάθε τιμή που μπορεί να πάρει, αυτά τα συμβολίζουμε ως $k_{x_i^j}$. Δηλαδή αν $|x| = 2$ και $|y| = 3$ τότε η έξοδος **out** για την ανάθεση τιμών $x = 01$ και $y = 101$ θα κρυπτογραφηθεί ως $Enc_{k_{x_1^0}, k_{x_0^1}, k_{y_2^1}, k_{y_1^0}, k_{y_0^1}}(\mathbf{out})$. Ένα παράδειγμα της μορφής του πίνακα αυτού βρίσκεται στο Σχήμα ??. Ο P_1 αφού έχει ολοκληρώσει τη δημιουργία του κρυπτογραφημένου πίνακα αληθείας, εφαρμόζει ομοιόμορφα τυχαίες μεταθέσεις στις γραμμές του και τον αποστέλλει ολόκληρο στον P_2. Οι τυχαίες μεταθέσεις εφαρμόζονται στον πίνακα διότι αν ήταν γνωστή η διάταξη των γραμμών του από τον P_2 τότε αυτός αφού θα λάμβανε τα κατάλληλα κλειδία που αντιστοιχούν στην ιδιωτική του είσοδο θα μπορούσε βλέποντας την γραμμή που αντιστοιχεί στα κλειδιά που έχει λάβει να καταλάβει ποια είναι η είσοδος του P_1. Ωστόσο η μετάθεση των γραμμών δημιουργεί ένα άλλο πρόβλημα. Ο P_2 θα πρέπει να προσπαθήσει να αποκρυπτογραφήσει κάθε γραμμή του πίνακα προκειμένου να βρει την σωστή στη χειρότερη περίπτωση ενώ κατά μέσο όρο θα πρέπει να προσπαθήσει να αποκρυπτογραφήσει τις μισές

$$\begin{pmatrix} f(\mathbf{x}=00,\mathbf{y}=000) \\ f(\mathbf{x}=00,\mathbf{y}=001) \\ f(\mathbf{x}=00,\mathbf{y}=010) \\ f(\mathbf{x}=00,\mathbf{y}=011) \\ \vdots \\ f(\mathbf{x}=11,\mathbf{y}=110) \\ f(\mathbf{x}=11,\mathbf{y}=111) \end{pmatrix} \rightarrow \begin{pmatrix} Enc_{k_{x_1^0},k_{x_0^0},k_{y_2^0},k_{y_1^0},k_{y_0^0}}(f(\mathbf{x}=00,\mathbf{y}=000)) \\ Enc_{k_{x_1^0},k_{x_0^0},k_{y_2^0},k_{y_1^0},k_{y_0^1}}(f(\mathbf{x}=00,\mathbf{y}=001)) \\ Enc_{k_{x_1^0},k_{x_0^0},k_{y_2^0},k_{y_1^1},k_{y_0^0}}(f(\mathbf{x}=00,\mathbf{y}=010)) \\ Enc_{k_{x_1^0},k_{x_0^0},k_{y_2^0},k_{y_1^1},k_{y_0^1}}(f(\mathbf{x}=00,\mathbf{y}=011)) \\ \vdots \\ Enc_{k_{x_1^1},k_{x_0^1},k_{y_2^1},k_{y_1^1},k_{y_0^0}}(f(\mathbf{x}=11,\mathbf{y}=110)) \\ Enc_{k_{x_1^1},k_{x_0^1},k_{y_2^1},k_{y_1^1},k_{y_0^1}}(f(\mathbf{x}=11,\mathbf{y}=111)) \end{pmatrix}$$

Σχήμα 4.1: Παράδειγμα της διαδικασίας κρυπτογράφησης του πίνακα αληθείας της συνάρτησης f σε έναν νέο πίνακα αληθείας με το Πρωτόκολλο Μπερδεμένων Δικτύων Yao. Ο πίνακας αυτός διαφέρει από τον πραγματικό αφού στον πραγματικό οι γραμμές υπόκεινται σε μετάθεση σύμφωνα με τη τεχνική Point-and-Permute.

γραμμές του πίνακα. Μια γνωστή αποδοτική τεχνική επίλυσης του προβλήματος αυτού ονομάζεται Point-and-Permute , στην οποία θεωρούμε το τελευταίο bit του κάθε κλειδιού ως ένα από τα bit του δείκτη που δείχνει που θα τοποθετηθεί η γραμμή αυτή κατά την μετάθεση. Δηλαδή αν στο παραπάνω παράδειγμα τα τελευταία bit των κλειδιών έχουν τις εξής τιμές :

Add reference

$$\text{Τελευταίο bit κλειδιού} : k_{x_1^0}=1, k_{x_0^1}=0, k_{y_2^1}=1, k_{y_1^0}=1, k_{y_0^1}=0$$

Τότε η γραμμή αυτή θα αποθηκευτεί στην θέση $10110 = 22$ του πίνακα. Εφόσον έχουμε υποθέσει ότι τα κλειδιά είναι τυχαία επιλεγμένα η μέθοδος αυτή δεν επιρεάζει την ασφάλεια του πρωτοκόλλου. Ο P_1 αποστέλλει επίσης μαζί τα κατάλληλα κλειδιά κρυπτογράφησης $k_{x_i^j}$ για τις τιμές του διανύσματος εισόδου $\mathbf{x}$ που έχει επιλέξει. Δηλαδή στο προηγούμενο παράδειγμα θα στείλει τα κλειδιά $k_{x_1^0}$, $k_{x_0^1}$. Στην συνέχεια αποστέλλει στον P_2 τα κλειδιά που αντιστοιχούν στις τιμές του διανύσματος εισόδου $\mathbf{y}$, μέσω $|y| - (2 \cdot |y|)$ Ανυποψίαστης Μεταφοράς. Τέλος ο P_2 αφού έχει αποκτήσει όλα τα απαραίτητα κλειδιά για την αποκρυπτογράφηση της γραμμής του Πίνακα Αναζήτησης της f που αντιστοιχεί στο $f(\mathbf{x},\mathbf{y})$, την αποκρυπτογραφεί, βλέπει το αποτέλεσμα **out** και στην συνέχεια το στέλνει στον P_1.

4.3.1.2 Υπολογισμός με χρήση του Boolean Δικτύου της f

Είναι προφανές ότι η παραπάνω μέθοδος δεν μπορεί να κλιμακωθεί σωστά αφού το μέγεθος του Πίνακα Αναζήτησης της f αυξάνεται εκθετικά με το μέγεθος των εισόδων. Μια ιδέα για την επιτάχυνση του υπολογισμού της f από ένα Μπερδεμένο Δίκτυο Yao είναι να

την εκφράσουμε ως ένα Boolean Δίκτυο, από την προσέγγιση αυτή προκύπτει ουσιαστικά και το όνομα αυτού του πρωτοκόλλου. Παρακάτω θα επανεξετάσουμε την αποτίμηση μιας συνάρτησης f από τους δύο συμμετέχοντες P_1, P_2.

Ο P_1 όπως και προηγουμένως είναι αυτός που αναλαμβάνει να αρχικοποιήσει το πρωτόκολλο. Στην προκείμενη περίπτωση να κατασκευάσει το Μπερδεμένο Κύκλωμα. Για να το επιτύχει αυτό, δημιουργεί ένα ζεύγος κλειδιών για κάθε καλώδιο του κυκλώματος καθώς και ένα τυχαία συμπληρωματικά bit δεικτοδότησης (παραπλήσια με την τεχνική point-and-permute) για κάθε ζεύγος. Έτσι σε κάθε ζεύγος κλειδιών ενός καλωδίου i αντιστοιχεί η εξής πλειάδα, την οποία ονομάζουμε **Ετικέτα του καλωδίου** :

$$w_i^b = (k_i^b \in \{0,1\}^{1n}, p_i^b \in \{0,1\}) \text{ όπου } p_i^b = 1 - p_i^{1-b}$$

Στη συνέχεια ο P_1 διατρέχει με το κύκλωμα σύμφωνα με την τοπολογική του διάταξη. Για μια αυθαίρετη πύλη g_l του κυκλώματος, όπως αυτή του Σχήματος **??** ανάλογα με το αν είναι πύλη εισόδου, ενδιάμεση ή εξόδου εκτελεί ένα από τα παρακάτω βήματα. Η διαφοροποίηση της παραλλαγής αυτής του πρωτοκόλλου σε σχέση με την προηγούμενη παραλλαγή που εξετάσαμε είναι πως για κάθε πύλη του κυκλώματος θα δημιουργήσουμε έναν πίνακα αληθείας. Για να διαχωρίσουμε τους δύο πίνακες θα ονομάσουμε τον πίνακα της προηγούμενης παραλλαγής ολικό πίνακα αληθείας και τον πίνακα αυτής της παραλλαγής τοπικό πίνακα αληθείας. Οι δύο πίνακες αυτοί έχουν μια βασική διαφορά στην κατασκευή τους. Στον τελευταίο η έξοδος κάθε γραμμής είναι η πραγματική έξοδος του κυκλώματος, ενώ στον πρώτο η έξοδος κάθε ενδιάμεσης πύλης είναι ένα μια ετικέτα που περιέχει και το κλειδί κρυπτογράφησης που χρησιμοποιείται στην κρυπτογράφηση των γραμμών των πινάκων που λαμβάνουν ως είσοδο την έξοδο αυτή. Στα σημείο αυτό, για χάριν απλότητας, υποθέτουμε ότι όλες οι πύλες του κυκλώματος διαθέτουν δύο bit εισόδου και ένα bit εξόδου. Προφανώς, ο αλγόριθμος μπορεί πολύ εύκολα να επεκταθεί και για αυθαίρετο αριθμό bit εισόδου και εξόδου. Επίσης, κατά την αρχικοποίηση του κυκλώματος ο P_1 μπορεί πολύ εύκολα μια πύλη αυτού του είδους με ένα ισοδύναμο συνδυασμό πυλών που αποτελείται μόνο από πύλες με δύο bit εισόδου και ένα bit εξόδου. για την οποία ισχύει $w_k = g_l(w_i, w_j)$

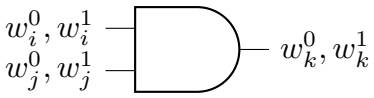
- **Οποιαδήποτε πύλη** : Στην περίπτωση αυτή, ο P_1, για κάθε μία από τις 2^2 πιθανές εισόδους v_i, v_j ∈0,1 πρέπει να δημιουργήσει μια γραμμή στον τοπικό πίνακα. Αν θεωρήσουμε e_{v_i,v_j} τη τιμή της γραμμής $v_i v_j$ τότε αυτή δημιουργείται ως εξής και τοποθετείται στη θέση $p_i^{v_i} p_j^{v_j}$:

  $$e_{v_i,v_j} = Enc_{k_i^{v_i}, k_j^{v_j}, i}(w_k^{g_l(v_a,v_b)})$$

  Ένα παράδειγμα μετασχηματισμού του τοπικού πίνακα μιας πύλης φαίνεται στο Σχήμα ??.

$$\begin{pmatrix} G_l(x=0,y=1) \\ G_l(x=0,y=0) \\ G_l(x=1,y=0) \\ G_l(x=1,y=1) \end{pmatrix} \rightarrow \begin{pmatrix} Enc_{k_i^0,k_j^0,i}(w_k^{g_l(x=0,y=0)}) \\ Enc_{k_i^0,k_j^1,i}(w_k^{g_l(x=0,y=1)}) \\ Enc_{k_i^1,k_j^0,i}(w_k^{g_l(x=1,y=0)}) \\ Enc_{k_i^1,k_j^1,i}(w_k^{g_l(x=1,y=1)}) \end{pmatrix}$$

Σχήμα 4.2: Παράδειγμα της διαδικασίας κρυπτογράφησης του πίνακα αληθείας της συνάρτησης f σε έναν νέο πίνακα αληθείας με το Πρωτόκολλο Μπερδεμένων Δικτύων Yao. Ο πίνακας αυτός διαφέρει από τον πραγματικό αφού στον πραγματικό οι γραμμές υπόκεινται σε μετάθεση σύμφωνα με τη τεχνική των bit δεικτοδότησης που αναλύσαμε.



Σχήμα 4.3: Παράδειγμα πύλης Μπερδεμένου Δικτύου Yao στην οποία φαίνονται οι δύο ετικέτες του κάθε καλωδίου μιας πύλης.

- **Καλώδια εξόδου** : Στην περίπτωση αυτή ακολουθείται ίδια διαδικασία με οποιοδήποτε άλλο καλώδιο μόνο που η τιμή που θα κρυπτογραφήσουμε θα πρέπει να είναι η έξοδος του καλωδίου και όχι κάποιο κρυπτογραφικό κλειδί που χρησιμοποιείται στην αποκρυπτογράφηση των εξόδων των πυλών που δέχονται το καλώδιο εξόδου ως είσοδο. Τα καλώδια εξόδου δημιουργούνται ως εξής :

$$e_v = Enc_{k_i^v}()$$

4.3.2 Πρωτόκολλο Ben-Or, Goldwasser, Wigderson (BGW Protocol)

Το πρωτόκολλο αυτό προτάθηκε από τους Ben-Or, Goldwasser, Wigderson στην εργασία [54]. Απαιτεί την μοντελοποίηση της f ως ένα Αριθμητικό Δίκτυο σε ένα πεπερασμένο πεδίο **F**. Στην αρχική του μορφή βασίζεται στη Διαμοίραση Μυστικών Shamir και πιο συγκεκριμένα στις ομομορφικές ιδιότητες που παρουσιάζουν οι μετοχές του (για περισσότερες πληροφορίες σχετικά με τις ομομορφικές ιδιότητες μπορείτε να ανατρέξετε στο Παράρτημα), ωστόσο μπορούν να χρησιμοποιηθούν και άλλες μέθοδοι διαμοίρασης μυστικών υπό την προϋπόθεση ότι παρουσιάζουν τις ίδιες ομομορφικές ιδιότητες. Όπως αναλύθηκε στο παράρτημα, το SSS είναι προσθετικά ομομορφικό από μόνο του, οπότε κύρια καινοτομία του πρωτοκόλλου BGW

Add reference

είναι η μετατροπή του SSS ώστε να μπορεί είναι και πολλαπλασιαστικά ομομορφικό. Παρακάτω θα εξετάσουμε τις περιπτώσεις της ομομορφικής πρόσθεσης και πολλαπλασιαμού στην περίπτωση της μιας αριθμητικής πύλης, στην εξετάσουμε πως μπορούμε να γενικέυσουμε το πρωτόκολλο στην περίπτωση των περισσότερον πυλών και τέλος θα παρουσιάσουμε αναλυτικά το πρωτόκολλο.

Έστω ότι n συμμετέχοντες όπου θέλουν να αποτιμήσουν μια αριθμητική συνάρτηση ή αντίστοιχα ένα αριθμητικό δίκτυο $(\mathbf{out}) = f(\mathbf{x}, \mathbf{y})$, όπου x, y είναι οι είσοδοι της συνάρτησης και η καθεμία ελέγχεται από κάποιον συμμετέχον. Η αριθμητική συνάρτηση f πρέπει να είναι γνωστή σε όλους τους συμμετέχοντες. Επίσης, ο κάθε συμμετέχον διαθέτει ένα σημείο $i \in \mathbf{F}$ το οποίο το γνωρίζουν όλοι οι συμμετέχοντες στο δίκτυο, για χάριν απλότητας ας υποθέσουμε ότι το i είναι ο αύξων αριθμός του συμμετέχοντος και $i = 0, \dots, n-1 \in \mathbf{F}$. Στην επόμενη παράγραφο ο συμβολισμός i θα χρησιμοποιείται για να αναφερθούμε είτε στο σημείο του παίχτη i είτε στον ίδιο τον παίχτη i.

Στην αρχή του πρωτοκόλλου οι δύο συμμετέχοντες που ελέγχουν αυτές τις δύο εισόδους x και y διαμοιράζουν μέσω $SSS(n, t)$, υπό τον περιορισμό ότι $t < 2n$ (θα αιτιολογηθεί στην συνέχεια ο περιορισμός αυτός), αντίστοιχες μετοχές $[x]_i$ και $[y_i]$ σε κάθε συμμετέχον $i \in 0, 1, \dots n$. Όπως γνωρίζουμε από το SSS, η μετοχή του παίχτη i είναι ένα σημείο $(i, p_x(i))$, για την είσοδο x και το p_x είναι το πολυώνυμο t-οστού βαθμού που επέλεξε ο συμμετέχον που ελέγχει την είσοδο x κατά το SSS.

Replace 2t with 2t+1

- **Πρόσθεση** : Κάθε συμμετέχον i διαθέτει μια μετοχή $[x]_i$ και μια $[y]_i$ αντίστοιχα. Κάθε συμμετέχον μπορεί να εκτελέσει την πράξη $x + y$, απλώς αθροίζοντας τις τετμημένες των αντίστοιχων μετοχών, δηλαδή $[c]_i = [x]_i + [y]_i$. Το παραπάνω προκύπτει από την ιδιότητα της γραμμικότητας του SSS που αναλύθηκε στην Ενότητα , αφού προκύπτει ότι $[c]_i = [x]_i + [y]_i = p_x(i) + p_y(i) = p_{x+y}(i) = [x + y]_i$. Η πράξη της πρόσθεσης δεν απαιτεί καμία αλληλεπίδραση μεταξύ των συμμετεχόντων εφόσον ο καθένας τους διαθέτει τις μετοχές $[x]_i$ και $[y]_i$.

Add reference

- **Πολλαπλασιασμός** :
  - **Βαθμωτός πολλαπλασιασμός** : Παρόμοια με την πράξη της πρόσθεσης από την γραμμική ιδιότητα του SSS προκύπτει ότι $[c]_i = b \cdot [x]_i = b \cdot p_x(i) = p_{b \cdot x}(i) = [b \cdot x_i]$.
  - **Πολλαπλασιασμός με μεταβλητή εισόδου** : Κάθε συμμετέχον i διαθέτει μια μετοχή $[x]_i$ και μια $[y]_i$ αντίστοιχα. Όπως και με τους υπόλοιπους τελεστές αρχικά ο κάθε συμμετέχον υπολογίζει την μετοχή $[c]_i = [x]_i \cdot [y]_i$. Γνωρίζουμε όμως ότι $[c]_i = [x]_i \cdot [y]_i = p_x(i) \cdot p_y(i) = p_{x \cdot y}(i) (= p_c(i)) = [x \cdot y]_i$. Όπως παρατηρούμε

το πρόβλημα που προκύπτει στην πράξη του πολλαπλασιασμό είναι ότι ο πολλαπλασιασμός δύο πολυωνύμων t-οστού βαθμού, όπως στην προκείμενη περίπτωση των p_x και p_y, μπορεί να μας δώσει ως αποτέλεσμα ένα πολυώνυμο μέχρι και $2t$-οστού βαθμού. Είναι απαραίτητο να κρατήσουμε σταθερό αυτό το κατώφλι t. Για να το επιτύχουμε αυτό αρχικά ο κάθε συμμετέχον εκτελεί τον πολλαπλασιασμό των μετοχών $[c]_i = [x]_i \cdot [y]_i$, όπως και με τους υπόλοιπους τελεστές. Στο σημείο αυτό το $[c]_i$ έχει κατώφλι διαμοίρασης $2t$. Για να μειωθεί το κατώφλι σε t ο κάθε συμμετέχον i διαμοιράζει το $[c]_i$ με $SSS(n,t)$, ας υποθέσουμε ότι χρησιμοποιεί για αυτό ένα τυχαίο πολυώνυμο $p_{h_i}(x)$ όπου προφανώς $p_{h_i}(0) = [c]_i = p_c(i)$, και στην συνέχεια διαμοιράζει τις μετοχές που αντιστοιχούν στον καθένα από τους υπόλοιπους $n-1$ συμμετέχοντες. Δηλαδή, στον παίχτη j ο παίχτης i θα στείλει την τιμή $p_{h_i}(j)$. Έτσι, ο κάθε συμμετέχον j διαθέτει τις μετοχές $p_{h_i}(j)$ για κάθε $i \in 1, 2, \dots n$. Στην συνέχεια κάθε συμμετέχον υπολογίζει την τελική του μετοχή ως εξής :

$$h(j) = \sum_{i=1}^{2t+1} \lambda_i(x) h_i(j) \tag{4.1}$$

όπου λ_i είναι ο κατάλληλος συντελεστής που αναφέρεται στο παράρτημα και μπορεί να υπολογιστεί από κάθε συμμετέχοντα αφού είναι δημόσια γνωστό στους συμμετέχοντες ότι $\alpha_i = i$. Επειδή, η ορθότητα της Σχέσης 4.1 ίσως να μην είναι προφανής ας σταθούμε λίγο περισσότερο σε αυτήν. Η Σχέση 4.1 μπορεί να αναπτυχθεί στην παρακάτω :

Add reference

$$p_h(j) = \sum_{i=1}^{2t+1} \lambda_i(x) p_{h_i}(j) = \lambda_1 p_{h_1}(j) + \lambda_2 p_{h_2}(j) + \dots + \lambda_n p_{h_n}(j)$$

και ουσιαστικά το πολυώνυμο $p_h(x)$ του οποίου σημείο είναι το $p_h(j)$ είναι το παρακάτω :

$$p_h(x) = \lambda_1 p_{h_1}(x) + \lambda_2 p_{h_2}(x) + \dots + \lambda_n p_{h_n}(x)$$

και ισχύει ότι $deg(p_h(x)) = t$ αφού είναι άθροισμα των πολυωνύμων p_{h_i} για καθένα από τα οποία ισχύει $deg(p_{h_i} = t)$. Επίσης, μπορούμε να επαληθεύσουμε ότι $h(0) = p_c(0)$ αφού

$$p_h(0) = \lambda_1 p_{h_1}(0) + \lambda_2 p_{h_2}(0) + ... + \lambda_n p_{h_n}(0) \tag{4.2}$$
$$= \lambda_1 p_c(1) + \lambda_2 p_c(2) + ... + \lambda_n p_c(n) \tag{4.3}$$
$$= p_c(0) \tag{4.4}$$

Το παραπάνω πρωτόκολλο μπορεί να παραλαχθεί ώστε να είναι ασφαλές ενάντια σε Ενεργητικούς αντιπάλους, όπως για παράδειγμα στις εργασίες , ωστόσο δεν θα αναφερθούμε σε αυτά τα πρωτόκολλα στα πλαίσια αυτής της εργασίας, λόγω της πολυπλοκότητας τους.

add reference

4.3.2.1 Επιτάχυνση του πρωτοκόλλου με χρήση Πολλαπλασιαστικών Τριάδων Beaver (Beaver Triples)

Στο πρωτόκολλο BGW, στην περίπτωση του πολλαπλασιασμού που έχει προκύψει από είσοδο, κάθε παίχτης πρέπει να εκτελέσει $SSS(n,t)$ και δηλαδή να διαμοιράσει n μετοχές. Αυτό έχει πολυπλοκότητα επικοινωνίας $O(n^2)$. Μπορούμε να βελτιώσουμε την παραπάνω πολυπλοκότητα επικοινωνίας αν χωρίσουμε τον υπολογισμό σε δύο φάσεις, την **Φάση προ-επεξεργασίας (Preprocessing phase)** και την **Online φάση**. Έτσι μπορούμε να επιτύχουμε βελτίωση της πολυπλοκότητας επικοινωνίας με τον εξής τρόπο.

