Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ανώνυμη επικοινωνία ανθεκτική σε επιθέσεις ανάλυσης κίνησης δικτύου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΝΙΚΟΛΑΟΣ Δ. ΑΛΕΞΟΠΟΥΛΟΣ

Επιβλέπων : Αριστείδης Θ. Παγουρτζής
Αν. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2016

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Ανώνυμη επικοινωνία ανθεκτική σε επιθέσεις ανάλυσης κίνησης δικτύου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΝΙΚΟΛΑΟΣ Δ. ΑΛΕΞΟΠΟΥΛΟΣ

Επιβλέπων : Αριστείδης Θ. Παγουρτζής
Αν. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 30η Μαρτίου 2016.

.....	.....	.....
Αριστείδης Θ. Παγουρτζής Αν. Καθηγητής Ε.Μ.Π.	Άγγελος Ι. Κιαγιάς Αν. Καθηγητής Ε.Κ.Π.Α.	Δημήτριος Α. Φωτάκης Επίκ. Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2016

Νικόλαος Δ. Αλεξόπουλος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Νικόλαος Δ. Αλεξόπουλος, 2016.
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η διατήρηση της ιδιωτικότητας κατά την επικοινωνία στο διαδίκτυο έχει εξελιχθεί σε πολύ σημαντικό ζήτημα τα τελευταία χρόνια. Μεμονωμένα άτομα, επιχειρήσεις ή ακόμα και κυβερνήσεις υποκλέπτουν δεδομένα ώστε να τα χρησιμοποιήσουν προς το συμφέρον τους. Η κρυπτογράφηση του περιεχομένου των μηνυμάτων δεν είναι πλέον αρκετή, καθώς η ταυτοποίηση των δύο μερών που επικοινωνούν είναι συχνά αρκετή για να βγουν συμπεράσματα. Λύσεις για τη διατήρηση της ιδιωτικότητας στην επικοινωνία, όπως το Tor, υπάρχουν αλλά είναι ευάλωτες σε επιθέσεις ανάλυσης της κίνησης του δικτύου από αντιπάλους που έχουν εποπτεία του συνόλου του. Σε αυτή την εργασία, θα μελετήσουμε το πρόβλημα της ανώνυμης επικοινωνίας, με τη μορφή της σχεδόν πραγματικού χρόνου ανταλλαγής μηνυμάτων (πχ email, sms) και τις πιθανές λύσεις του.

Αρχικά παρουσιάζουμε ένα μαθηματικό μοντέλο και ορίζουμε το πρόβλημα τυπικά. Στη συνέχεια, και αφού παρουσιάσουμε κάποια απαραίτητα εργαλεία, αναλύουμε το σύστημα Vuvuzela των van den Hooff, Lazar,Zaharia και Zeldovich, το οποίο είναι ένα νέο σύστημα που προσπαθεί να λύσει το πρόβλημα. Σε αυτό το σημείο παρουσιάζουμε μια εκδοχή της θεωρία της διαφορικής ιδιωτικότητας, προσαρμοσμένη στο μοντέλο του προβλήματος που ορίσαμε και συμβατή με το σύστημα Vuvuzela.

Έχοντας ως έμπνευση το συγκεκριμένο σύστημα, στη συνέχεια αναζητούμε μία λύση για το πρόβλημα που να προσφέρει πιο ισχυρή ασφάλεια από τη διαφορική ιδιωτικότητα που προσφέρει το Vuvuzela. Προς αυτή την κατεύθυνση ορίζουμε το πρόβλημα ως μαθηματική συνάρτηση και δημιουργούμε κυκλώματα ώστε να γίνει δυνατή η αποτίμησή της, μέσω τεχνικών Secure Multiparty Computation. Μέσω της ασφαλούς αποτίμησης της συνάρτησης από κάποιους servers, το πρόβλημα της ανώνυμης ανταλλαγής μηνυμάτων μπορεί να λυθεί με πολύ καλές εγγυήσεις ασφαλείας. Σαν πρώτο βήμα παρουσιάζουμε ένα κύκλωμα απλό στην κατανόηση αλλά μη αποδοτικό, και στη συνέχεια εξελίσσουμε αυτό το κύκλωμα χρησιμοποιώντας επιπλέον ιδέες δικτύων ταξινόμησης (sorting networks) ώστε να καταλήξουμε σε ένα κύκλωμα που επιλύει το πρόβλημα και το οποίο έχει βάθος $\mathcal{O}(log^2 n)$, όπου n ο αριθμός των χρηστών- εισόδων.

Λέξεις κλειδιά

ανώνυμη επικοινωνία, ανωνυμία, ανταλλαγή μηνυμάτων, σημεία συνάντησης, ανάλυση κίνησης δικτύου, διαφορική ιδιωτικότητα, ασφαλής υπολογισμός, δίκτυα ταξινόμησης, δυαδικά κυκλώματα

Abstract

Preservation of one's privacy while communicating through the internet has evolved into a very serious problem in recent years. Individuals, businesses and even governments try to steal data in order to use it to their advantage. Encrypting the content of the messages isn't enough anymore, as metadata can reveal the identities of the two parties and that is often enough to make conclusions without ever reading the contents of the messages. Privacy-preserving solutions, like Tor, exist but are susceptible to traffic analysis attacks by a global adversary - that is an adversary that can monitor all network traffic. In this thesis, we will study the problem of anonymous communication in the form of near-real-time message exchange (e.g. email, sms) and its possible solutions.

First, we present a mathematical model and a formal definition of the problem. Then, after going through some useful tools such as differential privacy, mixnets and secure multiparty protocols, we analyze the Vuvuzela system published in 2015 by van den Hooff, Lazar, Zaharia and Zeldovich. Vuvuzela is a new system that uses the notion of differential privacy to asses the security offered.

Inspired by ideas from this system, we explore a possible solution that may offer information-theoretic security instead of differential privacy without a detrimental effect on the scalability. Towards this direction, we define the problem as a mathematical function and then create circuits that enable us to solve the problem using techniques of Secure Multiparty Computation (SMC). Using SMC, it is possible for 3 servers to facilitate the exchange of messages while guaranteeing information-theoretic security in the semi-honest model. As a first step, we present a circuit that is easy to understand but is not scalable. Next, we introduce a circuit based on sorting networks that can solve the problem and that has a practical number of gates and depth $\mathcal{O}(log^2 n)$, where n is the number of users.

Key words

anonymous communication, anonymity, message exchange, rendezvous points, traffic analysis, differential privacy, secure multiparty computation, sorting networks, binary circuits

Ευχαριστίες

Με αυτή την εργασία ένα ταξίδι πέντε και πλέον ετών στο ΕΜΠ ολοκληρώνεται. Ένα ταξίδι που μου χάρισε πολλές χαρές και διαμόρφωσε σε μεγάλο βαθμό την προσωπικότητά μου τόσο ως επιστήμονα-μηχανικό όσο και ως σκεπτόμενο άτομο.

Αρχικά θα ήθελα να ευχαριστήσω τον κ. Άγγελο Κιαγιά για τη συνεχή καθοδήγησή του καθ'όλη τη διάρκεια της έρευνας που οδήγησε σε αυτή την εργασία καθώς πολλά κομμάτια της οφείλονται σε δικές του ιδέες. Επίσης ευχαριστώ τον κ. Άρη Παγουρτζή για τις συμβουλές του και την υποστήριξή του κατά τη διάρκεια των τελευταίων χρόνων φοίτησής μου στη σχολή, καθώς επίσης και για το γεγονός ότι μέσα από το μάθημά του ανακάλυψα το ενδιαφέρον μου για την κρυπτογραφία.

Ένα μεγάλο ευχαριστώ επίσης δικαιωματικά πηγαίνει στην οικογένειά μου που μου επέτρεψε να εργαστώ απερίσπαστα κατά τη διάρκεια της φοίτησής μου και με υποστήριξε σε κάθε βήμα. Τέλος, θέλω να ευχαριστήσω όλους τους συμφοιτητές μου και ειδικά τους δύο Μιχάληδες, τον Αλέξανδρο και τον Ιάκωβο με τους οποίους πάντα κάναμε πολύ ενδιαφέρουσες συζητήσεις και ελπίζω να μείνουμε σε επαφή και μετά το πέρας αυτού του σταδίου της ζωής μου.