Ας υποθέσουμε ότι x και y είναι οι είσοδοι της πολλαπλασιαστικής πύλης για τις οποίες κάθε συμμετέχον διαθέτει μετοχές $[x]$ και $[y]$ αντίστοιχα. Επίσης έστω οι τιμές $a, b \leftarrow\$ \mathbb{F}_p$, $c = a \cdot b$ και οι αντίστοιχες μετοχές $[a]$, $[b]$, $[c]$ τους. Τότε ο κάθε συμμετέχον μπορεί να υπολογίσει την πολλαπλασιαστική πύλη ακολουθώντας τα παρακάτω βήματα :

1. Υπολόγιζει τα $[x-a]$ και $[y-b]$ και ανακοινώνει δημόσια τις τιμές $d = x - a$ και $e = y - b$.

2. Υπολογίζει το $de + d[b] + e[a] + [c] = [x \cdot y]$ αφού,

$$xy = (x - a + a)(y - a + a) \tag{4.5}$$
$$= (d + a)(d + b) \tag{4.6}$$
$$= de + db + ae + ab \tag{4.7}$$
$$= de + db + ae + c \tag{4.8}$$

Τέλος, να σημειώσουμε ότι ο διαχωρισμός του πρωτοκόλλου σε περισσότερες από μία φάσεις είναι κάτι πολύ σύνηθες στην περίπτωση των SMPC και FHE πρωτοκόλλων, που αποσκοπεί συνήθως στην εξοικονόμηση χρονικής ή χωρικής πολυπλοκότητας κάθε φάσης μέσω των προηγούμενων της.

4.3.3 Πρωτόκολλο Goldreich, Micali, Wigderson (GMW Protocol)

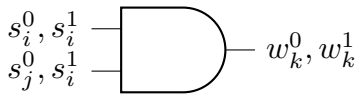
Το πρωτόκολλο αυτό προτάθηκε από τους Goldwasser, Micali, Wigderson στην εργασία [55] το 1987. Μπορεί να χρησιμοποιηθεί για την αποτίμηση συναρτήσεων μοντελοποιημένων ως Boolean δίκτυα αλλά και ως αριθμητικά δίκτυα ωστόσο θα επικεντρωθούμε στις Boolean συναρτήσεις. Το πρωτόκολλο αυτό μοιάζει σε ένα βαθμό, με αυτό του Yao, που περιγράψαμε προηγουμένως. Οι συμμετέχοντες έχουν ένα Boolean δίκτυο το οποίο αποτιμούν πύλη με πύλη (gate-by-gate) με τοπολογική σειρά. Ένα ακόμα κοινό χαρακτηριστικό είναι ότι και το GMW χρησιμοποιεί OT για τον υπολογισμό ορισμένων Boolean πυλών. Σε αυτό το χαρακτηριστικό βασίζεται και η κύρια διαφορά των δύο πρωτοκόλλων. Το GMW απαιτεί OT μόνο στην περίπτωση της AND πύλης, ενώ οι πύλες NOT και XOR μπορούν να υπολογιστούν χωρίς OT. Το γεγονός αυτό κάνει το GMW κατά μέσο όρο πιο γρήγορο από το πρωτόκολλο του Yao.

Στο πρωτόκολλο αυτό για κάθε καλώδιο του Boolean δικτύου δημιουργούνται δύο μετοχές και ο κάθε συμμετέχον κρατάει μια από αυτές.

- **Πύλη NOT** : Στην περίπτωση της Πύλης NOT, ένας από τους δύο συμμετέχοντες (όχι και οι δύο), το μόνο που έχει να κάνει είναι να αντιστρέψει το bit της μετοχής του.
- **Πύλη XOR** : Στην περίπτωση της Πύλης XOR, οι δύο συμμετέχοντες κάνουν XOR τις μετοχές που έχουν για κάθε μία από τις δύο εισόδους της πύλης.

Θα εξετάσουμε την περίπτωση δύο συμμετεχόντων, P_1 και P_2, και μιας Boolean πύλης με δύο bit εισόδου και ένα bit εξόδου :

- **Πύλη εισόδου** : Έστω πως ο συμμετέχον P_1 διαθέτει ένα ιδιωτικό bit εισόδου x_i το οποίο αντιστοιχεί σε ένα καλώδιο w_i του κυκλώματος. Έστω επίσης s_i^j η μετοχή για το καλώδιο w_i του παίχτη j. Για κάθε καλώδιο i επιθυμούμε $s_i^j \oplus s_i^{j-1} = w_i$. Ο P_1 δημιουργεί τις μετοχές ως εξής :
  - Στέλνει στον P_2 το $r_i \leftarrow\$ \{0,1\}$, η οποία είναι η μετοχή του P_2 για το καλώδιο w_i, $s_i^2 = r_i$
  - Δημιουργεί τη δική του μετοχή του w_i ως εξής : $s_i^1 = x_i \oplus r_i$
- **Πύλη NOT** : Έστω το καλώδιο εισόδου w_k και το καλώδιο εξόδου w_{k+1}. Στην περίπτωση της Πύλης NOT, ένας από τους δύο συμμετέχοντες (όχι και οι δύο), έστω ο P_1, το μόνο που έχει να κάνει είναι να αντιστρέψει το bit της μετοχής του, δηλαδή $s_{k+1}^1 = 1 - s_k^1$
- **Πύλη XOR** : Έστω τα καλώδια εισόδου w_k, w_{k+1} και το καλώδιο εξόδου w_{k+2}. Στην περίπτωση της Πύλης XOR, οι δύο συμμετέχοντες κάνουν XOR τις μετοχές που έχουν



Σχήμα 4.4

για κάθε μία από τις δύο εισόδους της πύλης. Δηλαδή $s_{k+2}^1 = s_k^1 \oplus s_{k+1}^1$ και $s_{k+2}^2 = s_k^2 \oplus s_{k+1}^2$

- **Πύλη AND**: Έστω τα καλώδια εισόδου w_k, w_{k+1} και το καλώδιο εξόδου w_{k+2}. Στην περίπτωση της Πύλης AND, ο ένας από τους δύο συμμετέχοντες, έστω ο P_1 πρέπει να ετοιμάσει ένα 1-4 OT της παρακάτω συνάρτησης :

$$S = S_{s_i^1, s_j^1}\left(s_i^2, s_j^2\right) = \left(s_i^1 \oplus s_i^2\right) \wedge \left(s_j^1 \oplus s_j^2\right)$$

Αυτό το κάνει επιλέγοντας $r \leftarrow\$ \{0,1\}$ και στη συνέχεια δημιουργόντας τον παρακάτω πίνακα από τον οποία στέλνει την κατάλληλη γραμμή μέσω 1/4 OT ανάλογα με το τι τιμή έχουν οι μετοχές του P_2 :

$$T_G = \begin{pmatrix} r \oplus S(0,0) \\ r \oplus S(0,1) \\ r \oplus S(1,0) \\ r \oplus S(1,1) \end{pmatrix}$$

4.4 Ασφαλής Υπολογισμός Πολλών Μερών ενάντια σε Ενεργητικούς Αντιπάλους

4.4.1 Τεχνικές Μετατροπής μετατροπής της ασφάλειας πρωτοκόλλων από Παθητική σε Ενεργητική

4.4.1.1 Τεχνικές Cut-and-Choose

4.4.2 Πρωτόκολλα Ασφαλή ενάντια σε Ενεργητικούς Αντιπάλους

4.4.3 SPDZ

Κεφάλαιο 5

Υλοποίηση

Στο κεφάλαιο αυτό θα περιγράψουμε την πειραματική υλοποίηση που πραγματοποιήθηκε στα πλαίσια αυτής της διπλωματικής εργασίας. Πρόκειται για την υλοποίηση Βασικών Υπορουτινών Γραμμικής Άλγεβρας (Basic Linear Algebra Subroutines ή BLAS) Επιπέδου 1 για Ασφαλή Υπολογισμό Δύο Μερών η οποία βασίζεται στη βιβλιοθήκη ABY για την εκτέλεση των πρωταρχικών κρυπτογραφικών και ασφαλούς υπολογισμού διεργασιών. Την βιβλιοθήκη που υλοποιήσαμε ονομάζουμε MPC-BLAS, ωστόσο ένα ίσως καταλληλότερο όνομα θα ήταν το 2PCBLAS. Πρωτού περάσουμε στην ανάλυση της MPC-BLAS βιβλιοθήκης, πρέπει πρώτα να αναλύσουμε ένα λιγότερο κρυπτογραφικό κομμάτι αυτής της εργασίας, τις απλές BLAS βιβλιοθήκες.

5.1 Η διεπαφή BLAS

Η Γραμμική Άλγεβρα αποτλεί πλέον θεμελιώδες μαθηματικό εργαλείο των θετικών επιστήμων, για αυτό και αποτελεί βασικό μάθημα των πρώτων ετών των προπτυχιακών σπουδών σε σχεδόν όλες τις σχολές θετικών επιστημών. Εξετάζοντας μόνο την επιστήμη της πληροφορικής, τη βλέπουμε σε πάρα κλάδους της, όπως τη Μηχανική Μάθηση (π.χ. SVM), στην Ψηφιακή Επεξεργασία Σημάτων (π.χ. FFT), στην Αριθμητική Ανάλυση, στην Κρυπτογραφία (π.χ. Κρυπτογραφία Πλεγμάτων), στα Γραφικά Υπολογιστών (π.χ. Ray Tracers) καθώς και σε πολλούς ακόμα. Η μαζικότητα αυτή, της χρήσης της γραμμικής άλγεβρας, δημιούργησε από τα πρώτα χρόνια των γλωσσών προγραμματισμού την επιτακτική ανάγκη για την εκτέλεση γρήγορων πράξεων γραμμικής άλγεβρας. Έτσι, από τα χρόνια της γλώσσας Fortran, εγκαθιδρύθηκε από το BLAS Technical Foroum ένα πρότυπο, μια διεπαφή συναρτήσεων, την οποία μπορούν να χρησιμοποιούν τα προγράμματα Fortran (Basic Linear Algebra Subroutines ή BLAS). Ο σκοπός της διεπαφής αυτής είναι η υλοποίηση να μπορεί να είναι διαφορετική από πλατφόρμα σε πλατφόρμα, ώστε να εκμεταλλεύεται τις ιδιομορφίες του υλικού της κάθε

πλατφόρμας. Κρατώντας τη διεπαφή σταθερή δε χρειάζεται να αλλάξει ένα πρόγραμμα που τη χρησιμοποιεί. Το μόνο που χρειάζεται να γίνει είναι να ξανά μεταγλωττιστεί το πρόγραμμα για τη συγκεκριμένη πλατφόρμα και να συνδεθεί (linked) με τη νέα BLAS υλοποίηση. Μάλιστα στην περίπτωση που η BLAS υλοποίηση υποστηρίζει δυναμική σύνδεση (dynamic linking) μπορεί να μην είναι απαραίτητο να ξαναμεταγλωττιστεί το πρόγραμμα για τη συγκεκριμένη πλατφόρμα. Με την έλευση της γλώσσας C, δημιουργήθηκε αντίστοιχη διεπαφή για τη C όπως και με αυτή της Fortran. Έχει επικρατήσει να καλούμε την διεπαφή για τη Fortran ως BLAS ενώ για τη C ως CBLAS.

Γενικότερα, η ύπαρξη τέτοιου είδους διεπαφής δίνει τη δυνατότητα σε κατασκευαστές υλικού, για παράδειγμα επεξεργαστών ή καρτών γραφικών, να δημιουργήσουν BLAS υλοποιήσεις που να εκμεταλλεύονται τα χαρακτηριστικά των κάθε υλικού τους. Έτσι σήμερα υπάρχουν στη βιβλιογραφία πάρα πολλές υλοποιήσεις της διεπαφής BLAS, ανοιχτού ή κλειστού κώδικα, από ποικίλους συγγραφείς, μεταξύ των οποίων και οι μεγαλύτεροι κατασκευαστές υλικού επεξεργαστών και καρτών γραφικών.

Η διεπαφή BLAS, χωρίζεται σε τρία επίπεδα. Το δεύτερο επίπεδο περιέχει πράξεις μόνο μεταξύ διανυσμάτων ή γενικότερα πράξεις επάνω σε διανύσματα, όπως για παράδειγμα η εύρεση της θέσεις του στοιχείου με τη μεγαλύτερη απόλυτη τιμή (πράξη IxAMAX). Το δεύτερο επίπεδο περιέχει μόνο πράξεις μεταξύ πινάκων και διανυσμάτων ενώ το τρίτο επίπεδο περιέχει μόνο πράξεις μεταξύ πινάκων. Στα πλαίσια της εργασίας αυτής θα αναφερθούμε μόνο στις πράξεις του πρώτου επιπέδου οι οποίες φαίνονται συνοπτικά και με αφαιρετικό τρόπο στο Σχήμα 5.1, ολόκληρη η διεπαφή BLAS βρίσκεται στον παρακάτω σύνδεσμο. Οι ονομασίες που χρησιμοποιούνται για τις συναρτήσεις και τις υπορουτίνες είναι στη μορφή xDOT. Το x είναι ένας χαρακτήρας μπαλαντέρ που παίρνει τιμές S, D, C, Z ανάλογα με το αν αναφερόμαστε σε πραγματικούς αριθμούς, μονής ή διπλή ακρίβειας, ή σε μιγαδικούς αριθμούς, μονής ή διπλής ακρίβειας, αντίστοιχα. Για παράδειγμα, η αντίστοιχη διεπαφή CBLAS, για τη συνάρτηση xDOT, φαίνεται στο Σχήμα 5.2.

Στο παράδειγμα του Σχήματος 5.1, ως συναρτήσεις (functions) θεωρούνται αυτές που επιστρέφουν την έξοδο τους ως την επιστρεφόμενη τιμή. Ωστόσο, δεν είναι εμφανές με ποιον τρόπο γίνεται η επιστροφή των εξόδων των υπορουτινών. Αυτή γίνεται με κλήση (call by reference) και επιστροφή μέσω αναφοράς (return by reference). Έτσι για παράδειγμα η υπορουτίνα xAXPY τοποθετεί την καινούργια τιμή του Y στην διεύθυνση που βρίσκεται η παλιά του τιμή. Οι παράμετροι $INCx$ που βρίσκονται σε όλες τις συναρτήσεις που παίρνουν ως όρισμα ένα διάνυσμα καθορίζουν το βήμα θα διαβαστούν οι τιμές του διανύσματος. Αν θέλουμε να χρησιμοποιήσουμε ολόκληρο το διάνυσμα X μπορούμε να χρησιμοποιήσουμε $INCX = 1$ ενώ αν θέλουμε να διαβάσουμε μόνο τις τιμές που βρίσκονται στις άρτιες θέσεις του μπορούμε να χρησιμοποιήσουμε την τιμή $INCX = 2$. Τέλος, η τιμή N είναι ο αριθμός των τιμών που θα διαβαστούν από το κάθε διάνυσμα.

$SUBROUTINE$	$xROTG$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	Generate plane rotation
$SUBROUTINE$	$xROTMG$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	Generate modified plane rotation
$SUBROUTINE$	$xROT$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	Apply plane rotation
$SUBROUTINE$	$xROTM$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	Apply modified plane rotation
$SUBROUTINE$	$xSWAP$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$x \leftrightarrow y$
$SUBROUTINE$	$xSCAL$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$x \leftarrow y$
$SUBROUTINE$	$xCOPY$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$y \leftarrow x$
$SUBROUTINE$	$xAXPY$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$y \leftarrow ax + y$
$FUNCTION$	$xDOT$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$dot \leftarrow x^T y$
$FUNCTION$	$xDOT$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$dot \leftarrow x^T y$
$FUNCTION$	$xDOTU$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$dot \leftarrow x^H y$
$FUNCTION$	$xDOTC$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$\leftarrow a + x^T y$
$FUNCTION$	$xxDOT$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$dot \leftarrow a + x^T y$
$FUNCTION$	$xNRM2$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$nrm2 \leftarrow \|x\|_2$
$FUNCTION$	$xASUM$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$asum \leftarrow \|re(x)\|_1 + \|im(x)\|_1$
$FUNCTION$	$IxAMAX$	(	N,	X,	$INCX$,	Y,	$INCY$,	$D1$,	$D2$,	A,	B,	C,	S,	$PARAM$	)	$amax \leftarrow 1^{st} k \|re(x_k)\| + \|im(x)\|_1$

Σχήμα 5.1: Πίνακας Βασικών Υπορουτινών Γραμμικής Άλγεβρας - Επιπέδου 1 (BLAS-Level 1)

5.2 Η βιβλιοθήκη ABY

Ο κώδικας της βιβλιοθήκης ABY βρίσκεται στο αποθετήριο Github σε αυτό τον σύνδεσμο και ο "Οδηγός για Προγραμματιστές" που παρέχει βρίσκεται στο αυτό το σύνδεσμο και σε περίπτωση που δε λειτουργεί μπορεί να βρεθεί σε αυτό τον σύνδεσμο (Internet Archive).

Η βιβλιοθήκη ABY είναι η βιβλιοθήκη στην οποία στηρίχθηκε η εφαρμογή μας για την εκτέλεση για την εκτέλεση των πρωταρχικών κρυπτογραφικών και ασφαλούς υπολογισμού διεργασιών. Παρουσιάστηκε στη βιβλιογραφία το 2018 στην εργασία [56]. Πρόκειται για μια βιβλιοθήκη Ασφαλούς Υπολογισμού Δύο Μερών, που υποστηρίζει πρωτόκολλα που βασίζονται σε Boolean και Αριθμητικά Δίκτυα τα οποία μπορούν να συνδυαστούν σε μια εκτέλεση λόγω της υβριδικής της φύσης. Για την περίπτωση των Boolean δικτύων υποστηρίζονται τα πρωτόκολλα GMW και Yao, ενώ για την περίπτωση των αριθμητικών δικτύων υποστηρίζεται το πρωτόκολλο BGW. Τα τρία αυτά πρωτόκολλα αναλύθηκαν στο Κεφάλαιο 4. Τέλος, το ABY υποστηρίζει με εύχρηστο τρόπο SIMD πράξεις οι οποίες έχουν αρκετά μεγάλο αντίκτυπο στη βελτίωση της απόδοσης της εκτέλεσης ενός κυκλώματος.

5.2.1 Εξέταση της καταλληλότητας της ABY για τους σκοπούς της MPC-BLAS

Πριν προχωρήσουμε με την ανάλυση της βιβλιοθήκης ας εξετάσουμε τα κριτήρια επιλογής της. Αρχικά, εφόσον θέλουμε να υλοποιήσουμε μια διεπαφή για C (τη CBLAS) είμαστε σχεδόν αναγκασμένοι να χρησιμοποιήσουμε C/C++. Αυτό συνεπάγεται πως η κυριότερη μας απαίτηση για όποιες κρυπτογραφικές βιβλιοθήκες επιθυμούμε να χρησιμοποιήσουμε, είναι να διαθέτουν διεπαφή για C/C++ και να μπορούν να μεταγλωττιστούν ώστε να είναι συνδέσιμες (linkable) από προγράμματα που χρησιμοποιούν τη C/C++. Ακόμα μια απαίτηση είναι ότι επιθυμούμε από τη βιβλιοθήκη που θα χρησιμοποιήσουμε να υποστηρίζει πράξεις με μι-

```
/*
 * Enumerated and derived types
 */
#define CBLAS_INDEX size_t /* this may vary between platforms */

/*
 * Integer type
 */
#ifndef CBLAS_INT
#ifdef WeirdNEC
  #define CBLAS_INT int64_t
#else
  #define CBLAS_INT int32_t
#endif
#endif


float  cblas_sdsdot(const CBLAS_INT N, const float alpha, const float *X,
              const CBLAS_INT incX, const float *Y, const CBLAS_INT incY);
double cblas_dsdot(const CBLAS_INT N, const float *X, const CBLAS_INT incX, const float *Y,
              const CBLAS_INT incY);
float  cblas_sdot(const CBLAS_INT N, const float  *X, const CBLAS_INT incX,
              const float  *Y, const CBLAS_INT incY);
double cblas_ddot(const CBLAS_INT N, const double *X, const CBLAS_INT incX,
              const double *Y, const CBLAS_INT incY);
void   cblas_cdotu_sub(const CBLAS_INT N, const void *X, const CBLAS_INT incX,
               const void *Y, const CBLAS_INT incY, void *dotu);
void   cblas_cdotc_sub(const CBLAS_INT N, const void *X, const CBLAS_INT incX,
               const void *Y, const CBLAS_INT incY, void *dotc);
void   cblas_zdotu_sub(const CBLAS_INT N, const void *X, const CBLAS_INT incX,
               const void *Y, const CBLAS_INT incY, void *dotu);
void   cblas_zdotc_sub(const CBLAS_INT N, const void *X, const CBLAS_INT incX,
               const void *Y, const CBLAS_INT incY, void *dotc);
```

Σχήμα 5.2: Διεπαφή CBLAS Επιπέδου 1.