Νικόλαος Δ. Αλεξόπουλος,

Αθήνα, 30η Μαρτίου 2016

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος σχημάτων

Κατάλογος συμβόλων

$Dec(k, M)$	αποκρυπτογράφηση του μηνύματος M με το κλειδί k
$DH(sk, pk)$	ανταλλαγή κλειδιού Diffie-Hellman με ιδιωτικό κλειδί sk και δημόσιο κλειδί pk
$Enc(k, M)$	κρυπτογράφηση του μηνύματος M με το κλειδί k
$H(m)$	συνάρτηση hash εφαρμοσμένη στη συμβολοσειρά m
$pk_k^{u_i}$	προσωρινό (για ένα γύρο) δημόσιο κλειδί του χρήστη u_i για χρήση με το server k
pk_{server_k}	δημόσιο κλειδί του server k
pk_{u_i}	δημόσιο κλειδί του χρήστη u_i
s_i^j	μοιραζόμενο μυστικό μεταξύ του χρήστη u_j και του server i
sk_{server_k}	μυστικό- ιδιωτικό κλειδί του server k
sk_{u_i}	μυστικό- ιδιωτικό κλειδί του χρήστη u_i
SMC	secure multiparty computation
MAC	message authentication code
rand	random value

Κεφάλαιο 1

Το πρόβλημα της ανώνυμης επικοινωνίας.

1.1 Εισαγωγή

Η επικοινωνία έχει υπάρξει ανάγκη του ανθρώπου από την αρχή της ιστορίας του. Αρχικά, τα μυστικά τα οποία έπρεπε να μεταδοθούν από έναν άνθρωπο σε κάποιον άλλο μεταδίδονταν προφορικά. Καθώς ο πολιτισμός και η τεχνολογία εξελίχθηκε, η ανάγκη για ασφαλή μετάδοση μυστικών χωρίς να υπάρχει η ανάγκη για άμεση επαφή των δύο μερών έγινε ολοένα και πιο επιτακτική και τελικά γέννησε την κρυπτογραφία. Στο σύγχρονο κόσμο, παρόλα αυτά, μπορεί να είναι το ίδιο σημαντικό όχι μόνο να αποκρύπτεται το περιεχόμενο των μηνυμάτων που ανταλλάσσονται, αλλά και οι ταυτότητες των ατόμων που τα ανταλλάσσουν. Δηλαδή, η προστασία των μετα-δεδομένων κατά την επικοινωνία έχει γίνει πιο επιτακτική από ποτέ. Σε μια εποχή που σύμφωνα με αλλεπάλληλες αποκαλύψεις διενεργείται παρακολούθηση ευρείας κλίμακας στις επικοινωνίες, τόσο από κυβερνήσεις όσο και από επιχειρήσεις, η ικανότητα της επικοινωνίας με εγγυημένη ιδιωτικότητα είναι σοβαρή ανάγκη για χιλιάδες ανθρώπους. Το πρόβλημα της ασφαλούς επικοινωνίας (privacy-preserving communication) έχει μελετηθεί εκτενώς τα τελευταία χρόνια. Σε αυτό το κεφάλαιο θα ορίσουμε διάφορες εκδοχές του όρου ασφαλής επικοινωνία και θα δώσουμε ένα πιο τυπικό μοντέλο για το πρόβλημα.

1.2 Συνεισφορά

Η συνεισφορά αυτής της εργασίας είναι πολυεπίπεδη. Αρχικά, το πρόβλημα της ασφαλούς, με σεβασμό στην ιδιωτικότητα ανταλλαγής μηνυμάτων ορίζεται τυπικά μέσω ενός αλγεβρικού μοντέλου και παρουσιάζεται μια λύση αυτού του προβλήματος μέσω μιας ιδεατής λειτουργικότητας (ideal functionality). Δεύτερον, η έννοια της Διαφορικής Ιδιωτικότητας (Differential Privacy) [3] προσαρμόζεται στο πρόβλημα της ανώνυμης επικοινωνίας και μελετάται η πρακτικότητά της. Στη συνέχεια αναφέρουμε υπάρχοντες μηχανισμούς που επιλύουν το πρόβλημα, ο κάθε ένας σε συγκεκριμένο βαθμό και αποδοτικότητα. Εστιάζουμε στην ανάλυση του μηχανισμού Vuvuzela [2], ο οποίος παρουσιάζεται και σχολιάζεται η αποδοτικότητά του. Τελικά προτείνουμε μια διαφορετική προσέγγιση στο πρόβλημα μέσω τεχνικών Secure Multiparty Computation (SMC) (κεφάλαιο A.4) και μια λύση η οποία διαφέρει από τις μέχρι τώρα προτεινόμενες και μπορεί να οδηγήσει στη δημιουργία ενός αρκετά αποδοτικού συστήματος.

1.3 Ορίζοντας την ιδιωτικότητα

Στον προφορικό λόγο, οι λέξεις ιδιωτικότητα και ανωνυμία μπορεί να έχουν διάφορες ερμηνείες που εξαρτώνται από τις περιστάσεις της χρήσης τους. Στο πλαίσιο της επικοινωνίας, και συγκεκριμένα της ανταλλαγής μηνυμάτων, η ανωνυμία ορίζεται ως η κατάσταση στην οποία κάποιος δεν είναι αναγνωρίσιμος μέσα σε ένα σύνολο από υποκείμενα, το σύνολο ανωνυμίας [1]. Συνεχίζοντας με βάση αυτό τον ορισμό, μπορούμε να διακρίνουμε τις ακόλουθες περιπτώσεις ιδιωτικότητας-ανωνυμίας:

- *Ανωνυμία αποστολέα (sender anonymity)*. Ο αποστολές ενός δεδομένου μηνύματος δε μπορεί να αναγνωριστεί-διακριθεί ανάμεσα σε υποκείμενα ενός συνόλου πιθανών αποστολέων.

- *Ανωνυμία (από)δέκτη (receiver anonymity)*. Ο δέκτης ενός δεδομένου μηνύματος δε μπορεί να αναγνωριστεί-διακριθεί ανάμεσα σε υποκείμενα ενός συνόλου πιθανόν δεκτών.

- *Ανεξαρτησία συνδέσεων (session unlinkability)*. Η ανεξαρτησία συνδέσεων προστατεύει το χρήστη από έναν επιτιθέμενο που θέλει να συσχετίσει της ενέργειες του χρήστη καθώς προχωράει ο χρόνος.

- *Ιδιωτικότητα τοποθεσίας (location privacy)*. Η ιδιωτικότητα τοποθεσίας προστατεύει τη γεωγραφική θέση (διεύθυνση ip) του χρήστη.

Ένα ασφαλές σύστημα ανταλλαγής μηνυμάτων μπορεί να έχει κάποιες ή όλες από τις παραπάνω ιδιότητες, κάθε μία σε διαφορετικό βαθμό. Για παράδειγμα, κάποιο σύστημα μπορεί να προσφέρει απόλυτη ανωνυμία δέκτη και καθόλου ανωνυμία αποστολέα. Κάποιο άλλο σύστημα μπορεί να προσφέρει τόσο ανωνυμία δέκτη, όσο και αποστολέα, αλλά με πιθανότητα μικρότερη της μονάδας.

1.4 Τυπική παρουσίαση του προβλήματος

1.4.1 Ένα παράδειγμα.

Ας υποθέσουμε ότι δύο άτομα, η Αλίκη (A) και ο Βασίλης (B) θέλουν να ανταλλάξουν μηνύματα διατηρώντας την ιδιωτικότητα-ανωνυμία τους. Κρυπτογραφώντας τα δεδομένα χρησιμοποιώντας κάποιο προσυμφωνημένο κλειδί ή χρησιμοποιώντας τεχνικές κρυπτογραφίας δημοσίου κλειδιού, δεν είναι αρκετό. Οι δύο χρήστες θέλουν να κρύψουν το ίδιο το γεγονός ότι επικοινωνούν, ενάντια σε έναν αντίπαλο που μπορεί να επιτηρεί και να μεταχειρίζεται κάθε δεδομένο (πακέτο) στο δίκτυο. Με αυτή τη σκέψη, προχωράμε προς τη γενίκευση του προβλήματος.

1.4.2 Το μοντέλο.

Έστω D το σύνολο των n οντοτήτων (ατόμων, υπολογιστών, προγραμμάτων κτλ) που χρησιμοποιούν ένα πρωτόκολλο ανώνυμης επικοινωνίας. Κάποιοι από τους n αυτούς χρήστες μπορεί να θέλουν να επικοινωνήσουν μεταξύ τους σε κάποια δεδομένη στιγμή. Το πρωτόκολλο επικοινωνίας πρέπει να λειτουργεί σε διακριτούς γύρους ώστε να μπορεί να ανέχεται επιθέσεις ανάλυσης και χειραγώγησης της κίνησης του δικτύου.

Στο πλαίσιο αυτής της μελέτης, σε κάθε γύρο r, κάθε ένας από τους χρήστες u_i, $i \in \{1, \cdots, n\}$ θα εκτελέσει μία από τις ακόλουθες ενέργειες:

- Θα στείλει ένα μήνυμα σε κάποιον άλλο χρήστη u_j.

- Θα στείλει ένα ψεύτικο (fake) μήνυμα με τυχαίο αποδέκτη (πχ τον εαυτό του) σε περίπτωση που δε θέλει να επικοινωνήσει.

Ο λόγος για τον οποίο ένας χρήστης που δε θέλει να επικοινωνήσει με κάποιον άλλο θα πρέπει να στείλει ένα ψεύτικο μήνυμα που μοιάζει σαν αληθινό (dummy traffic), είναι ότι θέλει να προστατέψει την ιδιωτικότητα του όσον αφορά στο ότι επικοινωνεί ή όχι. Για τον ίδιο λόγο, κάθε χρήστης θα πρέπει να δέχεται ένα μήνυμα σε κάθε γύρο και σε κάθε περίπτωση.

αποστολέας	δέκτης
u_1	u_{j1}
u_2	u_{j2}
...	...
u_n	u_{jn}

Πίνακας 1.1: Ο πίνακας κατάστασης

1.4.3 Η κατάσταση του προβλήματος

Η κατάσταση του προβλήματος της ανώνυμης ανταλλαγής μηνυμάτων μπορεί να αναπαρασταθεί με τον παρακάτω πίνακα:

Στον παραπάνω πίνακα, ο χρήστης u_i, $i \in \{1, \ldots, n\}$ στέλνει ένα μήνυμα στο χρήστη u_{ji}, $ji \in \{1, \ldots, n\}$ και ένα μήνυμα από κάποιο χρήστη u_i προς τον εαυτό του αναπαριστά ένα ψεύτικο μήνυμα.

Η ίδια πληροφορία μπορεί να αναπαρασταθεί και από έναν δυαδικό πίνακα, όπως αυτός που ακολουθεί:

user_id	1	2	…	j	…	n
1	0	1	0	0	0	0
2	1	0	0	0	0	0
…						
i	0	0	0	1	0	0
…						
n	0	0	0	0	0	1

Πίνακας 1.2: ο δυαδικός πίνακας κατάστασης S

Στο παραπάνω παράδειγμα, ο χρήστης u_1 στέλνει ένα μήνυμα στον u_2, ο u_2 στέλνει ένα μήνυμα στον u_1 (οι δύο χρήστες βρίσκονται σε συζήτηση). Επιπλέον, ο χρήστης u_i στέλνει ένα μήνυμα στον u_j και ο u_n στέλνει ένα ψεύτικο μήνυμα.

Τα ακόλουθα θα πρέπει να ισχύουν για το δυαδικό πίνακα κατάστασης S:

(a) ο S έχει n γραμμές, μία για κάθε χρήστη.

(b) ο S έχει n στήλες, μία για κάθε χρήστη.

(c) ο S έχει ακριβώς ένα 1 σε κάθε μία από τις γραμμές του.

Το αν η επικοινωνία πραγματοποιείται μόνο όταν δύο χρήστες στέλνουν μηνύματα ο ένας στον άλλο σε κάποιο δεδομένο γύρο ή και στην περίπτωση που κάποιος χρήστης A στέλνει μήνυμα σε κάποιον άλλο χρήστη B αλλά ο B δεν στέλνει στον A εξαρτάται από την υλοποίηση της λύσης.

1.5 Προσεγγίζοντας τη λύση

1.5.1 Σκιαγράφηση της λύσης

Μία λύση του προβλήματος πρέπει να είναι ένας μηχανισμός που να λειτουργεί σε γύρους και να υλοποιεί με κάποιο τρόπο την ανώνυμη επικοινωνία μεταξύ των χρηστών του. Ιδανικά θα πρέπει να

μπορεί να εξυπηρετεί μεγάλο πλήθος χρηστών, τόσο για θέματα πρακτικότητας και επεκτασιμότητας, όσο και για να υπάρχει μεγάλο σύνολο ανωνυμίας (anonymity set). Μπορούμε να σκεφτούμε έναν τέτοιο μηχανισμό σαν ένα μαύρο κουτί που να δέχεται αιτήματα επικοινωνίας, ένα από κάθε χρήστη και υλοποιεί την ανταλλαγή τους.

1.5.2 Η Ιδεατή λειτουργικότητα μιας λύσης που χρησιμοποιεί σημεία-ραντεβού

Σε αυτό το μέρος, θα παρουσιάσουμε πώς το πρόβλημα της ανώνυμης ανταλλαγής μηνυμάτων μπορεί να λυθεί παρουσιάζοντας την ιδεατή λειτουργικότητα F. Μια ιδεατή λειτουργικότητα (ideal functionality) είναι ένα πρωτόκολλο στο οποίο μία τρίτη οντότητα (third party) στην οποία όλοι οι χρήστες έχουν εμπιστοσύνη και μπορούν να επικοινωνήσουν με αυτή μέσω ασφαλών συνδέσεων, υπολογίζει- υλοποιεί το επιθυμητό αποτέλεσμα. Είναι δηλαδή μια περιγραφή μιας λύσης σε έναν ιδανικό κόσμο και χρησιμεύει ως το σημείο αναφοράς κάθε πρωτοκόλλου- λύσης στον πραγματικό κόσμο.

Η ιδεατή λειτουργικότητα παρουσιάζεται στο σχήμα 1.1

Functionality F

round r

Running with a set of users $D = \{u_1, \ldots, u_n\}$ and an ideal adversary A, proceeds as follows:

- *Conversation:*
  Upon receiving:
  (a) (t_i, y_i) from all parties $u_i \in D$ (some may be controlled by A),
  (b) a list $L = \{u_k, \cdots, u_l\}$ of blocked users from A,
  compute the permutation π (as defined below) over the unblocked users and send message y_{π_i} to user i, while leaking information $f_1(t_i, y_i)$ to A.
- *Dialing:* Upon receiving $(DIAL, u_i, u_j)$ from user $u_i \in D$, compute a random value t_{u_i,u_j} and return in to user u_i, while leaking information f_2 to A.
- *Dial Check:* Upon receiving $(DIALCHECK)$ from user $u_j \in D$, return a list of (u_i, t_{u_i,u_j}) for any previous DIAL, while leaking information f_3 to A.

Σχήμα 1.1: Η ιδεατή λειτουργικότητα F.

Εξηγώντας τα σύμβολα:

- y_i : το μήνυμα το οποίο στέλνει ένας χρήστης u_i σε κάποιο άλλο χρήστη u_j, κρυπτογραφημένο με ένα κλειδί γνωστό στο δέκτη.
- t_i : Μία αριθμητική τιμή που συμβολίζει το σημείο του ραντεβού. Περιέχει δηλαδή πληροφορία για τον παραλήπτη του μηνύματος και μπορεί να είναι μία τιμή που δύο χρήστες οι οποίοι επιθυμούν να επικοινωνήσουν, έχουν συμφωνήσει από κοινού.
- A : Ο αντίπαλος γενικά θεωρούμε ότι έχει πλήρη έλεγχο του δικτύου και τη δυνατότητα να διαφθείρει κάποιους από τους παίκτες.
- π : μετάθεση π που χρησιμοποιείται παραπάνω είναι συγκεκριμένη και ορίζεται πάνω στο σύνολο των ζευγών (t_i, y_i) ως εξής:

– if $t_i = t_j \Rightarrow \begin{cases} \pi(i) = j \\ \pi(j) = i \end{cases}$

– else if $\forall j \neq i : t_i \neq t_j \Rightarrow \pi(i) = i$

Διαισθητικά, η μετάθεση π αναπαριστά την έννοια της ανταλλαγής μηνυμάτων.