γαδικούς αριθμούς μονής και διπλής ακρίβειας. Σαν τελευταία απαίτηση, θα επιθυμούσαμε να έχει και κατάλληλη τεκμηρίωση και παραδείγματα σχετικά με τον πως να τη χρησιμοποιήσουμε. Δυστυχώς, δύο από τις πιο γνωστές βιβλιοθήκες για SMPC, η MP-SPDZ και η SCALE-MAMBA δεν ικανοποιούν την πρώτη και τελευταία απαίτηση. Ένα ακόμα σημαντικό αυτών των δύο βιβλιοθηκών είναι πως αρχικά υποστηρίζουν περισσότερους των δύο συμμετέχοντες και επίσης υποστηρίζουν πρωτόκολλα όπως το SPDZ που είναι ασφαλή ενάντια σε ενεργητικούς αντιπάλους. Χαρακτηριστικά σαν αυτά θα τα επιθυμούσαμε να τα έχει η βιβλιοθήκη μας. Δυστυχώς, όμως στην περίπτωση της MP-SPDZ το οποίο υποστηρίζει διεπαφή σε C/C++ η τεκμηρίωση που διαθέτει για αυτήν είναι μηδαμινή. Η περίπτωση της SCALE-MAMBA δεν υποστηρίζει καθόλου διεπαφή σε C/C++. Αυτό συμβαίνει διότι η βιβλιοθήκη SCALE-MAMBA, βασίζεται στην αρχιτεκτονική της υλοποίησης ενός εικονικού κατανεμημένου επεξεργαστή με δική του συμβολική γλώσσα, της οποίας οι εντολές εκτελούνται πάνω σε πρωτόκολλα SMPC. Από της λίγες βιβλιοθήκες της βιβλιογραφίας που υποστηρίζουν διεπαφή και σύνδεση με C/C++ είναι η ABY. Αυτή ικανοποιεί την πρώτη και τρίτη απαίτηση, ωστόσο έχει το μειονέκτημα πως δεν υποστηρίζει πράξεις με μιγαδικούς αριθμούς και με αριθμούς κινητής υποδιαστολής διπλής ακρίβειας για ορισμένες πράξεις, όπως αυτή της τε-

τραγωνικής ρίζας που χρησιμοποιείται σε κάποιες BLAS συναρτήσεις.

5.2.2 Εγκατάσταση

Η ABY είναι αρκετά εύχρηστη και αυτό αποτελεί και έναν από τους σημαντικότερους λόγους που την επιλέξαμε, αφού δε βρέθηκε κάποια βιβλιοθήκη που να ικανοποιεί όλες τις σχεδιαστικές μας ανάγκες. Στο σημείο αυτό να αναφέρουμε πως η βιβλιοθήκη ABY δε διαθέτει δημοσιευμένες μεταγλωττισμένες εκδόσεις σε μορφή object αρχείων (object files) ώστε να μπορεί κάποιο πρόγραμμα κατευθείαν να συνδεθεί (link) με αυτές χωρίς να χρειαστεί να μεταγλωττίσει τη βιβλιοθήκη από την αρχή. Ο ευκολότερος τρόπος για να εγκατασταθεί η βιβλιοθήκη ABY από το αποθετήριο της στο Github είναι μέσω της CMake συνάρτησης FetchContent_Declare() του CMake. Μέσω αυτής της συνάρτησης μπορούμε να προσθέσουμε το ABY σε κάποιο προϋπάρχον CMake πρότζεκτ ως απλά προσθέτοντας τον κώδικα του Σχήματος 5.3 στην αρχή του αρχείου CMakeLists.txt

```
include(FetchContent)
FetchContent_Declare(
  ABY
  GIT_REPOSITORY https://github.com/encryptogroup/ABY.git
)
FetchContent_MakeAvailable(ABY)

add_executable(my_application my_application.cpp)
target_link_libraries(my_application ABY::aby)
```

Σχήμα 5.3: Διαδικασία εγκατάστασης της βιβλιοθήκης ABY με χρήση της CMakeσυνάρτησης FetchContent_Declare() θεωρώντας ότι θέλουμε να μεταγλωττίσουμε ένα αρχείο πηγαίου κώδικα με όνομα my_application.cpp και με όνομα τελικού εκτελέσιμου my_application

Ένας εναλλακτικός τρόπος να εγκαταστήσουμε τη βιβλιοθήκη ABY είναι μέσω του αποθετηρίου του στο Github και στη συνέχεια χρησιμοποιώντας το CMake. Για να το επιτύχουμε αυτό πρέπει το πρότζεκτ στο οποίο θέλουμε να την χρησιμοποιήσουμε να το περάσουμε πρώτα στο Git και στη συνέχεια να περάσουμε επίσης και στο CMake. Δε θα μπούμε σε περισσότερες λεπτομέρειές σχετικά με το πως μπορούμε να το επιτύχουμε αυτό καθώς είναι μια σχετικά απλή διαδικασία και υπάρχει πληθώρα βιβλίων γύρω από τα προγράμματα αυτά στη βιβλιογραφία. Αφού περάσουμε το πρότζεκτ μας στο Git και στη συνέχεια στο CMake η ABY μπορεί να εγκατασταθεί στο πρότζεκτ μας εκτελώντας τον κώδικα κελύφους του Σχήματος 5.4. Στο βήμα αυτό συνδέουμε το git πρότζεκτ μας με ένα άλλο git πρότζεκτ που βρίσκεται σε κάποιο απομακρυσμένο αποθετήριο. Στη συνέχεια για να μπορούμε να έχουμε πρόσβαση στη βιβλιοθήκη ABY μέσα από το CMake πρότζεκτ μας πρέπει να προσθέσουμε τον κώδικα του Σχήματος 5.5 στην αρχή του αρχείου CMakeLists.txt.

```
#!/bin#bash

cd YOUR_PROJECT_FOLDER_NAME
git submodule add https://github.com/encryptogroup/ABY ./extern
```

Σχήμα 5.4: Διαδικασία προσθήκης της βιβλιοθήκης ABY ως git submodule στο git πρότζεκτ μας.

```
find_package(ABY QUIET)
if(ABY_FOUND)
   message(STATUS "Found ABY")
elseif (NOT ABY_FOUND AND NOT TARGET ABY::aby)
   message("ABY was not found: add ABY subdirectory")
   add_subdirectory(extern/ABY)
endif()

add_executable(my_application my_application.cpp)
target_link_libraries(my_application ABY::aby)
```

Σχήμα 5.5: Διαδικασία εγκατάστασης της βιβλιοθήκης ABY θεωρώντας ότι προηγουμένας έχουμε προσθέσει τη βιβλιοθήκη ABY ως git submodule στην τοποθεσία ./extern μέσα στο πρότζεκτ μας, σύμφωνα με το Σχήμα code:aby-cmake-2 και ότι θέλουμε να μεταγλωττίσουμε ένα αρχείο πηγαίου κώδικα με όνομα my_application.cpp και με όνομα τελικού εκτελέσιμου my_application.

5.2.3 Χρήση

Ας δούμε τώρα ένα παράδειγμα σχετικά με το πως μπορούμε να χρησιμοποιήσουμε την ABY. Η βιβλιοθήκη αυτή διαθέτει τρεις κύριες κλάσεις οι οποίες εμφανίζονται παντού στη χρήση της. Την κλάση ABYParty, η οποία αντιπροσωπεύει μια SMPC οντότητα. Η ABYParty χρησιμοποιείται για την αρχικοποίηση του συστήματος με τις απαραίτητες παραμέτρους, όπως η διεύθυνση και η πόρτα του άλλου μέρους. Εφόσον η ABY υποστηρίζει μόνο Ασφαλή Υπολογισμό 2 Μερών, είναι αντιληπτό πως θα χρειαστούν δύο στιγμιότυπα αυτής της κλάσης. Αυτές οι οντότητες μπορούν να δημιουργούνται είτε από δύο διαφορετικές διεργασίες που προέρχονται από δύο διαφορετικούς πηγαίους κώδικες, είτε από μια διεργασία η οποία εκτελείται με κάποιον τρόπο δύο φορές (είτε χειροκίνητα είτε μέσω κάποιας κλήσης συστήματος fork()). Σε μια διεργασία μπορεί να υπάρχει μόνο ένα στιγμιότυπο της κλάσης ABYParty. Η δεύτερη σημαντική κλάση είναι αυτή του δικτύου, δηλαδή είτε η BooleanCircuit είτε η ArithmeticCircuit. Η κλάση διαθέτει κατάλληλες συναρτήσεις για το χτίσιμο του επιθυμητού δικτύου. Η τρίτη σημαντική κλάση είναι η Sharing. Αυτή δε συνδέεται απαραίτητα με το Shamir Secret Sharing. Στιγμιότυπα αυτής της κλάσης συνήθως δε δημιουργούνται χειροκίνητα αλλά μας επιστρέφονται όταν καλούμε μια συνάρτηση προσθήκης πύλης σε ένα δίκτυο. Ουσιαστικά είναι μια κλάση που κρατάει αναφορές στα καλώδια εξόδου μιας πύλης ώστε να μπορούμε να τα χρησιμοποιήσουμε σαν μια έξοδο σε μια επόμενη πύλη. Η βιβλιο-

θήκη αυτή είναι κατασκευασμένη με τέτοιο τρόπο ώστε να μπορεί να τρέξει τόσο τοπικά όσο και απομακρυσμένα.

Ας εξετάσουμε τώρα τη διαδικασία που πρέπει να ακολουθήσει ένας συμμετέχον ώστε να αρχικοποιήσει και να εκτελέσει ένα Boolean κύκλωμα που να υπολογίζει το εσωτερικό γινόμενο δύο διανυσμάτων με πραγματικούς αριθμούς κινητής υποδιαστολής μονής ακρίβειας από τα οποία γνωρίζει μόνο το πρώτο διάνυσμα, ενώ ο άλλος συμμετέχον γνωρίζει μόνο το δεύτερο διάνυσμα. Ουσιαστικά θα αναλύσουμε ένα παράδειγμα υλοποίησης του παρακάτω προτύπου BLAS :

```
float  cblas_sdsdot(const CBLAS_INT N, const float alpha, const float *X,
                    const CBLAS_INT incX, const float *Y, const CBLAS_INT incY);
```

Το πρώτο πρόβλημα που αντιμετωπίζουμε είναι με πιο μηχανισμό θα αντιληφθούμε ότι ένας συμμετέχον δε γνωρίζει την τιμή μιας παραμέτρου. Για να επιλύσουμε το πρόβλημα αυτό μπορούμε να υποθέσουμε ότι όταν ένας συμμετέχον δε γνωρίζει την τιμή μιας παραμέτρου δίνει στο όρισμα την τιμή nullptr. Ακόμα υποθέτουμε πως μια float τιμή διαθέτει 32 bits, για αυτό και χρησιμοποιούμε την εξής μακροεντολή *#define BITLEN 32*. Η διαδικασία που πρέπει να ακολουθήσουμε για να υπολογίσουμε το παραπάνω πρότυπο είναι η εξής :

1. Δημιουργούμε ένα στιγμιότυπο της κλάσης ABYParty, περνώντας όλες τις απαραίτητες παραμέτρους που χρειάζεται για να γίνει η εκκίνηση της βιβλιοθήκης. Σημαντικό είναι να αναφέρουμε πως η παράμετρος port πρέπει να είναι ίδια για τους δύο συμμετέχοντες και ανεξαρτήτως αν εκτελούμε τοπικά ή απομακρυσμένα τη βιβλιοθήκη.

   ```
   ABYParty *party = new ABYParty(SERVER, addr="127.0.0.1", port=7766, seclvl=LT,
                       bitlen=32, n_threads=2, mg_algo=MT_OT, reserve_gates=4000000, "OUR_ABY_CIRCUIT_PATH_HERE");
   ```

2. Στην συνέχεια παίρνουμε πρόσβαση στο δίκτυο που επιθυμούμε του αντικειμένου ABYParty που δημιουργήσαμε. Τα στιγμιότυπα αυτά των δικτύων τα χρησιμοποιούμε για να προσθέσουμε πύλες στο δίκτυο μας.

   ```
   BooleanCircuit *boolean_circuit = (BooleanCircuit *) party->GetSharings()[S_BOOL]->GetCircuitBuildRoutine();
   ```

3. Έπειτα, εισάγουμε τις πύλες εισόδων στο δίκτυο δίνοντας ως ορίσματα σε αυτές τις επιθυμητές εισόδους. Αυτό το επιτυγχάνουμε καλώντας της μεθόδους PutINGate και PutDummyINGate. Η πρώτη χρησιμοποιείται αν ο συμμετέχον θέλει να εισάγει κάποια ιδιωτική είσοδο. Για κάθε ιδιωτική είσοδο που εισάγει κάποιος συμμετέχον ο άλλος συμμετέχον πρέπει να χρησιμοποιήσει τη δεύτερη μέθοδο. Δηλαδή έχουμε ένα ζεύγος κλήσεων των δύο μεθόδων για κάθε ιδιωτική είσοδο που εισάγεται στο κύκλο μα. Οι

διαστάσεις των πύλεων IN πρέπει να συμπίπτουν. Ακόμα, να αναφέρουμε πως υπάρχουν επίσης και οι μέθοδοι PutSharedINGate και PutCONSGate, για την είσοδο από κοινού γνωστών τιμών και σταθερών τιμών στο κύκλωμα αντίστοιχα αλλά και οι μέθοδοι PutDummySIMDINGate και PutDummySIMDINGate για την είσοδο πολλαπλών τιμών οι οποίες λειτουργούν όπως οι απλές πύλες εισόδου ωστόσο ενεργοποιούν τη δυνατότητα για SIMD πράξεις στις μετέπειτα πύλες.

```
share *s_X, *s_Y

if (s_X != nullptr) {
   float input_vector[N];
   convert_cblas_format_to_array<float>(N, s_X, inc, input_vector);
   s_X = boolean_circuit->PutSIMDINGate(N,
                        (uint32_t *)s_X,
                        BITLEN,
                        boolean_circuit->GetRole()));
} else {
   s_X = boolean_circuit->PutDummySIMDINGate(N, BITLEN);
}

if (s_X != nullptr) {
   float input_vector[N];
   convert_cblas_format_to_array<float>(N, s_Y, inc, input_vector);
   s_Y = boolean_circuit->PutSIMDINGate(N,
                        (uint32_t *)s_Y,
                        BITLEN,
                        boolean_circuit->GetRole()));
} else {
   s_Y = boolean_circuit->PutDummySIMDINGate(N, BITLEN);
}
```

Είναι σχεδόν προφανές ότι ο παραπάνω κώδικας μπορεί να βελτιωθεί θεωρώντας μια συνάρτηση και περνώντας σαν ορίσματα σε αυτή τα N, X, incX και my_application, ωστόσο για να μη γίνει δυσανάγνωστη η εργασία προτιμήσαμε την πιο απλή του μορφή[1].

4. Σε αυτό το σημείο όπου οι συμμετέχοντες αφού έχουν εισάγει όλες τις εισόδους τους στο κύκλωμα, μπορούν να εισάγουν τις ενδιάμεσες πύλες του κυκλώματος. Τα στιγμιότυπα των δικτύων που αποκτούμε στο Βήμα 2 διαθέτουν μεγάλη ποικιλομορφία πυλών που μπορούν να εισαχθούν.

```
share *s_multiply, *s_add;

s_multiply = boolean_circuit->PutFPGate(s_X-, s_Y, MUL, BITLEN, N);
```

[1]Βελτιστοποιήσεις όπως αυτή, αξιοποιούνται μαζικά στον κώδικα της MPC-BLAS, που θα εξετάσουμε στη συνέχεια, καθώς παρατηρήθηκε πως κατά την υλοποίηση της υπάρχει μεγάλη επαναληψημότητα αλγοριθμικής λογικής.

```
uint32_t pos[1] = {0};
s_add = boolean_circuit->PutSubsetGate(s_multiply, pos, 1);

for (uint32_t i = 1; i < N; i++) {
   pos[0] = i;
   s_add = boolean_circuit->PutFPGate(s_add,
                              boolean_circuit->PutSubsetGate(s_multiply, pos, 1),
                              ADD, BITLEN, 1);
}
```

5. Αφού έχει χτιστεί ολόκληρο το κύκλωμα το μόνο που απομένει στο χτίσιμο του δικτύου είναι να εισαχθούν πύλες εξόδου. Αυτό επιτυγχάνεται μέσω των μεθόδων PutOUTGate.

```
share *s_out;
s_out = mpcblas_get_circuit()->PutOUTGate(this->sharing, ALL);
mpcblas_get_party()->ExecCircuit();
```

6. Τέλος, το μόνο που απομένει είναι τρέξουμε το δίκτυο ώστε να γίνει η αποτίμηση του και οι συμμετέχοντες να λάβουν τις εξόδους της.

```
uint32_t out_bitlen, out_nvals;
uint32_t *out_values;
s_out->get_clear_value_vec(&out_values, &out_bitlen, &out_nvals);

return *((float *)(&out_values[0]));
```

Ολόκληρη η υλοποίηση για την υλοποίηση του πρωτύπου της συνάρτησης του εσωτερικού γινομένου που αναλύσαμε παραπάνω φαίνεται παρακάτω στο Σχήμα. Στην υλοποίηση αυτή έχουμε προσθέσει τις απαραίτητες κλήσεις fork() ώστε να μπορεί να εκτελεστεί και να ελεγχθεί και επίσης έχουμε εισάγει ενδεικτικές εισόδους που υποτίθεται ελέγχει ο κάθε συμμετέχον αυτές είναι οι.

```
#include <iostream>

#include "abycore/sharing/sharing.h"
#include "abycore/circuit/booleancircuits.h"
#include "abycore/aby/abyparty.h"
#include "../extern/ENCRYPTO_utils/src/ENCRYPTO_utils/typedefs.h"
#include "../extern/ENCRYPTO_utils/src/ENCRYPTO_utils/constants.h"

#define CBLAS_INT int32_t

float  cblas_sdsdot(const CBLAS_INT N, const float alpha, const float *X,
            const CBLAS_INT incX, const float *Y, const CBLAS_INT incY) {
ABYParty *party = new ABYParty(SERVER, addr="127.0.0.1", port=7766, seclvl=LT,
                               bitlen=32, n_threads=2, mg_algo=MT_OT, reserve_gates=4000000, "OUR_ABY_CIRCUIT_PATH_HERE");
    BooleanCircuit *boolean_circuit = (BooleanCircuit *) party->GetSharings()[S_BOOL]->GetCircuitBuildRoutine();
```

```
    share *s_X, *s_Y

    if (s_X != nullptr) {
      float input_vector[N];
      convert_cblas_format_to_array<float>(N, s_X, inc, input_vector);
      s_X = boolean_circuit->PutSIMDINGate(N,
                           (uint32_t *)s_X,
                           BITLEN,
                           boolean_circuit->GetRole()));
    } else {
      s_X = boolean_circuit->PutDummySIMDINGate(N, BITLEN);
    }

    if (s_X != nullptr) {
      float input_vector[N];
      convert_cblas_format_to_array<float>(N, s_Y, inc, input_vector);
      s_Y = boolean_circuit->PutSIMDINGate(N,
                           (uint32_t *)s_Y,
                           BITLEN,
                           boolean_circuit->GetRole()));
    } else {
      s_Y = boolean_circuit->PutDummySIMDINGate(N, BITLEN);
    }

    share *s_multiply, *s_add;

    s_multiply = boolean_circuit->PutFPGate(s_X->get_sharing(), s_Y->get_sharing(), MUL, BITLEN, N);

    uint32_t pos[1] = {0};
    s_add = boolean_circuit->PutSubsetGate(s_multiply, pos, 1);

    for (uint32_t i = 1; i < N; i++) {
      pos[0] = i;
      s_add = boolean_circuit->PutFPGate(s_add,
                          boolean_circuit->PutSubsetGate(s_multiply, pos, 1),
                          ADD, BITLEN, 1);
    }

    share *s_out;
    s_out = mpcblas_get_circuit()->PutOUTGate(this->sharing, ALL);
    mpcblas_get_party()->ExecCircuit();

    uint32_t out_bitlen, out_nvals;
    uint32_t *out_values;
    s_out->get_clear_value_vec(&out_values, &out_bitlen, &out_nvals);

    std::cout << *((float *)(&out_values[0])) << std::endl;
}

int main() {
  const N, incX, incY;
  float *X = nullptr, *Y = nullptr;

  if(fork() != -1) {
    float test_vector_1[4] = {1.21, 5.33, 3.345, 4.213};
    std::cout << sdot(4, test_vector_1, 1, nullptr, 1) << std::endl;
```

```
    }

    if (fork() != -1) {
        float test_vector_2[4] = {0.01, 3.82, 1023.905, 92302.43};
        std::cout << sdot(4, nullptr, 1, test_vector_2, 1) << std::endl;
    }
}
```

Σχήμα 5.6: Παράδειγμα προγράμματος με χρήση της βιβλιοθήκης ABY

5.3 Η βιβλιοθήκη MPC-BLAS

Ο κώδικας της βιβλιοθήκης MPC-BLAS βρίσκεται στο αποθετήριο Github σε αυτόν τον σύνδεσμο και επίσης θα διατίθεται σε μορφή .zip στο Ιδρυματικό Αποθετήριο.

Η βιβλιοθήκη MPC-BLAS είναι η βιβλιοθήκη που υλοποιήθηκε στα πλαίσια αυτής της διπλωματικής εργασίας με σκοπό την εφαρμογή των πρωτοκόλλων που εξετάσαμε στο Κεφάλαιο 4. Πρόκειται για την υλοποίηση των όλων των προτύπων BLAS Επιπέδου-1, εκτός από αυτά που σχετίζονται με την περιστροφή πινάκων, για πραγματικούς αριθμούς κινητής υποδιαστολής μονής ακρίβειας με τη χρήση της βιβλιοθήκης ABY. Η ανάπτυξη της έγινε σε C++20 και υποστηρίζεται μόνο για το λειτουργικό σύστημα Linux για τη διανομή Ubuntu 22.04 LTS, στην οποία έγινε η ανάπτυξη και ο έλεγχος της. Ο περιορισμός του λειτουργικού συστήματος Linux προέρχεται από τη βιβλιοθήκη ABY. Αν η βιβλιοθήκη ABY προσθέσει υποστήριξη και για άλλα λειτουργικά συστήματα (π.χ. Windows) τότε η βιβλιοθήκη MPC-BLAS πολύ πιθανόν να μπορεί να χρησιμοποιηθεί χωρίς αλλαγές και σε αυτά καθώς βασίζεται αποκλειστικά σε χαρακτηριστικά και λειτουργίες της C++20[2] και της ABY.

5.3.1 Εγκατάσταση

Η διαδικασία εγκατάστασης της βιβλιοθήκης MPC-BLAS είναι παρόμοια με αυτή της βιβλιοθήκης ABY, αφού και οι δύο βρίσκονται στο αποθετήριο Github και βασίζονται στο CMake. Έτσι χωρίς παραπάνω λεπτομέρειες οι τρόποι για να εγκατασταθεί η βιβλιοθήκη MPC-BLAS παρουσιάζονται παρακάτω. Στο Σχήμα 5.7 παρουσιάζεται ο πρώτος τρόπος εγκατάστασης .