Σχόλια πάνω στην F:
Το dialing μέρος, δηλαδή το μέρος της εκδήλωσης επιθυμίας επικοινωνίας ενός χρήστη προς έναν άλλο μπορεί να υλοποιείται με το ίδιο ή με διαφορετικό πρωτόκολλο σε σχέση με το conversation μέρος, το οποίο συμβολίζει την ανταλλαγή μηνυμάτων. Επίσης, το dial check μέρος υλοποιεί την ενημέρωση ενός χρήστη ότι κάποιος άλλος θέλει να εισέλθει σε συζήτηση μαζί του.

1.5.3 Αξιολογώντας μία λύση.

Ένα πρωτόκολλο- μηχανισμός που υλοποιεί την ιδεατή λειτουργικότητα F, μπορεί να αξιολογηθεί με βάση τα κάτωθι κριτήρια:

1. **Βαθμός ιδιωτικότητας (Degree of privacy)**: Σε σχέση με το ποσό της πληροφορίας που διαρρέει (leak) από το σύστημα, κάθε πρωτόκολλο πρέπει να εγγυάται ένα ποσοτικοποιήσιμο βαθμό ιδιωτικότητας.

2. **Κόστος επικοινωνίας (Communication Cost)**: Το κόστος επικοινωνίας ενός πρωτοκόλλου μπορεί να παρασταθεί από τον συνολικό αριθμό μηνυμάτων που στέλνονται από τις οντότητες που συμμετέχουν σε αυτό σε κάθε γύρο, καθώς και από το μέγεθος των εν λόγω μηνυμάτων.

3. **Υπολογιστικό κόστος (Computation Cost)**: Το συνολικό κόστος των υπολογισμών που απαιτούνται από το πρωτόκολλο.

4. **Πρακτικότητα (Practicality)**: Αφορά στο πραγματικό- οικονομικό κόστος των server και των υπόλοιπων υποδομών που απαιτούνται από το πρωτόκολλο.

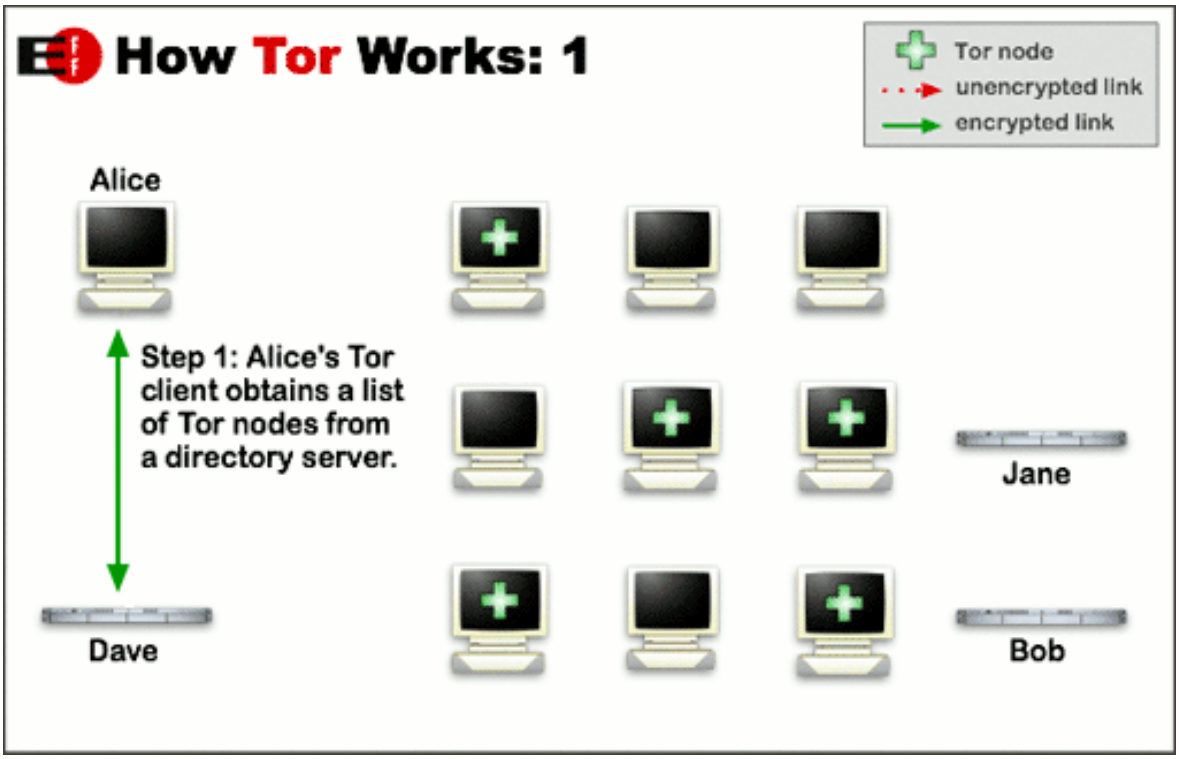
Κεφάλαιο 2

Υπάρχουσες λύσεις

2.1 Tor

Το δίκτυο Tor [28], είναι ένα σύνολο από εξυπηρετητές που υποστηρίζονται κυρίως από εθελοντές, και που επιτρέπει στο κοινό να βελτιώσει την ιδιωτικότητα και την ασφάλειά του στο διαδίκτυο. Το Tor είναι μακράν η πιο δημοφιλής λύση όσον αφορά την ασφαλή- ιδιωτική επικοινωνία για δημοσιογράφους, επιχειρήσεις και γενικά οποιοδήποτε επιθυμεί να κρατήσει την επικοινωνία του ιδιωτική. Σε ένα βασικό επίπεδο, το Tor προστατεύει την ιδιωτικότητα των χρηστών δρομολογώντας κάθε πακέτο που εξέρχεται από τον υπολογιστή ενός χρήστη μέσα από τρεις μεσολαβητές (relays) στο δρόμο προς τον προορισμό του. Περισσότερα για τη λειτουργία του Tor, βασιζόμενα στο υλικό από το επίσημο site του [29], όπως επίσης και συζήτηση για τους περιορισμούς και τις αδυναμίες του, ακολουθεί στο υπόλοιπου αυτού του μέρους.

2.1.1 Πώς λειτουργεί το Tor



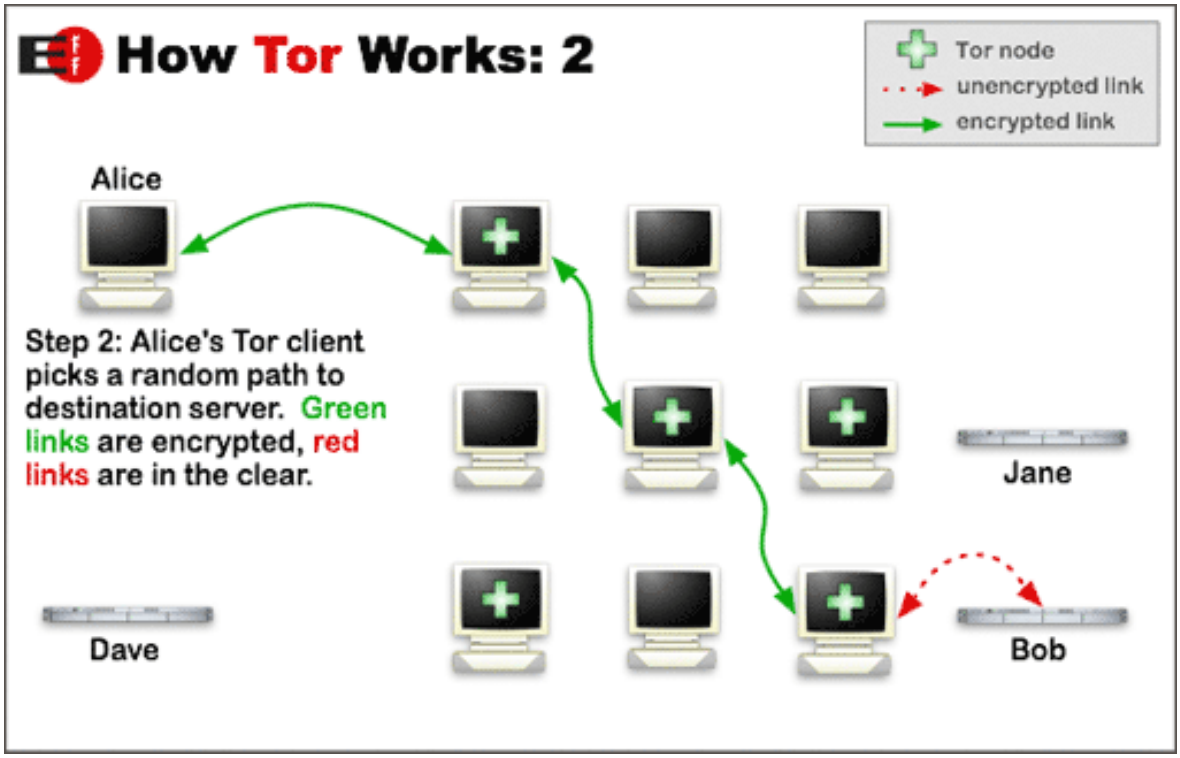
Σχήμα 2.1: How Tor works (1)

Το Tor προσπαθεί να κρύψει τη σχέση μεταξύ δύο μερών που θέλουν να επικοινωνήσουν δρο-

μολογώντας την κίνηση (πακέτα) μέσα από ένα δίκτυο αποτελούμενο από (συνήθως) τρεις τυχαίους εξυπηρετητές, οι οποίοι γνωρίζουν μόνο την ταυτότητα του προηγούμενου και του επόμενου από αυτούς σταθμό. Με αυτό τον τρόπο, ένας επιτιθέμενος, οποίος παρατηρεί τα πακέτα που εξέρχονται ή εισέρχονται στον υπολογιστή ενός χρήστη, ενώ ο εν λόγω χρήστης χρησιμοποιεί το δίκτυο Tor, δεν μπορεί να βγάλει κανένα άλλο συμπέρασμα για την επικοινωνία του χρήστη εκτός από το ότι όντως χρησιμοποιεί το Tor. Αυτό επιτυγχάνεται με επαναλαμβανόμενη κρυπτογράφηση της διεύθυνσης (ip) του παραλήπτη ενός μηνύματος, μαζί βέβαια με το σώμα του μηνύματος (πακέτου). Αυτή η επαναλαμβανόμενη κρυπτογράφηση που θυμίζει τα στρώματα ενός κρεμμυδιού, ονομάζεται και “onion”, κάτι το οποίο έδωσε το όνομά και του Tor, που είναι ακρωνύμιο για το The onion router.

Το πρωτόκολλο λειτουργεί όπως περιγράφεται ακολούθως:

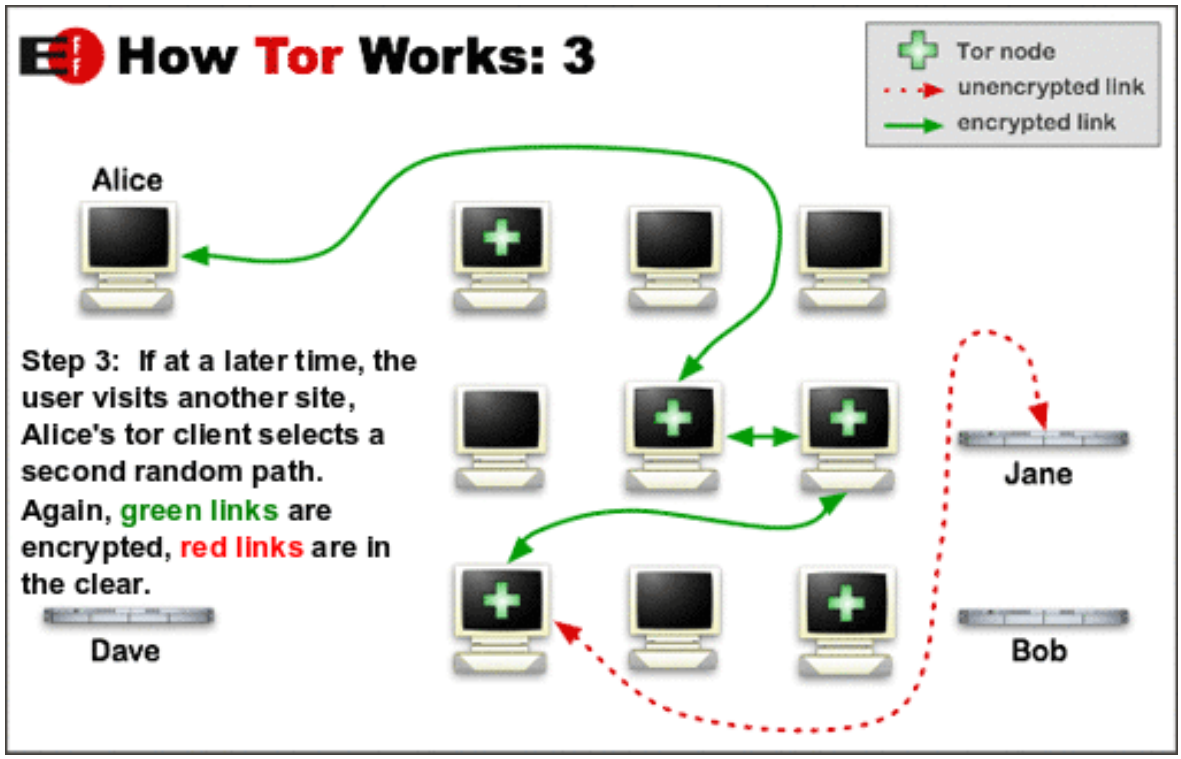
Αρχικά, ο Tor client ενός χρήστη (Alice), κατεβάζει από μια έμπιστη πηγή στο διαδίκτυο τη λίστα με τους διαθέσιμους Tor servers, όπως φαίνεται στο σχήμα 2.1. Αυτοί είναι όλοι πιθανοί διαμεσολαβητές για τη δημιουργία ενός κυκλώματος επικοινωνίας και διατηρούνται σε μεγάλη πλειοψηφία από εθελοντές. Για τη δημιουργία ενός κυκλώματος ιδιωτικοποίησης με το Tor, ο client του χρήστη βήμα βήμα δημιουργεί ένα κύκλωμα από κρυπτογραφημένες συνδέσεις μέσα από διαμεσολαβητές στο δίκτυο. Το κύκλωμα αυξάνεται ένα βήμα (hop) κάθε φορά, και κάθε διαμεσολαβητής γνωρίζει μόνο ποιος διαμεσολαβητής του δίνει δεδομένα, και σε ποιον πρέπει να τα προωθήσει. Κανένας μεμονωμένος server δεν γνωρίζει το ολοκληρωμένο μονοπάτι που έχει ακολουθήσει κάποιο πακέτο. Ο client του χρήστη συμφωνεί ένα διαφορετικό σετ από κλειδιά κρυπτογράφησης με κάθε ενδιάμεσο server του κυκλώματος- μονοπατιού, για να διασφαλίσει ότι κανένας διαμεσολαβητής δε θα μπορέσει να ακολουθήσει τα πακέτα καθώς αυτά περνάνε από το κύκλωμα. Αυτά φαίνονται στο σχήμα 2.2. Αφού ένα κύκλωμα έχει δημιουργηθεί, οποιοδήποτε είδος πληροφορίας μπορεί να ανταλλαχθεί



Σχήμα 2.2: How Tor works (2)

ανώνυμα, και μια πληθώρα εφαρμογών μπορούν να δρομολογήσουν την κίνησή τους μέσα από το δίκτυο του Tor. Επειδή κάθε ενδιάμεσος server έχει εποπτεία μόνο για ένα βήμα του κυκλώματος,

ένας κακοπροαίρετος τέτοιος server, πόσο μάλλον ένας απλός παρατηρητής δε μπορεί να βγάλει συμπεράσματα για τον αποστολέα και τον παραλήπτη ενός πακέτου. Για αυξημένη αποδοτικότητα, το λογισμικό του Tor χρησιμοποιεί το ίδιο κύκλωμα για συνδέσεις που συμβαίνουν στο ίδιο χρονικό παράθυρο περίπου δέκα λεπτών. Επόμενες αιτήσεις για σύνδεση δημιουργούν καινούργιο κύκλωμα, ώστε να αποτρέψουν τη σύνδεση των πράξεων του χρήστη με την πάροδο του χρόνου, όπως φαίνεται στο σχήμα 2.3.