Ενώ στα Σχήματα 5.9, 5.8 παρουσιάζεται ο δεύτερος τρόπος εγκατάστασης :

[2]Προφανώς για να συμβεί αυτό θα πρέπει και η C++20 να υποστηρίζεται στα λειτουργικά συστήματα αυτά, ωστόσο αυτό συμβαίνει για όλες σύγχρονες διανομές λειτουργικών συστημάτων

```
include(FetchContent)
FetchContent_Declare(
      mpc-blas
      GIT_REPOSITORY https://github.com/st1064870/mpc-blas.git
)
FetchContent_MakeAvailable(mpc-blas)

add_executable(my_application my_application.cpp)
target_link_libraries(my_application STATIC MPC-BLAS::mpc-blas)
```

Σχήμα 5.7: Διαδικασία εγκατάστασης της βιβλιοθήκης MPC-BLAS με χρήση της CMake συνάρτησης FetchContent_Declare() θεωρώντας ότι θέλουμε να μεταγλωττίσουμε ένα αρχείο πηγαίου κώδικα με όνομα my_application.cpp και με όνομα τελικού εκτελέσιμου my_application.

```
#!/bin#bash

cd YOUR_PROJECT_FOLDER_NAME
git submodule add https://github.com/st1064870/mpc-blas ./extern
```

Σχήμα 5.8: Διαδικασία προσθήκης της βιβλιοθήκης ABY ως git submodule στο git πρότζεκτ μας.

```
add_subdirectory(extern/mpc-blas)

add_executable(my_application my_application.cpp)
target_link_libraries(my_application STATIC MPC-BLAS::mpc-blas)
```

Σχήμα 5.9: Διαδικασία εγκατάστασης της βιβλιοθήκης MPC-BLAS θεωρώντας ότι προηγουμένας έχουμε προσθέσει την βιβλιοθήκη ABY ως git submodule στην τοποθεσία ./extern μέσα στο πρότζεκτ μας, σύμφωνα με το Σχήμα 5.4 και ότι θέλουμε να μεταγλωττίσουμε ένα αρχείο πηγαίου κώδικα με όνομα my_application.cpp και με όνομα τελικού εκτελέσιμου my_application.

5.3.2 Διεπαφή και χρήση

Η διεπαφή της βιβλιοθήκης έχει σχεδιαστεί με βασικότερο στόχο να κάνει τις ελάχιστες δυνατές αλλαγές ώστε να ένα πρόγραμμα γραμμένο για τη διεπαφή CBLAS να μπορεί με ελάχιστες αλλαγές να μετατραπεί σε ένα πρόγραμμα γραμμένο για τη διεπαφή MPC-BLAS. Ο δευτερεύον σχεδιαστικός της στόχος είναι η χρηστικότητα από την κρυπτογραφική σκοπιά αλλά και αυτή της ασφάλειας. Πιο συγκεκριμένα, επειδή πολλές συναρτήσεις και υπορουτίνες της CBLAS διεπαφής είναι μαθηματικά αντιστρέψιμες κρίναμε πολύ σημαντικό, για τη χρηστικότητα της βιβλιοθήκης, την ύπαρξη δυνατότητας να δημιουργηθεί αλυσίδα κλήσεων συναρτήσεων ή υπορουτινών χωρίς να φανερώνονται τα ενδιάμεσα αποτελέσματα από τις κλήσεις αυτές. Για παράδειγμα, να μπορεί να υπολογιστεί η Ευκλείδεια Νόρμα (xNRM2) του αποτελέσματος μιας AXPY πράξης (xAXPY), χωρίς να φανερωθεί το ενδιάμεσο αποτέλεσμα

της AXPY πράξης σε κανέναν από τους συμμετέχοντες. Τελευταίος σχεδιαστικός στόχος της MPC-BLAS είναι η διαλειτουργικότητα με τη CBLAS διεπαφή, κάτι που όπως αποδείχθηκε κατά την υλοποίηση είναι σχετικά εύκολο να επιτευχθεί. Στη συνέχεια θα αναλύσουμε βασικά στοιχεία της διεπαφής της βιβλιοθήκης MPC-BLAS. Στις επόμενες παραγράφους θα αναλύσουμε ορισμένα βασικά σημεία της βιβλιοθήκης MPC-BLAS στα οποία πρέπει να δώσουμε ιδιαίτερη σημασία καθώς διαφοροποιούν τη χρήση της από τη χρήση της CBLAS διεπαφής.

Η πρώτη λεπτομέρεια στην οποία πρέπει να δώσουμε ιδιαίτερη έμφαση είναι ο όνομα των συναρτήσεων. Οι συναρτήσεις της CBLAS διεπαφής έχουν πρόθεμα cblas και έχουν τη μορφή cblas_xxxxx. Οι συναρτήσεις της βιβλιοθήκης MPC-BLAS έχουν πρόθεμα mpcblas και είναι της μορφής mpcblas_xxxx. Η πρώτη συνάρτηση που πρέπει να κληθεί για να αρχικοποιηθεί η βιβλιοθήκη MPC-BLAS είναι η συνάρτηση mpcblas_initialize() και αντίστοιχα οι τελευταία συνάρτηση που πρέπει να κληθεί για να από-αρχικοποιηθεί η βιβλιοθήκη είναι mpclbas_uninitialize(). Στην πρώτη συνάρτηση περνάμε ως ορίσματα παραμέτρους όπως η διεύθυνση IP και η πόρτα στην οποία θα κάνει δέσιμο (bind) ο συμμετέχον αλλά και το bitlen που είναι το μέγεθος κάθε μοναδικής τιμής στο Boolean δίκτυο. Για παράδειγμα, αν στο δίκτυο μας έχουμε εισόδους float με μέγεθος 4 byte, το bitlen πρέπει να είναι 32 bit. Τα πρότυπα και των δύο συναρτήσεων φαίνονται στο Σχήμα 5.10.

```
void mpcblas_initialize(e_role role,
                        const std::string &address,
                        int port,
                        int bitlen,
                        seclvl security_level = LT,
                        int n_threads = 2,
                        e_mt_gen_alg mg_algo = MT_OT,
                        int reserve_gates = 4000000,
                        const std::string &aby_circ_dir = "/home/imslab/ABY/bin/circ/.");
void mpcblas_uninitialize();
```

Σχήμα 5.10: Τα πρότυπα των συναρτήσεων mpcblas_initialize() και mpcblas_uninitialize().

Το επόμενο σημείο στο οποίο πρέπει να σταθούμε είναι η μακροεντολή MPCBLAS_IGNORE. Η μακροεντολή αυτή χρησιμοποιείται από κάποιον συμμετέχον ως αντικατάστατο (placeholder) ενός ορίσματος που δε γνωρίζει ο συμμετέχον, αλλά γνωρίζει ο άλλος συμμετέχον. Η μακροεντολή αυτή εισήχθη ώστε να διατηρηθούν σχεδόν αναλλοίωτα τα πρότυπα της διεπαφής CBLAS. Εσωτερικά η μακροεντολή αυτή προκαλεί τη χρήση της PutDummyINGate της βιβλιοθήκης ABY από την πλευρά του συμμετέχον που τη δίνει ως όρισμα.

Επόμενο σημαντικό σημείο είναι η κλάσεις mpcblas_value και Ένα αντικείμενο (στιγμιότυπο) αυτής της συνάρτησης επιστρέφεται από κάθε συνάρτηση (προσοχή αναφερόμαστε σε συναρτήσεις και όχι ρουτίνες) που καλείται και επιστρέφει μια μοναδική τιμή. Το αντικείμενο αυτό αποθηκεύει εσωτερικά αντικείμενα τύπου share* της ABY. Η ιδιότητα αυτή κάνει ένα

αντικείμενο της κλάσης mpcblas_value κατάλληλο ώστε να περαστεί ως όρισμα σε κάποια άλλη συνάρτηση MPC-BLAS. Προφανώς, για να μπορεί να επιτευχθεί αυτό πρέπει να υπάρχουν και τα κατάλληλα πρότυπα για κάθε συνάρτηση MPC-BLAS. Στο Σχήμα 5.11 φαίνονται τα πρότυπα της βιβλιοθήκης για τη συνάρτηση sdot. Παρατηρούμε πως σε κάθε συνάρτηση της διεπαφής μπορεί να περαστεί ως όρισμα ένα αντικείμενο τύπου mpcblas_value.

```
mpcblas_value<float> *mpcblas_sdot(int N, std::optional<float *> X, int incX,
                                   std::optional<float *> Y, int incY);
mpcblas_value<float> *mpcblas_sdot(int N, mpcblas_value<float *> *s_X, int incX,
                                   std::optional<float *> Y, int incY);
mpcblas_value<float> *mpcblas_sdot(int N, std::optional<float *> X, int incX,
                                   mpcblas_value<float *> *s_Y, int incY);
mpcblas_value<float> *mpcblas_sdot(int N, mpcblas_value<float *> *X, int incX,
                                   mpcblas_value<float *> *Y, int incY);
```

Σχήμα 5.11: Τα πρότυπα των της βιβλιοθήκης MPC-BLAS για τη συναρτήση sdot.

Η σημαντικότερη ίσως μέθοδος ενός αντικειμένου της κλάσης mpcblas_value είναι η get_value(). Η μέθοδος αυτή μας επιτρέπει να μετατρέψουμε την κρυπτογραφημένη μορφή της τιμής που κρατάει εσωτερική το αντικείμενο σε απλή μορφή ώστε να μπορεί να ανακτηθεί το αποτέλεσμα του υπολογισμού. Η μέθοδος αυτή ουσιαστικά προσθέτει μια πύλη εξόδου στο Boolean κύκλωμα, μέσω της συνάρτησης PutOUTGate() της ABY, και ξεκινάει τη διαδικασία εκτέλεσης και αποτίμησης του κυκλώματος, μέσω της συνάρτησης ExecCircuit(). Μέχρι τη στιγμή της κλήσης της μεθόδου αυτής, όλες οι κλήσεις BLAS συναρτήσεων της βιβλιοθήκης απλώς χτίζουν το κύκλωμα, η μία ξεκινώντας από το σημείο που σταμάτησε η προηγούμενη.

Στο Σχήμα 5.14 παρατηρούμε ένα παράδειγμα υπολογισμού της του εσωτερικού γινομένου δύο διανυσμάτων με τη χρήση της CBLAS και με τη χρήση της MPC-BLAS, στο οποίο είναι με γκρι χρώμα σημειωμένες οι διαφορές τους.

Στο Σχήμα 5.18 παρατηρούμε τη χρήση της MPC-BLAS για έναν πιο σύνθετο υπολογισμό. Η συνάρτηση αυτή υπολογίζει το αποτέλεσμα της έκφρασης $\text{NORM}_2(\mathbf{x}\mathbf{y}^T\mathbf{x} + \mathbf{y})$.

Το μέρος διεπαφής της MPC-BLAS που περιλαμβάνει τις μαθηματικές αντίστοιχες μαθηματικές συναρτήσεις της CBLAS φαίνεται στο Σχήμα 5.19, ενώ το μέρος τις διεπαφής της MPC-BLAS που περιλαμβάνει συναρτήσεις για τις οποίες δεν υπάρχουν αντίστοιχες στη CBLAS, όπως π.χ. η mpcblas_initialize(), φαίνεται στο Σχήμα 5.20. Τα δύο μέρη αυτά παρουσιάζονται σε δύο διαφορετικά σχήματα εδώ για λόγους καλύτερης κατανόησης, ωστόσο και τα δύο αυτά μέρη κώδικα αποτελούν μέρος ενός ενιαίου αρχείου του ExecCircuit().

```
/*
 * Enumerated and derived types
 */
#define MPCBLAS_INDEX uint32_t  /* this may vary between platforms */
```

[b]0.5

```
result_1 = cblas_sdot(4,
                      test_vector_1, 1,
                      test_vector_2, 1);
```

Σχήμα 5.12: Χρήση CBLAS.

[b]0.5

```
mpcblas_initialize(SERVER, "127.0.0.1", 7766, bitlen);

result_1 = mpcblas_sdot(4,
                        test_vector_1, 1,
                        MPCBLAS_IGNORE, 1)
           ->get_value();

mpcblas_uninitialize();
```

Σχήμα 5.13: Χρήση MPC-BLAS.

Σχήμα 5.14: Παράδειγμα υπολογισμού του εσωτερικού γινομένου δύο διανυσμάτων με τη χρήση της CBLAS και της MPC-BLAS, στο οποίο είναι σημειωμένο με γκρι χρώμα οι γραμμές στις οποίες υπάρχουν αλλαγές που πρέπει να γίνουν στον κώδικα της πρώτης για να εκτελεστεί στα πλαίσια της δεύτερης.

[b]0.5

```
cblas_saxpy(4,
        cblas_sdot(4,
                test_vector_1, 1,
                test_vector_2, 1),
        test_vector_1, 1,
        test_vector_2, 1);



result = cblas_snrm2(4, test_vector_2, 1);
```

Σχήμα 5.15: Χρήση CBLAS.
[b]0.5

```
mpcblas_initialize(CLIENT, "127.0.0.1", 7766, bitlen);

auto y = mpcblas_saxpy(4,
                mpcblas_sdot(4,
                        MPCBLAS_IGNORE, 1,
                        test_vector_2, 1),
                MPCBLAS_IGNORE, 1,
                test_vector_2, 1);
result_2 = mpcblas_snrm2(4, y, 1)
        ->get_value();

mpcblas_uninitialize();
```

Σχήμα 5.16: Χρήση MPC-BLAS.

0.5

```
mpcblas_initialize(CLIENT, "127.0.0.1", 7766, bitlen);

mpcblas_saxpy(4,
                mpcblas_sdot(4,
                        MPCBLAS_IGNORE, 1,
                        test_vector_2, 1),
                MPCBLAS_IGNORE, 1,
                test_vector_2, 1);
          ->get_value();

mpcblas_uninitialize();
```

Σχήμα 5.17: Χρήση MPC-BLAS.

Σχήμα 5.18: Παράδειγμα υπολογισμού της μαθηματικής παράστασης $\text{NORM}_2(\mathbf{x}\mathbf{y}^T\mathbf{x} + \mathbf{y})$ με τη χρήση της CBLAS και της MPC-BLAS, στο οποίο είναι σημειωμένο με γκρι χρώμα οι γραμμές στις οποίες υπάρχουν αλλαγές που πρέπει να γίνουν στον κώδικα της πρώτης για να εκτελεστεί στα πλαίσια της δεύτερης.

```
/*
 * ========================================================================
 * Prototypes for level 1 BLAS functions (complex are recast as routines)
 * ========================================================================
 */

mpcblas_value<float> *mpcblas_sdot(int N, std::optional<float *> X, int incX,
                        std::optional<float *> Y, int incY);
mpcblas_value<float> *mpcblas_sdot(int N, mpcblas_value<float *> *s_X, int incX,
                        std::optional<float *> Y, int incY);
mpcblas_value<float> *mpcblas_sdot(int N, std::optional<float *> X, int incX,
                        mpcblas_value<float *> *s_Y, int incY);
mpcblas_value<float> *mpcblas_sdot(int N, mpcblas_value<float *> *X, int incX,
                        mpcblas_value<float *> *Y, int incY);

mpcblas_value<float> *mpcblas_snrm2(int N, std::optional<float *> X, int incX);
mpcblas_value<float> *mpcblas_snrm2(int N, mpcblas_value<float *> *s_X, int incX);

mpcblas_value<float> *mpcblas_sasum(int N, std::optional<float *> X, int incX);
mpcblas_value<float> *mpcblas_sasum(int N, mpcblas_value<float *> *s_X, int incX);

mpcblas_value<MPCBLAS_INDEX> *mpcblas_isamax(int N, std::optional<float *> X, int incX);
mpcblas_value<MPCBLAS_INDEX> *mpcblas_isamax(int N, mpcblas_value<float *> *s_X, int incX);

/*
 * ========================================================================
 * Prototypes for level 1 BLAS routines
 * ========================================================================
 */

mpcblas_values<types_list<share *, share *>, types_list<float *, float *>> *
mpcblas_sswap(int N, std::optional<float *> X, int incX,
          std::optional<float *> Y, int incY);
mpcblas_values<types_list<share *, share *>, types_list<float *, float *>> *
mpcblas_sswap(int N, mpcblas_value<float *> *s_X, int incX,
          std::optional<float *> Y, int incY);
mpcblas_values<types_list<share *, share *>, types_list<float *, float *>> *
mpcblas_sswap(int N, std::optional<float *> *X, int incX,
          mpcblas_value<float *> *&s_Y, int incY);
void mpcblas_sswap(int N, mpcblas_value<float *> *&s_X, int incX, mpcblas_value<float *> *&s_Y, int incY);

mpcblas_value<float *> *mpcblas_scopy(int N, std::optional<float *> X, int incX,
                           std::optional<float *> Y, int incY);
mpcblas_value<float *> *mpcblas_scopy(int N, mpcblas_value<float *> *s_X, int incX,
                           std::optional<float *> Y, int incY);
void mpcblas_sscopy(int N, std::optional<float *> X, int incX,
               mpcblas_value<float *> *&s_Y, int incY);
void mpcblas_sscopy(int N, mpcblas_value<float *> *s_X, int incX,
               mpcblas_value<float *> *&s_Y, int incY);

mpcblas_value<float *> *mpcblas_saxpy(int N, std::optional<float> alpha, std::optional<float *> X,
                           int incX, std::optional<float *> Y, int incY);
mpcblas_value<float *> *mpcblas_saxpy(int N,
                           mpcblas_value<float> *s_alpha,
                           std::optional<float *> X,
                           int incX,
```

```
                        std::optional<float *> Y,
                        int incY);
mpcblas_value<float *> *mpcblas_saxpy(int N,
                        std::optional<float *> s_alpha,
                        mpcblas_value<float *> *s_X,
                        int incX,
                        std::optional<float *> Y,
                        int incY);
mpcblas_value<float *> *mpcblas_saxpy(int N,
                        mpcblas_value<float> *s_alpha,
                        mpcblas_value<float *> *s_X,
                        int incX,
                        std::optional<float *> Y,
                        int incY);
void mpcblas_saxpy(int N,
            std::optional<float> alpha,
            std::optional<float *> X,
            int incX,
            mpcblas_value<float *> *&s_Y,
            int incY);
void mpcblas_saxpy(int N,
            mpcblas_value<float> s_alpha,
            std::optional<float *> X,
            int incX,
            mpcblas_value<float *> *&s_Y,
            int incY);
void mpcblas_saxpy(int N,
            mpcblas_value<float> *alpha,
            mpcblas_value<float *> *s_X,
            int incX,
            mpcblas_value<float *> *&s_Y,
            int incY);

mpcblas_value<float *> *mpcblas_sscal(int N, std::optional<float> alpha, std::optional<float *> X, int incX);
void mpcblas_sscal(int N, mpcblas_value<float> *s_alpha, mpcblas_value<float *> *&s_X, int incX);
void mpcblas_sscal(int N, std::optional<float> alpha, mpcblas_value<float *> *&s_X, int incX);
mpcblas_value<float *> *mpcblas_sscal(int N, mpcblas_value<float> *s_alpha, std::optional<float *> X, int incX);
```

Σχήμα 5.19: Παράδειγμα με χρήση της βιβλιοθήκης MPC-BLAS για τον υπολογισμό του εσωτερικού γινομένου δύο διανυσμάτων κινητής υποδιαστολής μονής ακρίβειας, όπου το κάθε διάνυσμα αποτελεί την ιδιωτική είσοδο του κάθε συμμετέχον.

```
void mpcblas_initialize(e_role role,
                const std::string &address,
                int port,
                int bitlen,
                seclvl security_level = LT,
                int n_threads = 2,
                e_mt_gen_alg mg_algo = MT_OT,
                int reserve_gates = 4000000,
                const std::string &aby_circ_dir = "/home/imslab/ABY/bin/circ/.");
void mpcblas_uninitialize();
void mpcblas_reset();
void mpcblas_set_party(mpcblas_party_t *context);
```

```
mpcblas_party_t *mpcblas_get_party();
BooleanCircuit *mpcblas_get_circuit();

template<typename... T>
class types_list;
template<typename... T>
class mpcblas_values;
template<typename... value_types, typename... sharings_types>
class mpcblas_values<types_list<value_types...>, types_list<sharings_types...>> {
 public:
   mpcblas_values(sharings_types ...shares1);
   mpcblas_values(value_types... values);
   std::tuple<sharings_types...> get_sharings();
   std::tuple<value_types...> get_values();
};

template<typename T>
class mpcblas_value {
 public:
   mpcblas_value(share *sharing);

   mpcblas_value(share *sharing, int nvals);

   mpcblas_value(T value);

   share *get_sharing();

   static mpcblas_value<T> *from_input(std::optional<T> input);

   static mpcblas_value<T> *from_input_ptr(int N, std::optional<T> input, int inc);

   T get_value();
};
```

Σχήμα 5.20: Παράδειγμα με χρήση της βιβλιοθήκης MPC-BLAS για τον υπολογισμό του εσωτερικού γινομένου δύο διανυσμάτων κινητής υποδιαστολής μονής ακρίβειας, όπου το κάθε διάνυσμα αποτελεί την ιδιωτική είσοδο του κάθε συμμετέχον.

5.3.2.1 Παραδείγματα χρήσης

Στην ενότητα αυτή παραθέτουμε δύο πλήρη παραδείγματα χρήσης της MPC-BLAS που αναλύσαμε παραπάνω. Στον Σχήμα 5.21 βλέπουμε ένα παράδειγμα υπολογισμού του εσωτερικού γινομένου δύο διανυσμάτων από δύο συμμετέχοντες (διεργασίες) οι οποίοι στην συνέχεια στέλνουν το αποτέλεσμα μέσω pipe στη γονεϊκή διεργασία για να συγκρίνει το αποτέλεσμα που βρήκαν με αυτό που επιστρέφει η χρήση της CBLAS διεπαφής.

```
#include <iostream>
#include <gtest/gtest.h>
#include "../../../core/mpc-blas.hpp"
#include "unistd.h"
#include "../../utils.hpp"
```

```
TEST(TESTxDOT, SIMPLE_PRODUCT) {
  float test_vector_1[4] = {1.21, 5.33, 3.345, 4.213};
  float test_vector_2[4] = {0.01, -3.82, -1023.905, 92302.43};

  uint32_t bitlen = 32;

  int pipe_fds_1[2];
  int pipe_fds_2[2];
  pipe(pipe_fds_1);
  pipe(pipe_fds_2);

  float result = 0, result_1 = 0, result_2 = 0;

  if (!fork()) {
    close(pipe_fds_1[0]);

    mpcblas_initialize(SERVER, "127.0.0.1", 7766, bitlen);
    result_1 = mpcblas_sdot(4, test_vector_1, 1, MPCBLAS_IGNORE, 1)->get_value();
    mpcblas_uninitialize();

    write(pipe_fds_1[1], (void *)&result_1, sizeof(result_1));
    exit(1);
  }

  if (!fork()) {
    close(pipe_fds_2[0]);

    mpcblas_initialize(CLIENT, "127.0.0.1", 7766, bitlen);
    result_2 = mpcblas_sdot(4, MPCBLAS_IGNORE, 1, test_vector_2, 1)->get_value();
    mpcblas_uninitialize();

    write(pipe_fds_2[1], (void *)&result_2, sizeof(result_2));
    exit(1);
  }

  result = cblas_sdot(4, test_vector_1, 1, test_vector_2, 1);

  close(pipe_fds_1[1]);
  close(pipe_fds_2[1]);

  int out_1 = read(pipe_fds_1[0], (void *)&result_1, sizeof(result_1));
  int out_2 = read(pipe_fds_2[0], (void *)&result_2, sizeof(result_2));

  if (out_1 == 0 || out_2 == 0) {
    std::cout << "EOF" << std::endl;
    EXIT_FAILURE;
  }

  if (out_1 == -1 || out_2 == -1) {
    std::cout << "ERROR" << std::endl;
    EXIT_FAILURE;
  }
  close(pipe_fds_1[0]);
  close(pipe_fds_2[0]);
```

```
    EXPECT_FLOAT_EQ(result_1, result_2);
    EXPECT_FLOAT_EQ(result, result_2);
    EXPECT_FLOAT_EQ(result, result_1);
}
```

Σχήμα 5.21: Παράδειγμα με χρήση της βιβλιοθήκης MPC-BLAS για τον υπολογισμό του εσωτερικού γινομένου δύο διανυσμάτων κινητής υποδιαστολής μονής ακρίβειας, όπου το κάθε διάνυσμα αποτελεί την ιδιωτική είσοδο του κάθε συμμετέχον.

Στο Σχήμα 5.22, το πρόγραμμα που εκτελείται μοιάζει πολύ με αυτό του Σχήματος 5.21 ωστόσο στην περίπτωση αυτή υπολογίζεται η αριθμητική παράσταση $\text{NORM}_2(\mathbf{x}\mathbf{y}^T\mathbf{x} + \mathbf{y})$.

```
#include "../utils.hpp"
#include <iostream>
#include <gtest/gtest.h>
#include "../../core/mpc-blas.hpp"
#include "unistd.h"

TEST(TESTIntegrations, TESTxNRM2xAXPYxDOT) {
    float test_vector_1[4] = {1.21, 5.33, 3.345, 4.213};
    float test_vector_2[4] = {0.01, 3.82, 1023.905, 92302.43};

    uint32_t bitlen = 32;

    int pipe_fds_1[2];
    int pipe_fds_2[2];
    pipe(pipe_fds_1);
    pipe(pipe_fds_2);

    float result, result_1, result_2;

    if (!fork()) {
        close(pipe_fds_1[0]);

        mpcblas_initialize(SERVER, "127.0.0.1", 7766, bitlen);
        auto y = mpcblas_saxpy(4,
                               mpcblas_sdot(4, test_vector_1, 1, MPCBLAS_IGNORE, 1),
                               test_vector_1, 1,
                               MPCBLAS_IGNORE, 1);
        result_1 = mpcblas_snrm2(4, y, 1)->get_value();
        mpcblas_uninitialize();

        write(pipe_fds_1[1], (void *)&result_1, sizeof(result_1));
        exit(1);
    }

    if (!fork()) {
        close(pipe_fds_2[0]);

        mpcblas_initialize(CLIENT, "127.0.0.1", 7766, bitlen);
        auto y = mpcblas_saxpy(4,
                               mpcblas_sdot(4, MPCBLAS_IGNORE, 1, test_vector_2, 1),
                               MPCBLAS_IGNORE, 1,
```

```
                    test_vector_2, 1);
    result_2 = mpcblas_snrm2(4, y, 1)->get_value();
    mpcblas_uninitialize();

    write(pipe_fds_2[1], (void *)&result_2, sizeof(result_2));
    exit(1);
  }

  cblas_saxpy(4,
          cblas_sdot(4, test_vector_1, 1, test_vector_2, 1),
          test_vector_1, 1,
          test_vector_2, 1);

  result = cblas_snrm2(4, test_vector_2, 1);

  int out_1 = read(pipe_fds_1[0], (void *)&result_1, sizeof(result_1));
  int out_2 = read(pipe_fds_2[0], (void *)&result_2, sizeof(result_2));

  if (out_1 == 0 || out_2 == 0) {
    std::cout << "EOF" << std::endl;
    EXIT_FAILURE;
  }

  if (out_1 == -1 || out_2 == -1) {
    std::cout << "ERROR" << std::endl;
    EXIT_FAILURE;
  }
  close(pipe_fds_1[0]);
  close(pipe_fds_2[0]);

  EXPECT_FLOAT_EQ(result_1, result_2);
  EXPECT_FLOAT_EQ(result, result_2);
  EXPECT_FLOAT_EQ(result, result_1);
}
```

Σχήμα 5.22: Παράδειγμα με χρήση της βιβλιοθήκης MPC-BLAS για τον υπολογισμό του εσωτερικού γινομένου δύο διανυσμάτων κινητής υποδιαστολής μονής ακρίβειας, όπου το κάθε διάνυσμα αποτελεί την ιδιωτική είσοδο του κάθε συμμετέχον. Στο συγκεκριμένο παράδειγμα εφαρμόζονται πολλαπλές πράξεις

5.3.3 Εσωτερική δομή

Στην τελευταία αυτή ενότητα θα περιγράψουμε την εσωτερική δομή και την οργάνωση του πηγαίου κώδικα της βιβλιοθήκης MPC-BLAS. Η εσωτερική οργάνωση του παρατηρείται στο Σχήμα 5.23.

```
.
|
|--- src
|      |--- core
|      |      |--- CMakeLists.txt
```

```
|   |   |--- level-1
|   |   |   |--- IxAMAX.cpp
|   |   |   |--- xASUM.cpp
|   |   |   |--- xAXPY.cpp
|   |   |   |--- xCOPY.cpp
|   |   |   |--- xDOT.cpp
|   |   |   |--- xNRM2.cpp
|   |   |   |--- xROT.cpp
|   |   |   |--- xROTG.cpp
|   |   |   |--- xROTM.cpp
|   |   |   |--- xROTMG.cpp
|   |   |   |--- xSCAL.cpp
|   |   |   |--- xSWAP.cpp
|   |   |--- mpc-blas.cpp
|   |   |--- mpc-blas.hpp
|   |   |--- utils
|   |   |   |--- utils.cpp
|   |   |   |--- utils.hpp
|   |--- tests
|   |   |--- CMakeLists.txt
|   |   |--- integrations
|   |   |   |--- TESTintegrations.cpp
|   |   |--- main.cpp
|   |   |--- units
|   |   |   |--- level-1
|   |   |   |   |--- TESTIxAMAX.cpp
|   |   |   |   |--- TESTxASUM.cpp
|   |   |   |   |--- TESTxAXPY.cpp
|   |   |   |   |--- TESTxCOPY.cpp
|   |   |   |   |--- TESTxDOT.cpp
|   |   |   |   |--- TESTxNRM2.cpp
|   |   |   |   |--- TESTxROT.cpp
|   |   |   |   |--- TESTxROTG.cpp
|   |   |   |   |--- TESTxROTM.cpp
|   |   |   |   |--- TESTxROTMG.cpp
|   |   |   |   |--- TESTxSCAL.cpp
|   |   |   |   |--- TESTxSWAP.cpp
|   |   |--- utils
|   |   |   |--- TESTutils.cpp
|   |   |--- utils.hpp
|--- CMakeLists.txt
```

Σχήμα 5.23: Η δομή του πηγαίου κώδικα της MPC-BLAS.

Στο σχήμα αυτό, το πρώτο επίπεδο παρατηρούμε πως υπάρχει ο φάκελος src και το αρχείο CMake ολόκληρου του πρότζεκτ. Εντός του φακέλους src υπάρχει ο πηγαίος κώδικας της βιβλιοθήκης, στον φάκελο core, και όλες οι περιπτώσεις ελέγχου (test cases) του κώδικα, στον φάκελο tests. Εντός του φακέλου core βρίσκεται ο πηγαίος κώδικας για κάθε συνάρτηση που υλοποιείται σε αρχείο με όνομα αντίστοιχο αυτού της συνάρτησης που υλοποιεί. Επίσης, υπάρχει το κεντρικό αρχείο της διεπαφής της MPC-BLAS, το mpc-blas.hpp, καθώς και το αρχείο CMakeLists.txt που χρησιμοποιείται για να χτιστεί η βιβλιοθήκη. Στον φάκελο tests υπάρχει ο κώδικας ελέγχου όλων των συναρτήσεων της διεπαφής καθώς και το αρχείο

CMakeLists.txt το οποίο χρησιμοποιείται για να χτιστεί το εκτελέσιμο που τρέχει των κώδικα ελέγχου της βιβλιοθήκης.

5.4 Πειράματα

Στην ενότητα αυτή θα διενεργήσουμε κάποια μικρά πειράματα κυρίως συγκρίνοντας την PC-BLAS και τη CBLAS ως προς τους χρόνους εκτέλεσης τους.

5.4.1 Σύγκριση χρόνων εκτέλεσης για σταθερό μέγεθος εισόδου

Αν και πρόκειται για μια άνιση σύγκριση συγκρίναμε τους χρόνους εκτέλεσης της, για όλες τις διαδικασίες ελέγχου, με τους αντίστοιχους χρόνους εκτέλεσης μιας BLAS βιβλιοθήκης που τρέχει τοπικά, την OpenBLAS [57]. Όλοι οι χρόνοι μετρήθηκαν στο ίδιο σύστημα και παρουσιάζονται στο Σχήμα 5.24. Παρατηρούμε πως για τις περισσότερες πράξεις η βιβλιοθήκη μας είναι περίπου της τάξης των 10.000-100.000 φορών πιο αργή από την OpenBLAS. Αυτό το θεωρούμε αναμενόμενο, αν λάβει κανείς υπόψιν ότι η υλοποίηση μας χρησιμοποιεί Boolean κυκλώματα για κάθε πύλη πολλαπλασιασμού κινητής υποδιαστολής, οι οποίες αποτελούν ιδιαίτερα μεγάλα σε μήκος κυκλώματα.

```
CBLAS ISAMAX :  duration : 0.69653ms
MPC-BLAS ISAMAX :  duration : 75.406ms

CBLAS SASUM :  duration : 0.002848ms
MPC-BLAS SASUM :  duration : 56.703ms

CBLAS SAXPY :  duration : 0.975238ms
MPC-BLAS SAXPY :  duration : 22.27ms

CBLAS SCOPY :  duration : 0.001115ms
MPC-BLAS SCOPY :  duration : 0.406ms

CBLAS SDOT :  duration : 0.001172ms
MPC-BLAS SDOT :  duration : 45.429ms

CBLAS SNRM2 :  duration : 0.00223ms
MPC-BLAS SNRM2 :  duration : 53.697ms

CBLAS SSCAL :  duration : 0.002149ms
MPC-BLAS SSCAL :  duration : 11.299ms

CBLAS SSWAP : 0.001115ms
MPC-BLAS SSWAP : 0.573ms

CBLAS SAXPY SDOT :  duration : 0.00236ms
MPC-BLAS SAXPY SDOT :  duration : 60.47ms
```

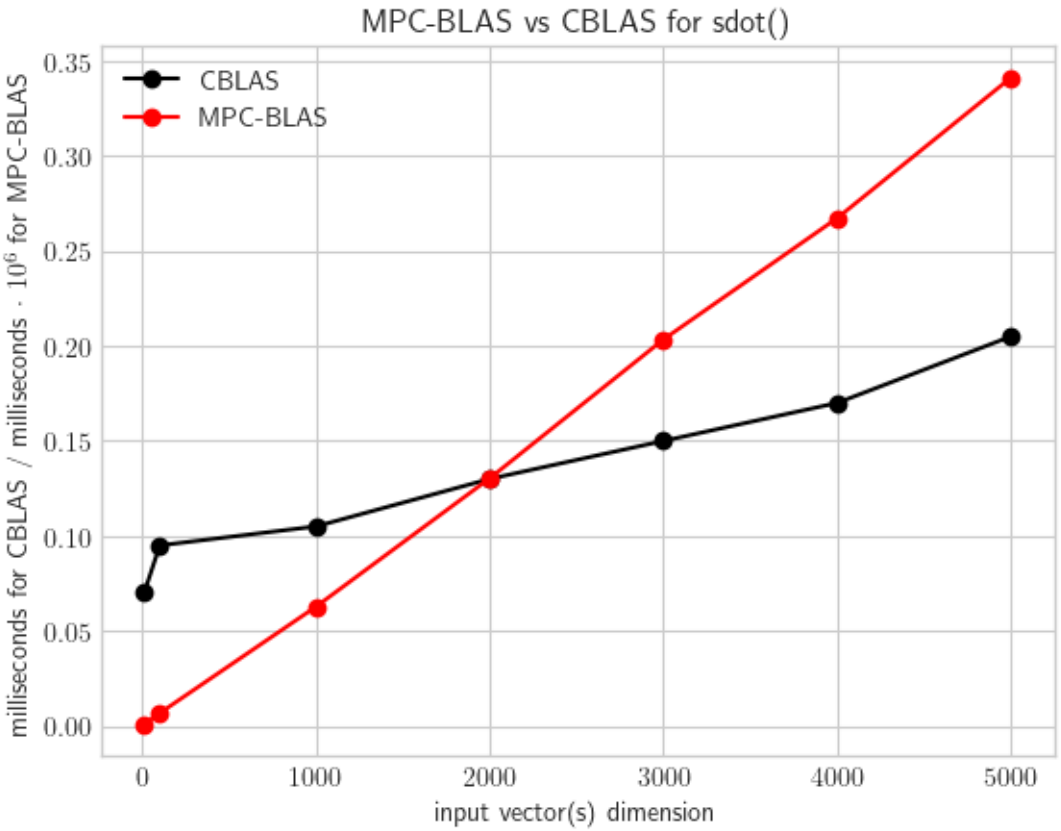
```
CBLAS SNRM2 SXAPY SDOT :  duration : 0.001205ms
MPC-BLAS SNRM2 SXAPY SDOT :  duration : 64.788ms
```

Σχήμα 5.24: Οι ενδιάμεσες τιμές του χρόνου εκτέλεσης επεξεργαστή (CPU time) της κάθε συνάρτησης της MPC-BLAS και τις αντίστοιχης OpenBLAS. Η MPC-BLAS εκτελείται τοπικά σε δύο διαφορετικές διεργασίες, όπου η κάθε διεργασία έτρεχε σε 5 νήματα. Όλες οι τιμές για την MPC-BLAS λήφθηκαν μέσω της εντολής *## define PRINT_PERFORMANCE_STATS* της βιβλιοθήκης ABY ενώ για την CBLAS μέσω του χρονομέτρου std::chrono της C++.

5.4.2 Σύγκριση χρόνων εκτέλεσης για μεταβαλλόμενο μέγεθος εισόδου

Ένα ακόμα ενδιαφέρον χαρακτηριστικό που θα προσπαθήσουμε να εξετάσουμε αν ισχύει πως η θεωρητική χρονική ασυμπτωτική πολυπλοκότητα των βιβλιοθήκων MPC-BLAS και CBLAS ταυτίζεται. Προσπαθήσαμε να παρατηρήσουμε το παραπάνω κάνοντας ένα πείραμα για μια συγκεκριμένη BLAS συνάρτηση, τη SDOT, την οποία τρέξαμε για συνεχώς αυξανόμενο μέγεθος εισόδου τόσο στη CBLAS όσο και στην MPC-BLAS. Τα αποτελέσματα φαίνονται στο Σχήμα 5.25. Δυστυχώς, λόγω του περιορισμένου υλικού που είχαμε στη διάθεση μας δεν μπορέσαμε να πραγματοποιήσουμε εκτελέσεις για μεγαλύτερο μέγεθος εισόδου. Παρατηρούμε πως και οι δύο βιβλιοθήκες έχουν χρόνο εκτέλεσης γραμμικό στο μέγεθος της εισόδου κάτι που επαληθεύει τον ισχυρισμό μας. Το αποτέλεσμα αυτό αν το ερμηνεύσουμε από μια θεωρητική αφαιρετική σκοπιά, χωρίς να μπούμε σε ιδιαίτερες μαθηματικές λεπτομέρειες, μας φαίνεται απόλυτα λογικό. Η περίπτωση της CBLAS είναι πιο απλή, αφού είναι σχεδόν προφανές ότι θα έχει γραμμική χρονική πολυπλοκότητα, υποθέτοντας μια αποδοτική υλοποίηση της, αφού εκτελεί απλώς πολλαπλασιασμούς και προσθέσεις. Η περίπτωση της MPC-BLAS είναι λίγο πιο πολύπλοκη αφού αποτελείται από πολλά συστατικά μέρη. Αν υποθέσουμε ότι η βιβλιοθήκη ABY έχει υλοποιηθεί αποδοτικά μη συμπεριλαμβανομένων των κρυπτογραφικών αλγορίθμων. Τότε το μόνο που μένει είναι να εξετάσουμε τη χρονική πολυπλοκότητα των κρυπτογραφικών αλγορίθμων που χρησιμοποιεί. Τα δύο κρυπτογραφικά μέρη είναι οι Ανυποψίαστες Μεταφορές και το πρωτόκολλο GMW. Οι πρώτες, παρότι είναι αργές, έχουν γραμμική ασυμπτωτική χρονική πολυπλοκότητα κάτι που ένας αναγνώστης μπορεί να αντιληφθεί αναλύοντας έναν βασικό αλγόριθμο όπως αυτόν του Κεφαλαίου 2 και το GMW επίσης έχει γραμμική ασυμπτωτική χρονική πολυπλοκότητα. Συμπεραίνουμε λοιπόν πως η διαφορά των δύο χρονικών ασυμπτωτικών πολυπλοκοτήτων είναι σταθερές τιμές, δηλαδή το όφσετ και η κλίση μιας ευθείας.

Ας κοιτάξουμε ξανά το παραπάνω αποτέλεσμα από μια πιο πρακτική ματιά. Οι χρονικές πολυπλοκότητες του Σχήματος 5.25 είναι άμεσα συσχετισμένες με το πλήθος των πυλών που χρησιμοποιούνται από αυτές τις δύο βιβλιοθήκες σε επίπεδο υλικού. Αν θεωρήσομε το



Σχήμα 5.25: Οι ενδιάμεσες τιμές του χρόνου εκτέλεσης επεξεργαστή (CPU time) της συνάρτησης SDOT για τις βιβλιοθήκες MPC-BLAS και OpenBLAS για συνεχώς μεγαλύτερη είσοδο. Η MPC-BLAS εκτελείται τοπικά σε δύο διαφορετικές διεργασίες, όπου η κάθε διεργασία έτρεχε σε 5 νήματα. Όλες οι τιμές για την MPC-BLAS λήφθηκαν μέσω της εντολής *## define PRINT_PERFORMANCE_STATS* της βιβλιοθήκης ABY ενώ για την CBLAS μέσω του χρονομέτρου std::chrono της C++.

πρωτόκολλο BGW ως τον "εικονικό" επεξεργαστή της MPC-BLAS πράξης, τότε μπορούμε να ισχυριστούμε πως η εκτέλεση ενός αλγορίθμου σε αυτόν τον "εικονικό" επεξεργαστή δεν επηρεάζει την ασυμπτωτική χρονική πολυπλοκότητα του παρά μόνο εισάγει μια γραμμική καθυστέρηση που εξαρτάται από το μέγεθος του αλγορίθμου αυτού. Δηλαδή, πως σε επίπεδο πραγματικών πυλών υλικού, εισάγει μια γραμμική στο μέγεθος της εισόδου επιβάρυνση. Ωστόσο, αυτός ο ισχυρισμός δεν είναι ιδιαίτερα ισχυρός καθώς εξαρτάται άμεσα από την πλατφόρμα στην οποία τρέχει αυτός ο εικονικός επεξεργαστής, η οποία συμπεριλαμβάνει το λειτουργικό σύστημα, το υλικό, το δίκτυο και αρκετούς ακόμα άγνωστους παράγοντες. Ο ισχυρισμός αυτός ισχύει μόνο στην περίπτωση που θεωρήσουμε πως η πλατφόρμα στην οποία εκτελείται ο εικονικός επεξεργαστής εισάγει κατά μέσο όρο γραμμική επιβάρυνση στον αριθμό των πυλών που χρησιμοποιεί ένα αυθαίρετο πρόγραμμα.

Κεφάλαιο 6

Επίλογος

Σε αυτό το σημείο, γίνεται μία σύντομη σύνοψη των προηγούμενων κεφαλαίων. Δίνεται περισσότερη έμφαση κυρίως στο κεφαλαίο της υλοποίησης, που είναι και ουσιαστικά η κύρια πρόταση αυτής της εργασίας, πέραν της μελέτης των SMPC πρωτοκόλλων και τις ανάλυσης όλων των προαπαιτούμενων γνώσεων για την κατανόηση αυτών.

6.1 Αποτελέσματα και Συμπεράσματα

Στο θεωρητικό κομμάτι αυτής της εργασίας έγινε μια εισαγωγή στην κρυπτογραφία, στη θεωρητική ασφάλεια καθώς και σε μεθόδους απόδειξης της, με απότερο σκοπό την ανάλυση πρωτοκόλλων Ασφαλούς Υπολογισμού Πολλών Μερών (SMPC). Στα πλαίσια της κρυπτογραφίας αναλύθηκαν σύγχρονα εργαλεία όπως Σχήματα Διαμοίρασης Μυστικών, Σχήματα Δέσμευσης, Σχήματα Ανυποψίαστης και Αποδείξεις Μηδενικής Γνώστης. Παράλληλα με αυτά αναλύθηκε όλο το μαθηματικό υπόβαθρο που απαιτείται για τη βαθύτερη κατανόηση τους. Σχετικά με τη θεωρητική ασφάλεια, αναλύθηκαν οι βασικοί ορισμοί και οι βασικές μεθόδοι απόδειξης της, η Μέθοδος Μεταπήδησης μεταξύ Παιχνιδιών και η Μέθοδος Προσομοίωσης. Αφού καλύφθηκε όλο το απαραίτητο υπόβαθρο, αναλύθηκαν τέσσερα πρωτόκολλα SMPC, το Πρωτόκολλο Μπερδεμένων Δικτύων Yao, το BMR, το GMW και το SPDZ. Τα πρώτα τρία από αυτά αποτελούν κλασσικά πρωτόκολλα τα οποία ήταν υποψήφια για να χρησιμοποιηθούν στην υλοποίηση, ενώ το τελευταίο είναι από τα πιο σύγχρονα και επιτυχημένα πρωτόκολλα το οποίο είναι ασφαλές ενάντια σε ενεργητικούς αντιπάλους.