Σχήμα 2.3: How Tor works (3)

2.1.2 Περιορισμοί του Tor

Το Tor προστατεύει την ταυτότητα του χρήστη απέναντι σε έναν αντίπαλο που έχει περιορισμένη εποπτεία του δικτύου. Ένας τέτοιος αντίπαλος, μπορεί για παράδειγμα μόνο να επιτηρεί τα πακέτα στο άκρο του δικτύου που αντιστοιχεί στον χρήστη. Εδώ και αρκετά χρόνια έχει υπάρξει αρκετή έρευνα σχετικά με πιθανές επιθέσεις στο Tor που επιτρέπουν σε έναν τέτοιο αντίπαλο να λάβει πληροφορία που υποβαθμίζει την ιδιωτικότητα του χρήστη [40]. Παρόλα αυτά δε θα εστιάσουμε σε τέτοιες επιθέσεις, αλλά θα εντοπίσουμε την de facto αδυναμία του πρωτοκόλλου του Tor, που είναι η έλλειψη ασφάλειας στην περίπτωση που ένας αντίπαλος έχει κάποια υποψία ότι δυο χρήστες μπορεί να επικοινωνούν και επίσης έχει τη δυνατότητα να παρατηρεί να πακέτα και στα δύο άκρα του δικτύου (global adversary). Ένας τέτοιος αντίπαλος, μπορεί εύκολα να συσχετίσει τα πακέτα που εξέρχονται από τον ένα χρήστη με αυτά που εισέρχονται στον υπολογιστή ενός άλλου, και εύκολα να φτάσει σε ασφαλές συμπέρασμα για τον εάν αυτοί οι δύο χρήστες επικοινωνούν μεταξύ τους. Αυτή η επίθεση ονομάζεται και *end-to-end correlation* και είναι εγγενής αδυναμία του Tor. Η μελέτη συστημάτων που μπορούν να αντιμετωπίσουν τέτοιου είδους επιθέσεις είναι ουσιαστικά το θέμα αυτής της εργασίας. Παρά τις αδυναμίες του, το Tor παραμένει πάντως η πιο ασφαλής επιλογή που είναι διαθέσιμη αυτή τη στιγμή και επίσης προσφέρει πολύ χαμηλή καθυστέρηση (latency) στα πακέτα συγκριτικά με άλλες λύσεις.

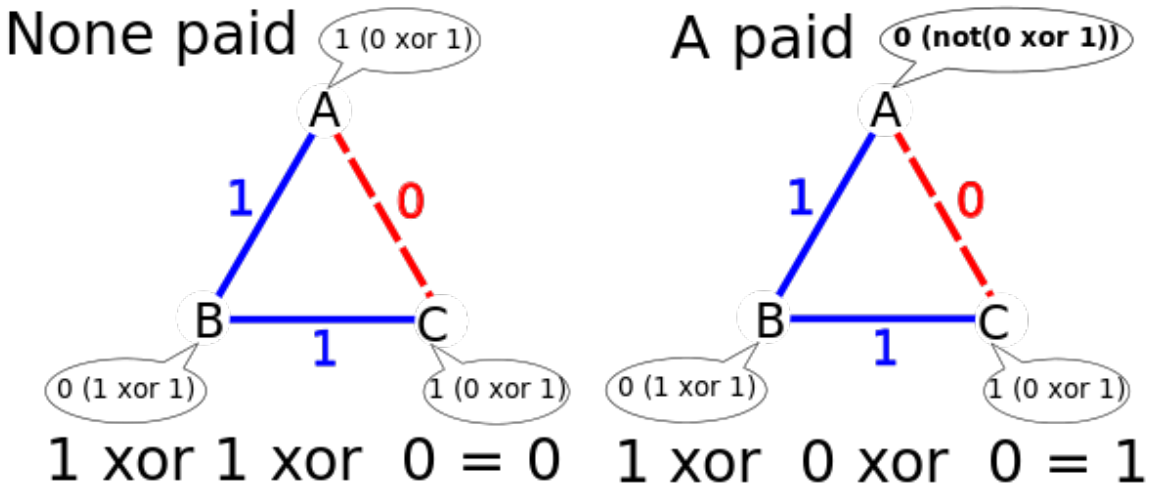
2.2 DC networks

Τα DC-networks (dc-nets) ή Dining Cryptographers networks, χρησιμοποιούν τεχνικές του secure multiparty computation ώστε να παράσχουν ένα τρόπο για ασφαλή επικοινωνία, ανθεκτική ενάντια σε οποιεσδήποτε επιθέσεις συσχέτισης της κίνησης του δικτύου. Ανάμεσα στα συστήματα που χρησιμοποιούν αρχές των DC-nets για να προσφέρουν ιδιωτικότητα, είναι υπηρεσίες ανταλλαγής μηνυμάτων όπως το Herbivore [33] και το Dissent [32]. Τα δίκτυα DC βασίζονται σε πρωτόκολλα όπως αυτό που παρουσιάζεται παρακάτω:

Τρεις κρυπτογράφοι μαζεύονται γύρω από ένα τραπέζι για δείπνο. Στο τέλος της βραδιάς, ο σερβιτόρος τους ενημερώνει ότι ο λογαριασμός έχει πληρωθεί από κάποιον, ο οποίος μπορεί να είναι είτε ένας από τους κρυπτογράφους ή η ΕΥΠ. Οι τρεις κρυπτογράφοι σέβονται το δικαίωμα του καθενός να πληρώσει ανώνυμα για το γεύμα αλλά θέλουν να διαπιστώσουν εάν το γεύμα έχει πληρωθεί από την ΕΥΠ. Για να το κατορθώσουν αυτό, ακολουθούν το εξής πρωτόκολλο δύο σταδίων. Αρχικά, κάθε κρυπτογράφος συμφωνεί σε ένα τυχαίο bit με κάθε έναν από τους άλλους δύο κρυπτογράφους, πιθανόν στρίβοντας ένα νόμισμα πίσω από τους καταλόγους. Τελικά οι κρυπτογράφοι A και B μοιράζονται το τυχαίο bit b_{AB}, οι B και C το b_{BC}, και οι A και C μοιράζονται το b_{AC}. Στο δεύτερο μέρος του πρωτοκόλλου, κάθε κρυπτογράφος ανακοινώνει μια τιμή ως εξής:

- Το XOR των bits που μοιράζεται με τους άλλους δύο κρυπτογράφους εάν δεν πλήρωσε αυτός για το γεύμα, ή
- Το αντίθετο από το XOR που αναφέραμε πριν, εάν αυτός όντως πλήρωσε για το γεύμα

Για να αποκαλυφθεί η αλήθεια, κάποιος απλά πρέπει να υπολογίσει το XOR των τριών τιμών που ανακοινώθηκαν. Εάν αυτή η τιμή υπολογιστεί ίση με το 0, τότε η ΕΥΠ πλήρωσε για το γεύμα, σε αντίθετη περίπτωση κάποιος από τους κρυπτογράφους πλήρωσε. Η ταυτότητα του συγκεκριμένου κρυπτογράφου παραμένει άγνωστη σε αυτή την περίπτωση. Ένα παράδειγμα εκτέλεσης αυτού του πρωτοκόλλου φαίνεται στο σχήμα 2.4. Βέβαια, αυτό είναι ένα πολύ απλοϊκό πρωτόκολλο, το οποίο