Στο πρακτικό κομμάτι αυτής της εργασίας υλοποιήθηκε, και περιγράφηκε στο Κεφάλαιο 5, η βιβλιοθήκη MPC-BLAS, μια 2MPC BLAS Level-1 βιβλιοθήκη για πραγματικούς αριθμούς κινητής υποδιαστολής μονής ακρίβειας για δύο συμμετέχοντες. Η βιβλιοθήκη αυτή είναι η πρώτη βιβλιοθήκη αυτού του είδους, που προτείνεται στη βιβλιογραφία, τη στιγμή που γράφεται αυτή η εργασία.

Στα πλαίσια της υλοποίησης εκτελέστηκαν επίσης δύο πειράματα. Προσπαθήσαμε να συγκρίνουμε το ποσοστό της επιβάρυνσης που εισάγει μια SMPC υλοποίηση για τους αλγορίθμους που υλοποιήσαμε. Παρατηρήσαμε ότι η επιβάρυνση που εισάγεται είναι γραμμική και είναι της τάξης των 10.000-100.000 χρονικών μονάδων. Δηλαδή, αν μια εκτέλεση κάποιας συνάρτησης μια βιβλιοθήκης CBLAS απαιτούσε 1 δευτερόλεπτο, η αντίστοιχη κλήση στην περίπτωση της MPC-BLAS θα απαιτούσε 10.000-100.000 δευτερόλεπτα. Αυτό ίσως ακούγεται ιδιαίτερα αποθαρρυντικό για μια τυπική/εμπορική χρήση αυτής της βιβλιοθήκης. Ωστόσο, εκτιμούμε πως υπάρχει τεράστιο περιθώριο βελτίωσης, ίσως και κατά αρκετές τάξεις μεγέθους και αυτό αποδεικνύεται από το γεγονός πως τα πρωτόκολλα SMPC έχουν σήμερα παραδείγματα εμπορικών εφαρμογών και μάλιστα για μεγάλους όγκους δεδομένων. Να σημειώσουμε βέβαια πως περισσότερα από τα παραδείγματα αυτά επικεντρώνονται κυρίως σε σχετικά μικρό αριθμό συμμετεχόντων, συνήθως μικρότερο των 10. Πολλές ιδέες για τη βελτίωση της απόδοσης της βιβλιοθήκης αναλύονται στην επόμενη ενότητα. Τέλος, θεωρούμε πως μια βελτιωμένη, ως προς την ταχύτητα, μορφή MPC-BLAS μπορεί να αποκτήσει πρακτική εφαρμογή. Ο ισχυρισμός αυτός βασίζεται στο γεγονός ότι υπάρχουν πάρα πολλά προγράμματα και βιβλιοθήκες που χρησιμοποιούν BLAS πράξεις, όπως το MATLAB και η NumPy στις οποίες βασίζονται εφαρμογές που θα επιθυμούσαν την ιδιότητα της ιδιωτικότητας.

6.2 Ιδέες και θέματα μελλοντικής μελέτης

Στην Ενότητα αυτή θα παρουσιάσουμε ορισμένες ιδέες για βελτίωση της βιβλιοθήκης MPC-BLAS που υλοποιήσαμε. Στις περισσότερες περιπτώσεις η βιβλιοθήκη ABY στην οποία βασίζεται η MPC-BLAS αποτελεί τον περιοριστικό παράγοντα και άρα μια βελτίωση ή προσθήκη στη βιβλιοθήκη ABY συνεπάγεται με μικρές αλλαγές βελτίωση της MPC-BLAS.

6.2.1 Υλοποίηση των BLAS-2 και BLAS-3 υπορουτινών και ολοκλήρωση μιας εύχρηστης βιβλιοθήκης

Η υλοποίηση των λειτουργιών BLAS-2 και BLAS-3 ήταν εξ' αρχής εκτός των σκοπών αυτής της εργασίας, αφού κύριο μέλημα της ήταν η μελέτη του κλάδου του SMPC ξεκινώντας από τα πρώτα του βήματα και φτάνοντας ως τη σύγχρονη βιβλιογραφία, τόσο τη θεωρητική όσο και την υλοποιητική και τέλος η προσπάθεια πειραματισμού και εφαρμογής αυτών σε μια καινοτόμα υλοποίηση. Η υλοποίηση σε καμία περίπτωση δεν είναι πλήρης και δεν υπήρχε ο χρόνος για να συμβεί αυτό. Έτσι, η υλοποίηση των BLAS-2 και BLAS-3 υπορουτινών της βιβλιοθήκης είναι το σημαντικότερο κομμάτι του παζλ που λείπει αυτή τη στιγμή από την υλοποίηση για να ολοκληρωθεί και να εξασφαλίσει έτσι πιθανότητες να χρησιμοποιηθεί και σε άλλα πειράματα στη βιβλιογραφία. Σαν επόμενο βήμα από αυτό θα ήταν η δημιουργία

διεπαφής/δεσιμάτων (bindings) και σε άλλες γλώσσες προγραμματισμού ή βιβλιοθήκες προκειμένου να γίνει πιο εύχρηστη. Ας μην ξεχνάμε άλλωστε, ότι τα υλοποιητικά παράγωγα της κρυπτογραφίας, στην οποιαδήποτε μορφή τους, καλούνται να παίξουν τον ρόλο του διαφανούς ενδιάμεσου σε έναν χρήστη ή έναν προγραμματιστή, ρόλος που πιστεύουμε ότι ως επί το πλείστον αποτυγχάνουν σε σημαντικό βαθμό να παίξουν και δεν είναι καθόλου εύκολο να επιτύχεις.

6.2.2 Υλοποίηση αποδοτικότερου αλγορίθμου Κινητής Υποδιαστολής

Όπως προαναφέραμε η βιβλιοθήκη ABY υποστηρίζει πράξεις κινητής υποδιαστολής μόνο μέσω Boolean κυκλώματος, το οποίο είναι μια υλοποίηση μιας Μονάδας Κινητής Υποδιαστολής σε γλώσσα περιγραφής υλικού (π.χ. Verilog, VHDL κτλ.) η οποία στη συνέχεια έχει μεταφραστεί και μετατραπεί σε κατάλληλη μορφή ώστε να μπορεί να την διαβάσει το ABY με σκοπό να δημιουργήσει το αντίστοιχο Μπερδεμένο Δίκτυο Yao. Είναι προφανές ότι παρά τις τεχνικές επιτάχυνσης των δικτύων αυτών, η πολυπλοκότητα επικοινωνίας τους δεν μπορεί να συγκριθεί με αυτή αντίστοιχων ομομορφικών πρωτοκόλλων που υποστηρίζουν αυτές τις πράξεις [58] [59]. Αυτό κυρίως οφείλεται στο ότι η πολυπλοκότητα επικοινωνίας τους εξαρτάται από το βάθος του κυκλώματος που χρησιμοποιούν για να αναπαραστήσουν τον υπολογισμό, όπου στην παρούσα περίπτωση είναι αρκετά μεγάλο. Είναι σημαντικό να αναφέρουμε, ότι υποστήριξη πράξεων κινητής υποδιαστολής που συμμορφώνονται σε συγκεκριμένα πρότυπα, όπως για παράδειγμα το IEEE754 **8766229**, δεν είναι μια ιδιαίτερα απλή αλγοριθμική και υλοποιητική διαδικασία [60] η οποία γίνεται ακόμα δυσκολότερη όταν θέλουμε να υποστηρίξουμε και συναρτήσεις όπως η εύρεση ρίζας ($SQRT$, που χρειάζεται για παράδειγμα στον υπολογισμό του μέτρου ενός διανύσματος στην περίπτωση του BLAS, ή το ημίτονο (SIN) που χρησιμοποιείται σε έναν πίνακα περιστροφής. Η διαδικασία γίνεται ακόμα πιο δυσμενής αν αναλογιστούμε τους παράγοντες της ασφάλειας και της αποδοτικότητας που εισάγει ο MPC. Μόλις το 2021 αναπτύχθηκε η πρώτη σουίτα αλγορίθμων που υποστηρίζουν πράξεις κινητής υποδιαστολής καθώς και ένα πλήθος από συναρτήσεις όπως αυτή του ημίτονου και της ρίζας που αναφέραμε, στην εργασία [58] με όνομα SECFLOAT. Η SECFLOAT βασίζεται στον διαμοιρασμό μυστικών και υποστηρίζει πράξεις κινητής υποδιαστολής μέχρι 32-bit με συμμόρφωση στο πρότυπο IEE754. Η υλοποίηση αυτής της σουίτας αλγορίθμων θα αύξανε δραματικά την απόδοση του συστήματος μας αφού οι BLAS συναρτήσεις χρησιμοποιούν αποκλειστικά πράξεις κινητής υποδιαστολής. Πιο συγκεκριμένα, οι συγγραφείς της SECFLOAT την σύγκριναν με την απόδοση του ABY σε πράξεις κινητής υποδιαστολής και παρατήρησαν ότι είναι 5-11 φορές γρηγορότερη ανάλογα με την πράξη που επιλέγεται. Ο μοναδικός περιοριστικός παράγοντάς είναι ότι η SECFLOAT έτσι όπως έχει προταθεί στη βιβλιογραφία υποστηρίζει πράξεις μόνο μέχρι 32-bit κάτι που θα επέτρεπε μόνο περίπου το 1/4

μιας βιβλιοθήκης BLAS να βασιστεί σε αυτήν καθώς στα περισσότερα συστήματα τα 32-bit αντιστοιχούν στον τύπου float. Εκτιμούμε όμως ότι οι αλγόριθμοι της SECFLOAT με μικρές αλλαγές μπορούν να υποστηρίξουν πράξεις κινητής υποδιαστολής μέχρι και 64-bit.

6.2.3 Επέκταση της ABY για SMPC ή χρήση βιβλιοθήκης για υποστήριξη SMPC

Η βιβλιοθήκη ABY υποστηρίζει μόνο υπολογισμού για πλαίσια του 2PC. Αυτό ίσως αποτελεί έναν περιοριστικό παράγοντα για τη βιβλιοθήκη μας. Θα μπορούσαμε να εξετάσουμε να την βασίσουμε σε κάποια άλλη βιβλιοθήκη που να υποστηρίζει εξ'αρχής SMPC πρωτόκολλα. Όπως αναφέρουμε και στο Κεφάλαιο 4, πριν την υλοποίηση η προσπάθεια μας για εύρεση κατάλληλης βιβλιοθήκης για SMPC απέβη άκαρπη. Στην υλοποίηση αυτής της ιδέας, παρότι θεωρούμε ότι θα είχε πολύ θετικό αντίκτυπο, κρίνουμε πως είναι ιδιαίτερα δύσκολη διότι είτε θα πρέπει να επεκτείνουμε τη βιβλιοθήκη ABY για υποστήριξη SMPC είτε να χρησιμοποιήσουμε κάποια δύσχρηστη βιβλιοθήκη που να υποστηρίζει υπολογισμούς σε αυτό το πλαίσιο όπως η MP-SPDZ [53].

6.2.4 Επέκταση της ABY ή χρήση βιβλιοθήκης με υποστήριξη πρωτοκόλλων ασφαλών ενάντια σε Ενεργητικούς Αντιπάλους

Η βιβλιοθήκη ABY δεν υποστηρίζει πρωτόκολλα ασφαλή απέναντι σε ενεργητικούς αντιπάλους, παράπονο ενάντια σε παθητικούς αντιπάλους και άρα η παρούσα έκδοση της MPCBLAS δεν υποστηρίζει αυτού του είδους πρωτόκολλα. Στο Κεφάλαιο 4 μελετήθηκε το πρωτόκολλο SPDZ το οποίο, ως και σήμερα, με παραλλαγές του, αποτελεί από τα πιο αποδοτικά πρωτόκολλα ασφαλή ενάντια σε ενεργητικούς αντιπάλους που έχουν προταθεί στη βιβλιογραφία. Θα μπορούσαμε να επεκτείνουμε τη βιβλιοθήκη ABY στο να υλοποιεί το SDPZ και στη συνέχεια να το χρησιμοποιήσουμε στη βιβλιοθήκη MPCBLAS. Η συγκεκριμένη πρόταση γίνεται διότι, παρότι υπάρχουν ορισμένες υλοποιήσεις το SPDZ όπως οι [53], [61], [62], πολλές από αυτές δε διαθέτουν ιδιαίτερα εύχρηστη διεπαφή για C/C++, όπως η [61], δεν διαθέτουν κατάλληλη τεκμηρίωση της διεπαφής, όπως η [53], ή δε διαθέτουν διεπαφή για C/C++, όπως η [62]. Ωστόσο, στην περίπτωση που θέλουμε να επιτύχουμε και υψηλή απόδοση στη βιβλιοθήκη θα πρέπει να διερευνηθεί το κατά πόσο μπορεί η σουίτα αλγορίθμων SECFLOAT να υλοποιηθεί με βάση το SPDZ.

Κεφάλαιο 7

Βιβλιογραφία

[1] T. Kelly, «The myth of the skytale», *Cryptologia*, τόμ. 22, αρθμ. 3, σσ. 244–260, 1998. DOI: 10.1080/0161-119891886902. eprint: https://doi.org/10.1080/0161-119891886902. διεύθν.: https://doi.org/10.1080/0161-119891886902.

[2] J. Holden, *The Mathematics of Secrets: Cryptography from Caesar Ciphers to Digital Encryption*. 2019.

[3] L. Pearce, «A whirlwind history of cryptography», Οκτ. 2020. DOI: 10.2172/1671059. διεύθν.: https://www.osti.gov/biblio/1671059.

[4] F. L. Bauer, «Cryptanalysis», στο *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg και S. Jajodia, επιμελητές. Boston, MA: Springer US, 2011, σσ. 277–281, ISBN: 978-1-4419-5906-5. DOI: 10.1007/978-1-4419-5906-5_164. διεύθν.: https://doi.org/10.1007/978-1-4419-5906-5_164.

[5] A. Kerckhoffs, *La cryptographie militaire, ou, Des chiffres usités en temps de guerre: avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. Librairie militaire de L. Baudoin, 1883.

[6] C. E. Shannon, «A mathematical theory of cryptography», *Mathematical Theory of Cryptography*, 1945.

[7] W. Diffie και M. Hellman, «New directions in cryptography», *IEEE Trans. Inf. Theor.*, τόμ. 22, αρθμ. 6, σσ. 644–654, Σεπτ. 2006, ISSN: 0018-9448. DOI: 10.1109/TIT.1976.1055638. διεύθν.: https://doi.org/10.1109/TIT.1976.1055638.

[8] R. L. Rivest, A. Shamir και L. Adleman, «A method for obtaining digital signatures and public-key cryptosystems», *Commun. ACM*, τόμ. 21, αρθμ. 2, σσ. 120–126, Φεβ. 1978, ISSN: 0001-0782. DOI: 10.1145/359340.359342. διεύθν.: https://doi.org/10.1145/359340.359342.

[9] R. Drath και A. Horch, «Industrie 4.0: Hit or hype?[industry forum]», *IEEE industrial electronics magazine*, τόμ. 8, αρθμ. 2, σσ. 56–58, 2014.

[10] M. Skilton και F. Hovsepian, *The 4th industrial revolution*. Springer, 2018.

[11] A. Aljabre, «Cloud computing for increased business value», *International Journal of Business and social science*, τόμ. 3, αρθμ. 1, 2012.

[12] L. Golightly, V. Chang, Q. A. Xu, X. Gao και B. S. Liu, «Adoption of cloud computing as innovation in the organization», *International Journal of Engineering Business Management*, τόμ. 14, σ. 18 479 790 221 093 992, 2022.

[13] T. Chen, T.-T. Chuang και K. Nakatani, «The perceived business benefit of cloud computing: An exploratory study», *Journal of International Technology and Information Management*, τόμ. 25, αρθμ. 4, σ. 7, 2016.

[14] N. Gershenfeld, R. Krikorian και D. Cohen, «The internet of things», *Scientific American*, τόμ. 291, αρθμ. 4, σσ. 76–81, 2004.

[15] S. Li, L. D. Xu και S. Zhao, «The internet of things: A survey», *Information systems frontiers*, τόμ. 17, αρθμ. 2, σσ. 243–259, 2015.

[16] G. Ateniese, R. Burns, R. Curtmola κ.ά., «Provable data possession at untrusted stores», στο *Proceedings of the 14th ACM Conference on Computer and Communications Security*, σειρά CCS '07, Alexandria, Virginia, USA: Association for Computing Machinery, 2007, σσ. 598–609, ISBN: 9781595937032. DOI: 10.1145/1315245.1315318. διεύθν.: https://doi.org/10.1145/1315245.1315318.

[17] I. Gupta, A. K. Singh, C. Lee και R. Buyya, «Secure data storage and sharing techniques for data protection in cloud environments: A systematic review, analysis, and future directions», *IEEE Access*, τόμ. 10, σσ. 71 247–71 277, 2022. DOI: 10.1109/ACCESS.2022.3188110. διεύθν.: https://doi.org/10.1109/ACCESS.2022.3188110.

[18] C. Gentry, «Fully homomorphic encryption using ideal lattices», στο *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, σειρά STOC '09, Bethesda, MD, USA: Association for Computing Machinery, 2009, σσ. 169–178, ISBN: 9781605585062. DOI: 10.1145/1536414.1536440. διεύθν.: https://doi.org/10.1145/1536414.1536440.

[19] C. Gentry και S. Halevi, «Implementing gentry's fully-homomorphic encryption scheme», στο *Advances in Cryptology -- EUROCRYPT 2011*, K. G. Paterson, επιμελητής, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, σσ. 129–148, ISBN: 978-3-642-20465-4.

[20] A. Acar, H. Aksu, A. S. Uluagac και M. Conti, «A survey on homomorphic encryption schemes: Theory and implementation», *ACM Computing Surveys (Csur)*, τόμ. 51, αρθμ. 4, σσ. 1–35, 2018.

[21] A. C.-C. Yao, «How to generate and exchange secrets», στο *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, 1986, σσ. 162–167. DOI: 10.1109/SFCS.1986.25.

[22] A. C. Yao, «Protocols for secure computations», στο *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, 1982, σσ. 160–164. DOI: 10.1109/SFCS.1982.38.

[23] D. Boneh και V. Shoup, «A graduate course in applied cryptography», *Draft 0.5*, 2020.

[24] J. Hoffstein, J. Pipher, J. H. Silverman και J. H. Silverman, *An introduction to mathematical cryptography*. Springer, 2008, τόμ. 1.

[25] A. Pagourtzis και E. Zachos, «Computational cryptography», 2016.

[26] F. Miller, *Telegraphic code to insure privacy and secrecy in the transmission of telegrams*. CM Cornwell, 1882.

[27] E. Rescorla, «The Transport Layer Security (TLS) Protocol Version 1.3», RFC Editor, RFC 8446, Αύγ. 2018. διεύθν.: https://www.rfc-editor.org/rfc/rfc8446.txt.

[28] O. Goldreich, *A short tutorial of zero-knowledge*. 2013.

[29] M. Ben-Or, O. Goldreich, S. Goldwasser κ.ά., «Everything provable is provable in zero-knowledge», στο *Conference on the Theory and Application of Cryptography*, Springer, 1988, σσ. 37–56.

[30] G. Couteau, «Zero-knowledge proofs for secure computation», Διδακτορική διατρ., Université Paris sciences et lettres, 2017.

[31] R. Cramer, I. Damgård και B. Schoenmakers, «Proofs of partial knowledge and simplified design of witness hiding protocols», στο *Annual International Cryptology Conference*, Springer, 1994, σσ. 174–187.

[32] D. Evans, V. Kolesnikov και M. Rosulek, «A pragmatic introduction to secure multi-party computation», *Found. Trends Priv. Secur.*, τόμ. 2, αρθμ. 2-3, σσ. 70–246, Δεκ. 2018, ISSN: 2474-1558. DOI: 10.1561/3300000019. διεύθν.: https://doi.org/10.1561/3300000019.

[33] M. O. Rabin, *How to exchange secrets with oblivious transfer*, Cryptology ePrint Archive, Paper 2005/187, https://eprint.iacr.org/2005/187, 2005. διεύθν.: https://eprint.iacr.org/2005/187.

[34] S. Even, O. Goldreich και A. Lempel, «A randomized protocol for signing contracts», *Communications of the ACM*, τόμ. 28, αρθμ. 6, σσ. 637–647, 1985.

[35] A. Shamir, «How to share a secret», *Commun. ACM*, τόμ. 22, αρθμ. 11, σσ. 612–613, Νοέ. 1979, ISSN: 0001-0782. DOI: 10.1145/359168.359176. διεύθν.: https://doi.org/10.1145/359168.359176.

[36] G. R. Blakley, «Safeguarding cryptographic keys», στο *Managing Requirements Knowledge, International Workshop on*, IEEE Computer Society, 1979, σσ. 313–313.

[37] V. Shoup, «Lower bounds for discrete logarithms and related problems», στο *International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 1997, σσ. 256–266.

[38] N. Koblitz και A. Menezes, *Another look at generic groups*, Cryptology ePrint Archive, Paper 2006/230, https://eprint.iacr.org/2006/230, 2006. διεύθν.: https://eprint.iacr.org/2006/230.

[39] M. Bellare και P. Rogaway, «Optimal asymmetric encryption», στο *Workshop on the Theory and Application of of Cryptographic Techniques*, Springer, 1994, σσ. 92–111.

[40] P. Q. Nguyen και I. E. Shparlinski, «On the insecurity of a server-aided rsa protocol», στο *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2001, σσ. 21–35.