Σχήμα 2.4: Βασικό dc-net πρωτόκολλο

έχει πολλούς περιορισμούς. Για παράδειγμα, αν δύο κρυπτογράφοι έχουν πληρώσει για το γεύμα, το αποτέλεσμα θα είναι πάλι 0, κάτι το οποίο θα οδηγεί στο λάθος συμπέρασμα. Αυτό δείχνει ότι το πρωτόκολλο επιτρέπει μόνο σε ένα συμμετέχοντα να εκπέμψει κάποιο μήνυμα. Αυτοί οι περιορισμοί και

οι αδυναμίες αντιμετωπίζονται με διάφορες τεχνικές και γενικεύσεις του πρωτοκόλλου αυτού έχουν βρει χρήση στα συστήματα που αναφέρθηκαν παραπάνω. Εδώ είναι σημαντικό να σημειωθεί ότι γενικά αυτά τα συστήματα απαιτούν επικοινωνία μεταξύ κάθε ζευγαριού χρηστών και ως αποτέλεσμα αυτού, δεν κλιμακώνουν καλά όσο αυξάνεται ο αριθμός των χρηστών. Μπορούν δηλαδή να εξυπηρετήσουν δεκάδες, εκατοντάδες ή μέχρι και μερικές χιλιάδες χρήστες. Από την άλλη, χρησιμοποιώντας κρυπτογραφικές τεχνικές, αυτά τα δίκτυα προσφέρουν πολύ καλή ασφάλεια απέναντι σε κάθε είδους αντίπαλο.

2.3 Το σύστημα Vuvuzela

Μία άλλη προσέγγιση για την επίτευξη ισχυρής ανωνυμίας και ασφάλειας ενάντια σε επιθέσεις ανάλυσης της κίνησης του δικτύου είναι η χρήση mixnets (παράρτημα Α.3). Ένα πολύ ενδιαφέρον και πρόσφατο σύστημα που λύνει το πρόβλημα χρησιμοποιώντας mixnets είναι το Vuvuzela [2] των van den Hooff, Lazar,Zaharia και Zeldovich. Το Vuvuzela είναι ένα σύστημα που χρησιμοποιεί mixnets και κίνηση κάλυψης (cover traffic) για να επιτύχει διαφορική ιδιωτικότητα (παράρτημα Β) απέναντι σε έναν ισχυρό αντίπαλο που έχει πλήρη εποπτεία του δικτύου, μπορεί επίσης να διαγράψει ή να καθυστερήσει μηνύματα της επιλογής του και έχει υπό την επήρειά του όλους τους servers του συστήματος εκτός από έναν και όλους τους άλλους χρήστες εκτός από τους δύο που θέλουν να επικοινωνήσουν. Αυτό το σύστημα είναι η έμπνευση για τη γενικότερη ιδέα της ιδεατής λειτουργικότητας F του σχήματος 1.1. Χρησιμοποιεί σημεία ραντεβού που αποκαλούνται dead drops και λειτουργεί σε δύο διακριτά πρωτόκολλα, το πρωτόκολλο συζήτησης (conversation protocol) και το πρωτόκολλο κλήσης (dialing protocol). Για κάθε πρωτόκολλο, η ασφάλεια- ανωνυμία επιτυγχάνεται μειώνοντας όσο το δυνατόν την πληροφορία που διαρρέει, και προσθέτοντας εικονικά- ψεύτικα μηνύματα ως θόρυβο για να αποκρύψουν το μικρό ποσό της πληροφορίας που διέρρευσε Ο μηχανισμός αποτελείται στην υλοποίησή του από μια σειρά από mix servers (2 ή 3 στα παραδείγματά μας). Σε αυτό το κεφάλαιο θα παρουσιάσουμε τη σχεδίαση του Vuvuzela, θα δούμε παραδείγματα της εκτέλεσής του και θα εξηγήσουμε πώς και σε ποιο βαθμό επιτυγχάνει την ιδιωτικότητα για τους χρήστες του.

2.3.1 Υποθέσεις

Πριν ξεκινήσουμε την ανάλυση του συστήματος, πρέπει να γίνουν σαφείς οι υποθέσεις των συγγραφέων. Συγκεκριμένα:

1. Τα δημόσια κλειδιά των server και οι σειρά τους στην αλυσίδα είναι γνωστά σε όλους τους χρήστες.

2. Δύο χρήστες που θέλουν να επικοινωνήσουν γνωρίζουν ο καθένας το δημόσιο κλειδί του άλλου (δεδομένο PKI).

3. Κρυπτογραφικά ασφαλής κρυπτογράφηση δημοσίου και συμμετρικού κλειδιού, μηχανισμοί ανταλλαγής κλειδιών, ψηφιακές υπογραφές και συναρτήσεις σύνοψης (hash functions).

4. Οι τίμιου clients και servers εκτελούν μια υλοποίηση χωρίς bugs.

5. Οι χρήστες μπορεί να συμπεριφέρονται όπως θέλουν, αλλά οι servers θα διαχειρίζονται τα αιτήματα με σωστό τρόπο ώστε να αποφεύγονται denial of service επιθέσεις.

2.3.2 Πρωτόκολλο κλήσης (Dialing Protocol)

Αυτό το πρωτόκολλο υλοποιεί τις λειτουργίες *dialing* και *dial check* της ιδεατής λειτουργικότητας F του σχήματος 1.1. Ο στόχος του πρωτοκόλλου είναι να επιτρέπει στο χρήστη u_i να ειδοποιήσει το

χρήστη u_j ότι θέλει να αρχίσει μια συνομιλία μεταξύ τους. Ως ιδέα το πρωτόκολλο έχει πολλά κοινά με τη διενέργεια ενός τηλεφωνήματος, όπου ο ένας χρήστης επανειλημμένα καλεί έναν άλλο, μέχρι ο δεύτερος να αποφασίσει να μπει σε συνομιλία μαζί του. Εδώ ο ένας χρήστης στέλνει αιτήματα και περιμένει προσπαθώντας να επικοινωνήσει μέχρι σε κάποιο γύρο να λάβει απάντηση. Οι συμβολισμοί σε αυτό το κεφάλαιο είναι όπως παρουσιάζονται στον πίνακα συμβολισμών στην αρχή της παρούσας εργασίας. Σημειώστε επίσης ότι ένα (u) δίπλα στον αριθμό του βήματος υποδηλώνει ενέργεια κάποιου χρήστη ενώ ένα (s), ενέργεια κάποιου server, καθώς επίσης και ότι ο αριθμός των servers είναι n.

Dialing Protocol round r.

1. (s) Vuvuzela servers agree on a number m of invitation dead drops.
2. (u) If a user u_i wants to start a conversation with user u_j, she generates:

   - $e_{n+1} = (b, Enc(pk_{u_j}, M))$
   - $M = (pk_{u_i}, nonce, MAC)$
   - $b = H(pk_{u_j}) \, mod \, m$

   If a user doesn't want to start a conversation, she generates e_{n+1} as above with:
   $pk_{u_j} = rand$
   $b = 0$ (special trash dead drop)
3. (u) Onion wrap the request and send.

   - Generate temporary (for one round) key-pair $(pk_k^{u_i}, sk_k^{u_i})$ for each server.
   - $s_k = DH(sk_k^{u_i}, pk_{server_k})$
   - $e_k = (pk_k^{u_i}, Enc(s_k, e_{k+1}))$

   e_1 is the message sent to the first server.
4. (s) Collect and decrypt requests.
   For each request:

   - compute all $s_k = DH(sk_{server_k}, pk_k^{u_i})$
   - $e_{k+1} = Dec(s_k, e_k)$
5. (s) Generate cover traffic.
   Add $n = \max(0, \lceil Laplace(\mu, b) \rceil)$ fake invitations to each invitation dead drop.
6. (s) Shuffle requests with a random permutation π.
7. (s*) Server n places invitations in the corresponding dead drops.
8. (u) User u_j downloads dead drop $H(pk_{u_j}) \, mod \, m$.
9. (u) If u_j wants to accept the invitation from u_i, she calculates the shared secret $s_{n+1}^{u_j} = DH(sk_{u_j}, pk_{u_i})$ and proceeds with the conversation protocol.