[41] S. Goldwasser και S. Micali, «Probabilistic encryption & how to play mental poker keeping secret all partial information», στο *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, σειρά STOC '82, San Francisco, California, USA: Association for Computing Machinery, 1982, σσ. 365–377, ISBN: 0897910702. DOI: 10.1145/800070.802212. διεύθν.: https://doi.org/10.1145/800070.802212.

[42] M. Bellare και P. Rogaway, «Random oracles are practical: A paradigm for designing efficient protocols», στο *Proceedings of the 1st ACM Conference on Computer and Communications Security*, σειρά CCS '93, Fairfax, Virginia, USA: Association for Computing Machinery, 1993, σσ. 62–73, ISBN: 0897916298. DOI: 10.1145/168588.168596. διεύθν.: https://doi.org/10.1145/168588.168596.

[43] M. Bellare και P. Rogaway, «The game-playing technique», *International Association for Cryptographic Research (IACR) ePrint Archive: Report*, τόμ. 331, σ. 2004, 2004.

[44] V. Shoup, *Sequences of games: A tool for taming complexity in security proofs*, Cryptology ePrint Archive, Paper 2004/332, https://eprint.iacr.org/2004/332, 2004. διεύθν.: https://eprint.iacr.org/2004/332.

[45] T. Alpcan, «Game theory for security», στο *Encyclopedia of Systems and Control*, J. Baillieul και T. Samad, επιμελητές. London: Springer London, 2019, σσ. 1–5, ISBN: 978-1-4471-5102-9. DOI: 10.1007/978-1-4471-5102-9_37-2. διεύθν.: https://doi.org/10.1007/978-1-4471-5102-9_37-2.

[46] Y. Lindell, *How to simulate it - a tutorial on the simulation proof technique*, Cryptology ePrint Archive, Paper 2016/046, https://eprint.iacr.org/2016/046, 2016. διεύθν.: https://eprint.iacr.org/2016/046.

[47] R. Canetti, *Universally composable security: A new paradigm for cryptographic protocols*, Cryptology ePrint Archive, Paper 2000/067, https://eprint.iacr.org/2000/067, 2000. διεύθν.: https://eprint.iacr.org/2000/067.

[48] R. Cramer, I. B. Damgård κ.ά., *Secure multiparty computation*. Cambridge University Press, 2015.

[49] Y. Lindell, *Secure Multiparty Computation (MPC)*, Cryptology ePrint Archive, Paper 2020/300, https://eprint.iacr.org/2020/300, 2020. DOI: 10.1145/3387108. διεύθν.: https://eprint.iacr.org/2020/300.

[50] C. Zhao, S. Zhao, M. Zhao κ.ά., «Secure multi-party computation: theory, practice and applications», *Information Sciences*, τόμ. 476, σσ. 357–372, 2019.

[51] C. Gentry, S. Halevi, M. Raykova και D. Wichs, *Garbled RAM Revisited, Part I*, Cryptology ePrint Archive, Paper 2014/082, https://eprint.iacr.org/2014/082, 2014. διεύθν.: https://eprint.iacr.org/2014/082.

[52] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan και N. Zeldovich, «How to run turing machines on encrypted data», στο *Annual Cryptology Conference*, Springer, 2013, σσ. 536–553.

[53] M. Keller, «MP-SPDZ: A versatile framework for multi-party computation», στο *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. DOI: 10.1145/3372297.3417872. διεύθν.: https://doi.org/10.1145/3372297.3417872.

[54] M. Ben-Or, S. Goldwasser και A. Wigderson, «Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract)», στο *STOC*, 1988.

[55] O. Goldreich, S. Micali και A. Wigderson, «How to play any mental game, or a completeness theorem for protocols with honest majority», στο *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, σσ. 307–328.

[56] D. Demmler, T. Schneider και M. Zohner, «ABY-A framework for efficient mixed-protocol secure two-party computation.», στο *NDSS*, 2015.

[57] Z. Xian-Yi, W. Qian, Z. Yun-Quan κ.ά., «openblas: a high performance blas library on loongson 3a cpu», 2011.

[58] D. Rathee, A. Bhattacharya, R. Sharma, D. Gupta, N. Chandran και A. Rastogi, *Secfloat: Accurate floating-point meets secure 2-party computation*, Cryptology ePrint Archive, Paper 2022/322, https://eprint.iacr.org/2022/322, 2022. διεύθν.: https://eprint.iacr.org/2022/322.

[59] P. S. N. ALX, N. V. ALX, P. F. AU κ.ά., «D1. 1 state of the art analysis of mpc techniques and frameworks»,

[60] W. J. Cody, *Software Manual for the Elementary Functions (Prentice-Hall Series in Computational Mathematics)*. USA: Prentice-Hall, Inc., 1980, ISBN: 0138220646.

[61] A. Aly, K. Cong, D. Cozzo κ.ά., *Scale--mamba v1. 12: Documentation*, 2021.

[62] Alexandra Institute, *FRESCO - a FRamework for Efficient Secure COmputation*, https://github.com/aicis/fresco.

[63] C. C. Pinter, *A book of abstract algebra*. Courier Corporation, 2010.

[64] G. L. Miller, «Riemann's hypothesis and tests for primality», *Journal of computer and system sciences*, τόμ. 13, αρθμ. 3, σσ. 300–317, 1976.

[65] L. Babai, «Trading group theory for randomness», στο *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, σειρά STOC '85, Providence, Rhode Island, USA: Association for Computing Machinery, 1985, σσ. 421–429, ISBN: 0897911512. DOI: 10.1145/22145.22192. διεύθν.: https://doi.org/10.1145/22145.22192.

[66] A. Shamir, «Ip= pspace», *Journal of the ACM (JACM)*, τόμ. 39, αρθμ. 4, σσ. 869–877, 1992.

[67] C. Lund, L. Fortnow, H. Karloff και N. Nisan, «Algebraic methods for interactive proof systems», *Journal of the ACM (JACM)*, τόμ. 39, αρθμ. 4, σσ. 859–868, 1992.

Κεφάλαιο 8

Παράρτημα

8.1 Μαθηματικό Υπόβαθρο

8.1.1 Αφηρημένη Άλγεβρα

Σε αυτήν την ενότητα θα γίνει μια μικρή και ιδιαίτερα περιορισμένη εισαγωγή στην Αφηρημένη Άλγεβρα που είναι απαραίτητο εργαλείο στην μελέτη της κρυπτογραφίας. Σε καμία περίπτωση δεν είναι πλήρης και κυρίως θα μελετηθούν έννοιες που πιστεύουμε θα διευκολύνουν τον αναγνώστη αποκλειστικά της συγκεκριμένης εργασίας. Ακόμα, παραλείπουμε τις αποδείξεις από κάθε θεώρημα, ωστόσο οι περισσότερες από αυτές είναι αρκετά απλές και μπορούν να βρεθούν σε ένα εισαγωγικό βιβλίο Αφηρημένης Άλγεβρας. Παροτρύνουμε τον μη μυημένο αναγνώστη που επιθυμεί να αποκτήσει μια πιο σφαιρική γνώση γύρω από το αντικείμενο αυτό να απευθυνθεί σε εισαγωγικά βιβλία, όπως το [63] το οποίο αποτελεί και κύρια βιβλιογραφική πηγή της παρούσας ενότητας.

Στην συνέχεια θα ξεκινήσουμε παραθέτοντας βασικούς ορισμούς της Αφηρημένης Άλγεβρας ξεκινόντας από τη θεωρία ομάδων και έπειτα συνεχίζοντας με τη θεωρία των δακτυλίων. Όπου κρίνεται απαραίτητο θα γίνονται σχόλια και αναφορές που έχουν άμεση σχέση με τα κρυπτογραφικά εργαλεία που χρησιμοποιούνται στα προηγούμενα κεφάλαια.

Definition 8.1.1. Ομάδα (Group) : Είναι η αλγεβρική δομή $\langle G, \circ \rangle$ που αποτελείται από ένα μη κενό σύνολο G, εφοδιασμένο μία πράξη $\circ : G \times G \to G$, που ικανοποιεί τις παρακάτω ιδιότητες:

- Κλειστότητα : $\forall a, b \in G : a \circ b \in G$
- Προσεταιριστικότητα : $\forall a, b, c \in G : (a \circ b) \circ c = a \circ (b \circ c)$
- Ουδέτερο στοιχείο : $\exists! e \in G, \forall a \in G : a \circ e = a$

- Αντίστοφο στοιχείο : $\forall a \in G, \exists b \in G : a \circ b = e$

Στα πλαίσια αυτής της ενότητας, μια ομάδα $\langle G, \circ \rangle$ την αποκαλούμε με το όνομα του συνόλου που εμπεριέχει, δηλαδή G στην προκείμενη περίπτωση και όπου δεν κρίνεται αναγκαίο μπορεί να παραλείπεται η αναφορά στην πράξη που συνοδεύει την ομάδα G. Ακόμα η προκαθορισμένη σημειογραφία που θα χρησιμοποιηθεί για την πράξη της ομάδας είναι ο πολλαπλασιασμός (χωρίς αυτό να σημαίνει κάτι παραπάνω για την εσωτερική της δομή) και χρησιμοποιείται διαφορετικό σύμβολο μόνο όπου κρίνεται απαραίτητο για ευκολότερη ανάγνωση.

Ανάλογα με το είδος της πράξης, αν είναι για παράδειγμα πολλαπλασιασμός ή αφαίρεση, συνηθίζουμε να την αποκαλούμε **Πολλαπλασιαστική** ή **Προσθετική Ομάδα** αντίστοιχα και ουδέτερα στοιχεία τους 0 και 1 αντίστοιχα. Ωστόσο, αυτό δεν είναι καθοριστικό για την εσωτερική της δομή, καθώς πολλές ομάδες περιέχουν πράξεις που ουδεμία σχέση έχουν με τον πολλαπλασιασμό ή την αφαίρεση. Αν η ομάδα διαθέτει επίσης και την ιδιότητα της αντιμεταθετικότητας την αποκαλούμε **Αβελιανή Ομάδα**. Ακόμα, η τάξη μια ομάδας G, ορίζεται ως το πλήθος των στοιχείων της $ord(G) = |G|$, σε περίπτωση που είναι πεπερασμένη η τάξη της η ομάδα ονομάζεται **Πεπερασμένη Ομάδα**.

Definition 8.1.2. Υποομάδα (Subgroup) : Έστω μια ομάδα $\langle G, \circ \rangle$ και $\emptyset \subset \mathrm{H} \subseteq G$. Η αλγεβρική δομή $\langle H, \circ \rangle$ εφοδιασμένη με την ίδια ακριβώς πράξη $\circ$ ονομάζεται υποομάδα της G και την συμβολίζουμε $H \leq G$ αν, και μόνο αν είναι κλειστή ως προς την $\circ$ και την ύπαρξη αντίστροφου.

Μπορούμε τώρα να ορίσουμε μια ειδική μορφή υποομάδας που θα αποδειχθεί χρήσιμη στη συνέχεια.

Definition 8.1.3. Κανονική Υποομάδα : Έστω η ομάδες G, H με $G \leq H$. Το H είναι κανονική υποομάδα αν είναι κλειστεί ως προς την συζυγία και την συμβολίζουμε $H \trianglelefteq G$. Δηλαδή :

$$\forall a \in H, x \in G : xax^{-1} \in H$$

Μια ακόμη πολύ σημαντική κατηγορία ομάδων που έχουν μεγάλη χρησιμότητα στον κλάδο της κρυπτογραφίας είναι αυτή των κυκλικών ομάδων.

Definition 8.1.4. Κυκλική Ομάδα (Cyclic group : Μια ομάδα G όπου κάθε στοιχείο της $a \in G$ είναι μπορεί να εκφραστεί ως δύναμη κάποιου συγκεκριμένου στοιχείου της $g \in G$, το οποίο ονομάζεται **γεννήτορας (generator** της κυκλικής ομάδας. Δηλαδή :

$$\exists g \in G, \forall a \in G, \exists n \in \mathbf{Z} : a = g^n$$

Στο σημείο αυτό μπορούμε να ορίσουμε την συνάρτηση του Διακριτού Λογαρίθμου, μια πολύ γνωστή συνάρτηση στην κρυπτογραφία.

Definition 8.1.5. Διακριτός Λογαρίθμος (Discrete Logarithm ή DL) : Έστω μια κυκλική ομάδα $\langle G, \cdot \rangle$, τότε γνωρίζουμε ότι $G = \langle g \rangle$ όπου g ένας γεννήτορας της ομάδας, δηλαδή $\forall a \in G \exists x : g^x = a \in G$. Ο Διακριτός Λογάριθμος ορίζεται ως η συνάρτηση $log_g(a)$ που επιστρέφει την δύναμη x για την οποία $g^x = a$, δηλαδή $log_g(a) = x \Leftrightarrow g^x = a$.

Από τον Διακριτό Λογάριθμο προκύπτει και το γνωστό **Πρόβλημα του Διακριτού Λογαρίθμου (Discrete Logarithm Problem ή DLP)** το οποίο έγκειται στον υπολογισμό την συνάρτησης του DL με κάποιον γενικευμένο αλγόριθμο αναζήτησης. Το πρόβλημα αυτό θεωρείται πως στο κλασσικό μοντέλο υπολογισμού ότι είναι δυσεπίλυτο δηλαδή ότι δεν υπάρχει πολυωνυμικός στον αριθμό των ψηφίων της ομάδας G (δηλαδή του μεγέθους της τάξης της ομάδας) αλγόριθμος που να το επιλύει.

Στην συνέχεια μπορούμε να δούμε πως ορίζονται οι βασικές ιδιότητες του ομομορφισμού και του ισομορφισμού οι οποίες απαντώνται στα Ομομορφικά Κρυπτογραφικά Σχήματα αλλά και έχουν σημαντικό ρόλο στο παρακάτω θεώρημα.

Definition 8.1.6. Ομομορφισμός Ομάδας (Group Homomorphism) : Έστω $\langle G, \circ \rangle$, $\langle H, \star \rangle$ ομάδες, μια συνάρτηση $f : G \to H$ είναι ομομορφισμός αν και μόνο αν :

$$\forall a, b \in G : f(a \circ b) = f(a) \star f(b)$$

Definition 8.1.7. Ισομορφισμός (Isomoprhism) : Μια συνάρτηση f είναι ισομορφισμός αν, και μόνο αν, είναι ομομορφισμός και αμφιμονοσήμαντη.

Σε αυτό το σημείο μπορούμε να αναφερθούμε σε ένα θεμελιώδες θεώρημα της Θεωρίας Ομάδων, το οποίο βρίσκεται πίσω από πολλές ιδιότητες των ομάδων και των δακτυλίων που χρησιμοποιούνται στην κρυπτογραφία.

Theorem 1. Θεώρημα Lagrange (Lagrange's theorem) : Σε κάθε πεπερασμένη ομάδα G, η τάξη κάθε υποομάδας της $\mathrm{H} \leq G$ διαιρεί την τάξη της G, δηλαδή

$$\forall H \leq G, ord(G) | ord(H)$$

Από το παραπάνω μπορεί να αποδειχθεί το εξής θεώρημα :

Theorem 2. Κάθε πεπερασμένη ομάδα πρώτης τάξης είναι κυκλική και κάθε στοιχείο της είναι γεννήτορας της ομάδας.

Οι επόμενοι δύο ορισμοί είναι απαραίτητα εργαλεία στην κατανόηση του παρακάτω θεωρήματος.

Definition 8.1.8. Συσύνολο Ομάδας (Group Coset) : Έστω $\langle G, \circ \rangle$ με $G = \{a_1, a_2, ...\}$ (η ομάδα μπορεί να είναι πεπερασμένη ή και μη) μια πολλαπλασιαστική ομάδα και μια υποομάδα Η της G. Για κάθε στοιχείο $a \in G$ ορίζουμε τα παρακάτω :

- Αριστερό Συσύνολο : $a \circ H = \{a \circ a_1, a \circ a_2, ...\}$
- Δεξί Συσύνολο : $a \circ H = \{a_1 \circ a, a_2 \circ a, ...\}$

Για τα συσύνολα ομάδων αφού ως προκαθορισμένη σημειογραφία για την πράξη της ομάδας είναι ο πολλαπλασιασμός προκύπτουν οι συμβολισμοί aΗ και Ha αντίστοιχα.

Definition 8.1.9. Ομάδα Υπολοίπου : Έστω οι G, Η του Ορισμού 8.1.8 με τον περιορισμό ότι $H \trianglelefteq H$. Ορίζουμε τον παρακάτω συμβολισμό $G/H = \{Ha_1, Ha_2, Ha_3\}$. Το σύνολο G/H εφοδιασμένο με την πράξη του πολλαπλασιασμού συσυνόλων, μπορεί να αποδειχτεί ότι αποτελεί ομάδα και το ονομάζουμε **Ομάδα Υπολοίπου της G προς το Η**.

Definition 8.1.10. Θεμελιώδες Ομομορφικό Θεώρημα (Fundamental Homomorphism Theorem (FHT)) : Έστω G,H δύο ομάδες όπου $H \trianglelefteq G$ και $f : G \to H$ ένας ομομορφισμός από το G στο H. Αν K είναι ο πυρήνας της f τότε :

$$\text{Η} \cong G/K$$

Στην συνέχεια θα γίνει μια μικρή εισαγωγή στη θεωρία των δακτυλίων, η οποία είναι ιδιαίτερα περιορισμένη και δίνεται έμφαση όπου αυτό κρίνεται απαραίτητο.

Definition 8.1.11. Δακτύλιος : Είναι αλγεβρική δομή $\langle R, \circ, \star \rangle$, η οποία αποτελείται από ένα μη κενό σύνολο R, εφοδιασμένο με δύο διμελείς πράξεις $\circ : R \times R \to R$ και $\star : R \times R \to R$, που ικανοποιεί τα ακόλουθα αξιώματα :

- Η αλγεβρική δομή $\langle R, \circ \rangle$ είναι αβελιανή ομάδα.
- Η αλγεβρική δομή $\langle R, \star \rangle$ είναι μονοειδές.
- Η πράξη $\star$ είναι επιμεριστική ως προς την $\circ$:
  - Αριστερός Επιμεριστικός νόμος : $\forall a, b, c \in R : a \star (b \circ c) = (a \star b) \circ (a \star c)$
  - Δεξιός Επιμεριστικός νόμος :$\forall a, b, c \in R : (b \circ c) \star a = (b \star a) \circ (c \star a)$

Αντίστοιχα με την περίπτωση των ομάδων έτσι και με αυτή των δακτυλίων θα ακολουθήσουμε παρόμοια τακτική στην σημειολογία. Με την μόνη διαφορά ότι στην περίπτωση των δακτυλίων η προκαθορισμένη σημειογραφία που θα χρησιμοποιήσουμε για τις πράξεις είναι

αυτή της πρόσθεσης και του πολλαπλασιασμού για την πρώτη και την δεύτερη πράξη ενός δακτύλιο αντίστοιχα.

Σε αυτό το σημείο πρέπει να αναφέρουμε ότι σε ένα μικρότερο ποσοστό της βιβλιογραφίας ο παραπάνω ορισμός του δακτυλίου αναφέρεται ως δακτύλιος με μονάδα, αναφερόμενος στο πολλαπλασιαστικό ταυτοτικό στοιχείο. Στη περίπτωση που η δομή του δακτυλίου δεν εμπεριέχει πολλαπλασιαστικό ταυτοτικό στοιχείο τότε η δομή του πολλαπλασιασμού μετατρέπεται σε ημι-ομάδα (semi-group). Η σύμβαση που ακολουθείται ως επί το πλείστον στην βιβλιογραφία, είναι σύμφωνη με τον παραπάνω ορισμό του δακτυλίου που δώσαμε ενώ η δομή χωρίς την πολλαπλασιαστική μονάδα απαντάται ως Δακτύλιος χωρίς μονάδα (Ring without "i"dentity ή Rng). Ένας δακτύλιος ο οποίος είναι αντιμεταθετικός ως προς τον πολλαπλασιασμό αποκαλείται **Αντιμεταθετικός Δακτύλιος**.

Definition 8.1.12. Υποδακτύλιος (Subring) : Έστω ένας δακτύλιος $\langle R, \circ, \star \rangle$ και $\emptyset \subset \mathrm{H} \subseteq G$. Η αλγεβρική δομή $\langle H, \circ, \star \rangle$ εφοδιασμένη με τις ίδιες ακριβώς πράξεις με αυτή του R είναι υποδακτύλιος αν και μόνο αν είναι κλειστή ως προς την $\circ$, τον αντίστροφο ως προς την $\circ$ και κλειστή ως προς την $\star$.

Definition 8.1.13. Ιδεώδες (Ideal) : Έστω οι δακτύλιοι A, B με το B να είναι υποδακτύλιος του A. Το B είναι ιδεώδες αν, και μόνο αν απορροφά τον πολλαπλασιασμό στο A. Δηλαδή :

$$\forall a \in A, \forall b \in B : ab \in B$$

Definition 8.1.14. Ομομορφισμός Δακτυλίου (Ring Homomorphism) : Έστω $\langle A, \circ_1, \star_1 \rangle$, $\langle B, \circ_2 \star_2 \rangle$ ομάδες, μια συνάρτηση $f : A \to B$ είναι ομομορφισμός αν και μόνο αν :

- $\forall a, b \in A : f(a \circ_1 b) = f(a) \circ_2 f(b)$
- $\forall a, b \in A : f(a \circ_2 b) = f(a) \star_2 f(b)$

Definition 8.1.15. Συσύνολο Δακτυλίου (Ring coset) : Έστω ένας δακτύλιος $\langle A, \circ, \star$ με $\mathrm{A} = \{a_1, a_2, ...\}$ και B ένα ιδεώδες του A. Για κάθε στοιχείο $a \in A$ το συσύνολο δακτυλίου ορίζεται ως εξής :

$$a \circ J = \{a \circ a_1, a \circ a_2, ...\}$$

Για τα συσύνολα δακτυλίων αφού ως προκαθορισμένη σημειογραφία για την πρώτη πράξη είναι η πρόσθεση προκύπτει ο συμβολισμός $a + J$. Προφανώς στην περίπτωση των δακτυλίων αφού ως προς την πρόσθεση είναι αβελιανή ομάδα έχουμε $a + J = J + a$.

Definition 8.1.16. Δακτύλιος Υπολοίπου (Quotient Ring) : Έστω το A ένας δακτύλιος και το J ένα ιδεώδες του. Το A/J εφοδιασμένο με τις πράξεις της πρόσθεσης και του πολλαπλασιασμού συσυνόλων είναι δακτύλιος και ονομάζεται **Δακτύλιος Υπολοίπου του A προς το J**

Definition 8.1.17. Θεμελιώδες Ομομορφικό Θεώρημα για δακτύλιους (Fundamental Homomorphism Theorem (FHT)) : Έστω δύο δακτύλιοι A και B όπου το B είναι ιδεώδες του A και $f : A \to B$ ένας ομομορφισμός από το A στο B με $K = ker(f)$. Τότε μπορεί να αποδειχθεί ότι :

$$\mathrm{B} \cong A/K$$

Definition 8.1.18. Πεδίο (Field) : Ένας αντιμεταθετικός δακτύλιος στον οποίο κάθε στοιχείο έχει πολλαπλασιαστικό και προσθετικό αντίστροφο.

Definition 8.1.19. Πεπερασμένο πεδίο (Finite field ή Galois Field) : Ένα πεδίο με πεπερασμένα στοιχεία.

Μπορεί να αποδειχθεί ότι η τάξη ενός πεπερασμένου πεδίου είναι πάντα πρώτος αριθμός ή δύναμη πρώτου αριθμού.

8.1.2 Θεωρία αριθμών

Στην αναφορά μας στη Θεωρία Αριθμών θα θα αρκεστούμε μόνο σε απαραίτητα συμπεράσματα τα οποία απαιτούνται για την κατανόηση της συγκεκριμένης εργασίας. Το παρακάτω θεώρημα είναι ένα από τα πολύ βασικά στην κρυπτογραφία και είναι έμμεση απορία του Θεωρήματος του Lagrange.

Definition 8.1.20. Μικρό Θεώρημα του Fermat (Fermat's Little Theorem ή FLT) : Έστω μια ομάδα $\langle G, \circ \rangle$ με τάξη $ord(G) = p$, όπου p ένας πρώτος αριθμός, τότε μπορεί να αποδειχθεί ότι :

$$\forall g \in G : g^p \equiv g(\quad (\mathrm{mod}\ p))$$

Προφανώς, αφού κάθε ομάδα πρώτης τάξης είναι και κυκλική, μπορούμε να εξάγουμε και άλλες ταυτότητες από το παραπάνω θεώρημα, όπως ότι $\forall g \in G : g^{p-1} \equiv 1(\ (\mathrm{mod}\ p))$. Στην συνέχεια θα αναφερθούμε σε μια πολύ σημαντική συνάρτηση για τον κλάδο της κρυπτογραφίας.

Definition 8.1.21. Συνάρτηση Euler (Euler's totient function) : Ορίζεται ως η συνάρτηση $\varphi : \mathbf{Z} \to \mathbf{Z}$, $\varphi(n)$ και επιστρέφει τον αριθμό των σχετικά πρώτων αριθμών με τον αριθμό του ορίσματος. Μπορεί να υπολογιστεί από τον παρακάτω τύπο :

$$\varphi(n) = n \prod_{p|n} \left(1 - \frac{1}{p}\right)$$

Η συγκεκριμένη συνάρτηση έχει αποδειχθεί ότι για οποιονδήποτε μη πρώτο αριθμό n, ο υπολογισμός της είναι ισοδύναμος με το πρόβλημα της **Παραγοντοποίησης σε γινόμενο**

πρώτων παραγόντων [64]. Στην ισοδυναμία αυτή βασίζεται η ασφάλεια πολλών κρυπτογραφικών σχημάτων, όπως του γνωστού RSA. Το τελευταίο πρόβλημα εικάζεται ότι ανήκει στην κλάση προβλημάτων $co - NP \cap NP$ και μια από τις πολύ βασικές κρυπτογραφικές υποθέσεις είναι ότι το συγκεκριμένο πρόβλημα είναι δυσεπίλυτο στο κλασσικό μοντέλο υπολογισμού. Αντιθέτως, έχει αποδειχθεί ότι στο κβαντικό μοντέλο υπολογισμού ανήκει μπορεί να υπολογιστεί αποδοτικά από τον Αλγόριθμο του Shor και ότι ανήκει και στην κλάση BQP.

8.1.3 Άλγεβρα

8.1.3.1 Αναπαράσταση Πολυωνύμου

Στην παρούσα Ενότητα παρουσιάζεται, χωρίς ιδιαίτερη εμβάθυνση, κυρίως το μαθηματικό υπόβαθρο που απαιτείται για να γίνει απολύτως κατανοητή η Διαμοίραση Μυστικών Shamir που μελετάμε στο Κεφάλαιο 4 αλλά και η περίπτωση της πράξης του πολλαπλασιασμού του Πρωτόκολλου BGW που μελετάμε επίσης στο ίδιο κεφάλαιο.

Theorem 3. Ισοδυναμία Αναπαράστασης Πολυωνύμου : Κάθε πολυώνυμο n-οστού βαθμού $\alpha(x) = \alpha_n x^n + \alpha_{n-1} x^n + ... + \alpha_0 x + \gamma$ μπορεί να περιγραφεί κατά μοναδικό τρόπο είτε από τους $n+1$ συντελεστές του $[\alpha_n, \alpha_{n-1}, ..., \alpha_0, \gamma]$ είτε από $n+1$ διαφορετικές αποτιμήσεις του πολυωνύμου $[\alpha(x_1), \alpha(x_2), ..., \alpha(x_{n+1}]$ στα σημεία $x_1, x_2, ..., x_{n+1}$. Δηλαδή αυτές οι δύο αναπαραστάσεις είναι ισοδύναμες.

Η απόδειξη του παραπάνω είναι κατασκευαστική είναι πολύ απλή. Ας υποθέσουμε ότι $\mathbf{c} = [\alpha_n, \alpha_{n-1}, ..., \alpha_0, \gamma]^T$ το διάνυσμα των συντελεστών του $\alpha(x)$ και ότι $\mathbf{c} = [\alpha(x_1), \alpha(x_2), ..., \alpha(x_{n+1}]^T$ το διάνυσμα των αποτιμήσεων. Υπάρχει μια $1-1$ συνάρτηση η οποία μετατρέπει από την μια αναπαράσταση στην άλλη. Η συνάρτηση για την μετατροπή από την περιγραφή μέσω συντελεστών του πολυωνύμου στην περιγραφή μέσω αποτιμήσεων είναι η παρακάτω :

$$\mathbf{V} = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & ... & \alpha_1^{2t} \\ 1 & \alpha_2 & \alpha_2^2 & ... & \alpha_2^{2t} \\ \vdots & & & & \\ 1 & \alpha_n & \alpha_n^2 & ... & \alpha_n^{2t} \end{bmatrix}$$

και μπορεί να εφαρμοστεί ως εξής :

$$\mathbf{s} = \mathbf{V} \times \mathbf{c}$$

Αντίστοιχα μπορούμε επειδή το $\mathbf{V}$ είναι μητρώο Vandermonde είναι και αντιστρέψιμο. Έτσι μπορούμε να υπολογίσουμε το διάνυσμα συντελεστών $\mathbf{c}$ από το διάνυσμα αποτιμήσεων $\mathbf{s}$ ως εξης :

$$\mathbf{c} = \mathbf{V}^{-1} \times \mathbf{s}$$

8.1.3.2 Παρεμβολή Lagrange (Lagrange Interpolation)

Πρόκειται από τις πιο γνωστές και βασικές μεθόδους παρεμβολής.

Definition 8.1.22. Παρεμβολή Lagrange : Έστω ότι θέλουμε να παρεμβάλουμε $k+1$ σημεία για τα οποία μας δίνονται οι τιμές τους στη μορφή (x_i, y_i) όπου $i \in 0, 1, ... k$ και $x_i \neq x_j \forall i, j$. Τότε αρκεί να παρατηρήσουμε ότι χρειαζόμαστε μια συνάρτηση l_j με την παρακάτω μορφή :

$$l_j(x) = \begin{cases} 1, x = x_j \\ 0, x = x_i \\ \text{κάτι άλλο, σε οποιαδήποτε άλλη περίπτωση} \end{cases}$$

Τότε μπορούμε να δημιουργήσουμε μια παρεμβολή από τα σημεία που μας δίνονται ως εξής :

$$L(x) := \sum_{j=0}^{k} y_j l_j(x)$$

Η $L(x)$ μπορούμε να επαληθέυσουμε ότι παρεμβάλει τα σημεία (x_i, y_i) αφού $L(x_j) = y_j \forall j \in 0, 1, ..., k$ Τέλος αρκεί να βρούμε μια κατάλληλη συνάρτηση $l_j(x)$. Αυτή προφανώς μπορεί να οριστεί ως εξής :

$$\ell_j(x) := \prod_{\substack{0 \leq m \leq k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)}$$

Μπορούμε να επαληθεύσουμε ότι η $l_j(x)$ όντως έχει τα χαρακτηριστικά που επιθυμούμε :

- Για $j = i$:
$$\ell_j(x_j) := \prod_{m \neq j} \frac{x_j - x_m}{x_j - x_m} = 1$$
- Για $j \neq i$
$$\ell_j(x_i) = \prod_{m \neq j} \frac{x_i - x_m}{x_j - x_m} = \frac{(x_i - x_0)}{(x_j - x_0)} \cdots \frac{(x_i - x_i)}{(x_j - x_i)} \cdots \frac{(x_i - x_k)}{(x_j - x_k)} = 0.$$
- Για οποιαδήποτε άλλη περίπτωση
$$\ell_j(x_j) := \prod_{m \neq j} \frac{x_j - x_m}{x_j - x_m}$$

8.2 Αλγοριθμικό Υπόβαθρο

Στην ενότητα αυτή παραθέτουμε και αναλύουμε, χωρίς εμβάθυνση, ορισμένους βασικούς ορισμούς από τη Θεωρία Πολυπλοκότητας. Κύρια βιβλιογραφική πηγή στη συγκεκριμένη ενότητα αποτέλεσε η [4].

8.2.1 Βασικοί Ορισμοί

Αρχικά, θεωρούμε σημαντικό να διασαφηνίσουμε τον όρο αποδοτικός αλγόριθμος στα πλαίσια της κρυπτογραφίας. Αυτός ορίζεται ως εξής :

Definition 8.2.1. Αποτελεσματικός αλγόριθμος (efficient algorithm) : Κάθε αλγόριθμος ο οποίος έχει χρονική και χωρική πολυπλοκότητα $O(poly(n))$, όπου n είναι το μέγεθος της εισόδου του αλγορίθμου.

Ακόμα, θέλουμε να διασαφηνίσουμε τον όρο Boolean δίκτυο στα πλαίσια του SMPC, αυτό θα μας είναι χρήσιμο στο Κεφάλαιο 4, σε SMPC πρωτόκολλα που πραγματοποιούν υπολογισμούς πάνω σε Boolean δίκτυα. Ένα Boolean δίκτυο ορίζεται ως εξής :

Definition 8.2.2. Boolean Δίκτυο : Ένα κατευθυνόμενο ακυκλικό γράφημα (Directed Acyclic Graph ή DAG), $G = (E, V)$, όπου κάθε κόμβος $v \in V$ είναι μια Boolean πύλη και κάθε ακμή μια σύνδεση μεταξύ των boolean πυλών. Για κάθε κόμβο $v \in V, deg^{-}(v) \geq 1$ και $deg^{+}(v) = 1$.

Παραθέτουμε επίσης τον ορισμό του Μαντείου, που απαντάται στο Κεφάλαιο 3, υπό την μορφή του Τυχαίου Μαντείου που είναι και η πιο συνήθης μορφή του που τον συναντάμε στον κλάδο της κρυπτογραφίας.

Definition 8.2.3. Μαντείο (Oracle) : Πρόκειται για έναν αλγόριθμο μαύρου κουτιού ο οποίος δέχεται ερωτήσεις και δίνει απαντήσεις.

8.2.2 Διαδραστικές Αποδείξεις

Τέλος, θα κάνουμε μια μικρή αναφορά στις Διαδραστικές Αποδείξεις στις οποίες βασίζονται οι Διαδραστικές ή Μη Αποδείξεις Μηδενικής Γνώσης, οι οποίες αποτελούν βασικό εργαλείο της σύγχρονης κρυπτογραφίας και στις οποίες αναφερθήκαμε στο Κεφάλαιο 2.

Για να μπορέσουμε να αντιληφθούμε την έννοια και την χρησιμότητα των αποδείξεων πρέπει να ξεκινήσουμε τη μελέτη από το παραδοσιακό σύστημα αποδείξεων. Ας πάρουμε για παράδειγμα μια απόδειξη για μια Boolean έκφραση x που θέλουμε να αποδείξουμε ότι είναι, μη ικανοποιήσιμη, δηλαδή ότι δεν υπάρχει ανάθεση τιμών που να οδηγεί σε αληθή αποτίμηση. Ονομάζουμε **Γλώσσα** L, το (άπειρο σύνολο) όλων των Boolean εκφράσεων που είναι μη ικανοποιήσιμες και ονομάζουμε **Μάρτυρα ή Πιστοποιητικό** ενός στοιχείου x την

απόδειξη ότι $x \in L$. Το συγκεκριμένο πρόβλημα ανάγεται στο να αποδείξουμε η έκφραση x ανήκει στη γλώσσα UNSAT, δηλαδή $x \in$ UNSAT. Επίσης, γνωρίζουμε ότι UNSAT $\in$ coNP. Επιστρέφοντας στο παράδειγμα μας, ας υποθέσουμε ότι ένας άνθρωπος, ο **Αποδεικνύον (Prover)** P (θεωρούμε ότι διαθέτει υπολογιστικά απεριόριστους πόρους), θέλει να αποδείξει σε κάποιον άλλο, τον **Επαληθευτή (Verifier)** V (θεωρούμε ότι διαθέτει υπολογιστικά περιορισμένους πόρους), ότι $x \in$ UNSAT. Σε μια μη διαδραστική απόδειξη για το UNSAT, η καλύτερη λύση που γνωρίζουμε είναι ο P να γράψει τις αποτιμήσεις της x για όλες τις πιθανές τιμές των μεταβλητών της σε ένα χαρτί[1], ο V να πάρει το χαρτί να το διαβάσει και αν πειστεί από την απόδειξη να δηλώσει ότι αποδέχεται ότι $x \in$ UNSAT. Στο παράδειγμα μας, αν θεωρήσουμε ότι υπάρχουν n μεταβλητές στην έκφραση x τότε το μέγεθος της απόδειξης είναι τουλάχιστον $\Omega(2^n)$, αυτό σημαίνει ότι ο V δεν μπορεί να διαθέτει υπολογιστικά περιορισμένους πόρους παρά μόνο υπολογιστικά απεριόριστους πόρους για να μπορέσει να διαβάσει όλη την απόδειξη από το χαρτί. Παρατηρούμε ότι αυτό δε θα ήταν το ίδιο με ένα πρόβλημα $x \in$ NP, αφού ο μάρτυρας στη συγκεκριμένη περίπτωση είναι πολυωνυμικός στο μέγεθος της εισόδου και έτσι ένας V με υπολογιστικά περιορισμένους πόρους είναι αρκετός. Στην περίπτωση μάλιστα του SAT το μέγεθος του μάρτυρα είναι n, δηλαδή γραμμικό στο μέγεθος της εισόδου. Οι Διαδραστικές Αποδείξεις (Interactive Proofs ή IP) ουσιαστικά είναι μια μέθοδος αποδείξεων, μια αφηρημένη υπολογιστική μηχανή, η οποία μας επιτρέπει να διατηρήσουμε την ισχύ του P άπειρη και την ισχύ του V υπολογιστικά περιορισμένη, που συνεπάγεται ότι και το μέγεθος του μάρτυρα θα πρέπει να είναι πολυωνυμικό, με το μοναδικό μειονέκτημα να υπάρχει πιθανότητα $< 1/3$ ο P να μπορεί να κάνει τον V να αποδεχθεί χωρίς ο ίδιος ο P να γνωρίζει πραγματικά έναν μάρτυρα. Δηλαδή το μοναδικό μειονέκτημα των IP είναι ότι μπορεί να υπάρχουν πλαστοί μάρτυρες/πιστοποιητικά τα οποία να μπορούν να οδηγήσουν τον V σε πλάνη. Οι διαδραστικές αποδείξεις παρουσιάστηκαν για πρώτη φορά στην βιβλιογραφία από τον Babay στην εργασία [65] . Μπορούμε τώρα να ορίσουμε με πιο αυστηρό τρόπο όλα τα παραπάνω.

Investigate further

Definition 8.2.4. Διάδραση μεταξύ ντετερμινιστικών συναρτήσεων : Έστω συναρτήσεις $f, g : \{0,1\}^* \to \{0,1\}^*$. Μια k-γύρων διάδραση μεταξύ των f και g σε είσοδο $x \in \{0,1\}^*$, την οποία συμβολίζουμε ως $\langle f, g \rangle(x)$, είναι η ακολουθία των συμβολοσειρών $a_1, \ldots, a_k \in \{0,1\}^*$ που ορίζονται ως εξής :

[1] Η κλάση coNP περιέχει όλα τα προβλήματα για τα οποία εικάζουμε ότι δεν μπορεί να υπάρξει πολυωνυμική στο μέγεθος της εισόδου απόδειξη.

$$a_1 = f(x)$$
$$a_2 = g(x, a_1)$$
$$\dots$$
$$a_{2i+1} = f(x, a_1, \dots, a_{2i})$$
$$a_{2i+2} = g(x, a_1, \dots, a_{2i+1})$$

όπου x είναι η είσοδος που θέλουμε να αποδείξουμε ότι $x \in L$

Στην περίπτωση της γλώσσας UNSAT, τοx είναι η Boolean έκφραση που θέλουμε να αποδείξουμε ότι $x \in$ UNSAT. Προφανώς, αν ο οι συναρτήσεις f και g έχουν εσωτερική κατάσταση που μπορεί να διατηρήσει το ιστορικών των προηγούμενων μηνυμάτων, δεν χρειάζεται όλο το ιστορικό μηνυμάτων να αποστέλλεται σε κάθε αλληλεπίδραση. Τώρα μπορούμε να ορίσουμε αυστηρά την κλάση των Ντετερμινιστικών Διαδραστικών Αποδείξεων :

Definition 8.2.5. Κλάση Ντετερμινιστικών Διαδραστικών Συναρτήσεων (Deterministic Interactive Proofs ή dIP**)**: Έστω P ντετερμινιστική συνάρτησή ΤΜ που είναι απεριόριστης ισχύος. Η γλώσσα $L \in dIP$ λέμε ότι διαθέτει ένα ντετερμινιστικό σύστημα διαδραστικής απόδειξης αν υπάρχει ντετερμινιστική ΤΜ V με είσοδο $x, a_1, \dots, a_i$ και χρονική πολυπλοκότητα $O(poly(|x|))$ που ικανοποιεί τις παρακάτω ιδιότητες :

- **Πληρότητα (Completeness)** : $x \in L \Rightarrow \exists P : \{0,1\}^* \to \{0,1\}^*$ out $_V\langle V, P\rangle(x) = 1$
- **Ορθότητα (Soundness)** : $x \notin L \Rightarrow \forall P : \{0,1\}^* \to \{0,1\}^*$ out $_V\langle V, P\rangle(x) = 0$

Μερικές σημείωσεις σχετικά με τον παραπάνω ορισμό είναι ότι αφού ο V έχει χρονική πολυπλοκότητα $O(poly(|x|))$, προφανώς και μπορεί να επεξεργαστεί $O(poly(|x|))$ αριθμό μηνυμάτων και άρα το μέγεθος του μάρτυρα $|w| = O(poly(|x|))$. Επίσης, ρόλος του V στον ορισμό αυτό είναι διπλός, είναι η ΤΜ που αλληλεπιδρά με τον P για την παραγωγή των μηνυμάτων της διάδρασης, δηλαδή του μάρτυρα, αλλά και η ΤΜ που θα επαληθεύσει τον μάρτυρα. Σχετικά με την κλάση dIP αποδειχθεί ότι dIP = NP . Ωστόσο, μεγαλύτερο ενδιαφέρον προκύπτει αν θεωρήσουμε μη ντετερμινιστικό τον Επαληθευτή V. Τότε η κλάση που προκύπτει είναι η γνωστή IP και ορίζεται ως εξής :

Add reference.

Definition 8.2.6. Κλάση (Μη Ντετερμινιστικών) Διαδραστικών Αποδείξεων ((Non-Deterministic) Interactive Proofs ή IP**)** : Έστω P ντετερμινιστική συνάρτησή ΤΜ που είναι απεριόριστης ισχύος. Η γλώσσα L λέμε ότι ανήκει στην κλάση IP, δηλαδή $L \in IP$ αν υπάρχει μη ντετερμινιστική ΤΜ V με $x, r, a_1, \dots, a_i, V$ και χρονική πολυπλοκότητα σε $O(poly(|x|))$ που ικανοποιεί τις παρακάτω ιδιότητες :

- **Πληρότητα (Completeness)** : $x \in L \Rightarrow \exists \Pr[\text{ out } _V\langle V, P\rangle(x) = 1] \geq (\varepsilon =)\frac{2}{3}$

- **Ορθότητα (Soundness)** : $x \notin L \Rightarrow \forall \Pr[\text{ out }_V\langle V, P\rangle(x) = 1] \leq 1 - \varepsilon$

Σχετικά με τον παραπάνω ορισμό, γνωρίζουμε ότι η έξοδος μιας μη ντετερμινιστικής ΤΜ είναι πιθανοτική και μπορούμε να σκεφτούμε το σύνολο των εισόδων της ως μια πιθανοτική κατανομή. Σχετικά με τη σταθερά ε έχει αποδειχθεί ότι μπορούμε να έχουμε Τέλεια Πληρότητα και Τέλεια Ορθότητα μόνο για γλώσσες στην κλάση NP. Οι τιμή $2/3$ είναι αυθαίρετη και αν επαναλάβουμε το πρωτόκολλο m αριθμό φορών μπορούμε να την αντικαταστήσουμε με $1 - negl(m)$. Έτσι ουσιαστικά με τους ορισμούς του Κεφαλαίο 3 μπορούμε να πούμε ότι έχουμε **Υπολογιστική Πληρότητα (Computational Completeness)** και **Υπολογιστική Ορθότητα (Computational Soundness)**. Ένα από τα σημαντικότερα ευρήματα του κλάδου είναι ότι $IP = PSPACE$. Η απόδειξη παρουσιάστηκε από τον Shamir το 1992 στην [66]. Τέλος, μπορούμε να παραλείψουμε ένα από τα πρώτα IP πρωτόκολλα, το Sumcheck, που προτάθηκαν στην [67] για τη διαδραστική απόδειξη της UNSAT γλώσσας.