Εδώ σημειώνεται ότι τα βήματα 4,5,6 εκτελούνται από κάθε server με τη σειρά του. Αντίθετα, το βήμα 7 εκτελείται μόνο από τον τελευταίο server της αλυσίδας.

2.3.3 Πρωτόκολλο συνομιλίας (conversation protocol)

Αυτό το πρωτόκολλο υλοποιεί τη λειτουργία *conversation* της ιδεατής λειτουργικότητας F, και ουσιαστικά είναι αυτό που εκτελεί την ανταλλαγή των μηνυμάτων μεταξύ δύο χρηστών που έχουν συμφωνήσει να επικοινωνήσουν. Το πρωτόκολλο περιγράφεται παρακάτω:

Conversation Protocol round r

1. (u) If u_i wants to send a message to u_j:

   - $s_{n+1}^{u_i} = DH(sk_{u_i}, pk_{u_j})$
   - $b = H(s_{n+1}^{u_i}, r)$
   - $e_{n+1} = (b, Enc(s_{n+1}^{u_i}, m))$

   If u_i is idle, proceed as above with $pk_{u_j} = rand$ and m: zero or random message.

2. (u) Onion wrap the request and send it to the first server.

   - Generate temporary (for one round) key-pair $(pk_k^{u_i}, sk_k^{u_i})$ for each server.
   - $s_k = DH(sk_k^{u_i}, pk_{server_k})$
   - $e_k = (pk_k^{u_i}, Enc(s_k, e_{k+1}))$

   e_1 is the message sent to the first server.

3. (s) Collect and decrypt requests. (as in dialing).

4. (s) Generate cover traffic.
   Sample n_1 and n_2 from $Laplace(\mu, b)$ and add $\lceil n_1 \rceil$ individual requests to random dead drops and $\lceil \frac{n_2}{2} \rceil$ pairs of requests to random dead drops.

5. (s) Shuffle requests with a random permutation π.

6. (s*) Server n (the last server) exchanges the contents of the requests that access the same dead drop.

7. (s) Encrypt.
   $e'_k = Enc(s_k, e'_{k+1})$

8. (s) Apply inverse permutation π^{-1} and return to the previous server (first server returns message to the specified user).

9. (u) Receive result and unwrap.
   $e'_{k+1} = Dec(s_k, e'_k)$
   …
   e'_{n+1} is the received message.

Εδώ πρέπει να σημειωθεί ότι το βήμα 6 εκτελείτε μόνο από τον τελευταίο server, ενώ όλα τα υπόλοιπα βήματα που αντιστοιχούν σε ενέργειες των server εκτελούνται από όλους τους servers με τη σειρά της αλυσίδας.

2.3.4 Ένα παράδειγμα του πρωτοκόλλου συζήτησης

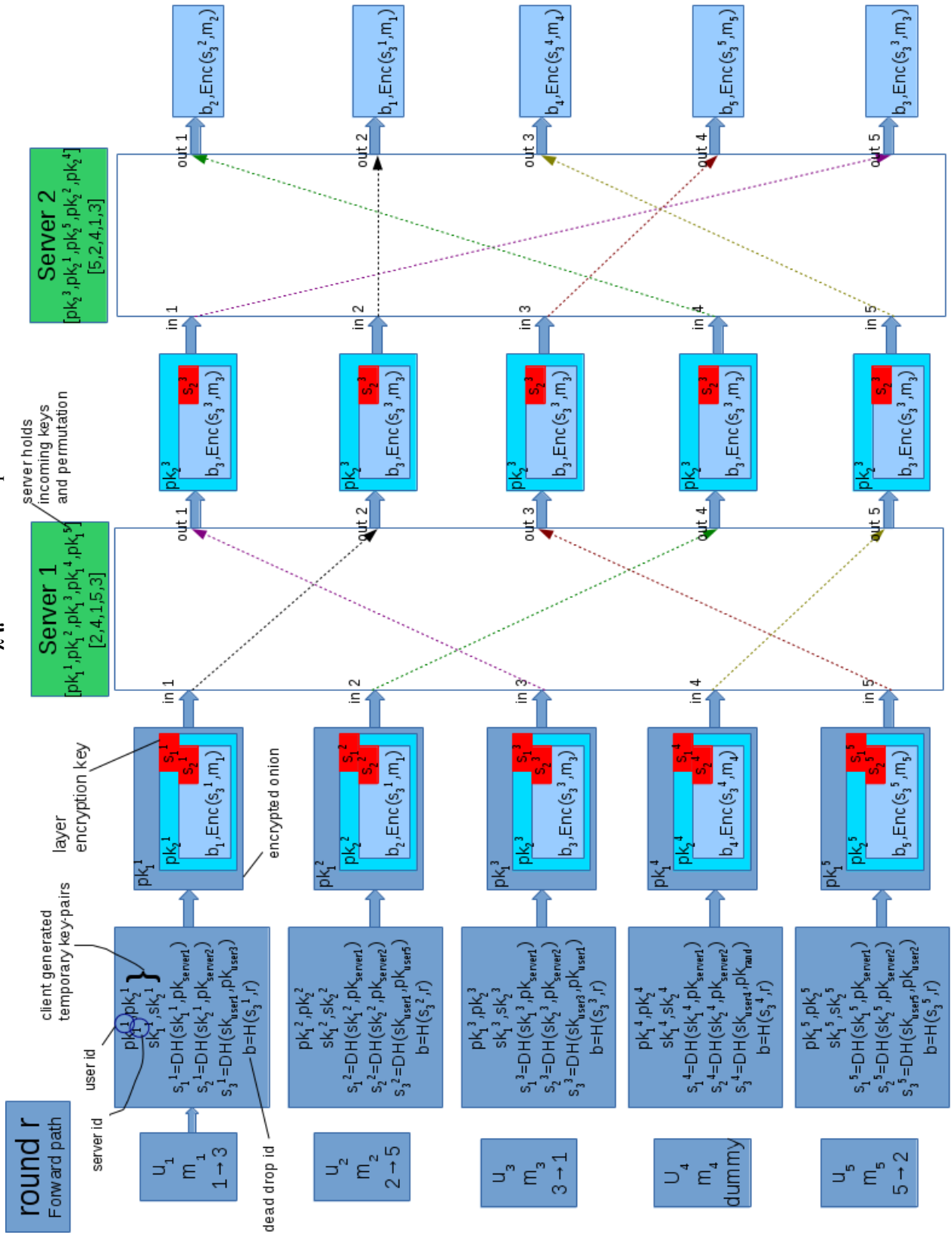
Στα σχήματα που ακολουθούν, παρουσιάζουμε ένα παράδειγμα εκτέλεσης του πρωτοκόλλου συνομιλίας του Vuvuzela. Στο συγκεκριμένο παράδειγμα έχουμε 5 χρήστες και 2 mix servers. Σημειώνουμε ότι αγνοούμε προς το παρόν το θόρυβο που προστίθεται από κάθε server ώστε να επιτευχθεί η διαφορική ιδιωτικότητα και επικεντρωνόμαστε στον τρόπο ανταλλαγής των μηνυμάτων. Ο θόρυβος, που πρόκειται για αιτήματα- μηνύματα με τυχαίο παραλήπτη αλλά αδιάκριτα από τα υπόλοιπα αληθινά μηνύματα, απλά ανακατεύεται μαζί με τα αληθινά μηνύματα στην forward διαδρομή και απορρίπτεται από τον server ο οποίος τον προσέθεσε στη return διαδρομή. Στο παράδειγμά μας, οι χρήστες u_1 και u_3, καθώς και οι u_2 και u_5 είναι σε συζήτηση ανά ζευγάρια. Αντίθετα, ο χρήστης u_4 δε βρίσκεται σε κάποια ενεργή συνομιλία. Κάθε χρήστης δημιουργεί αιτήματα συνομιλίας, τα κρυπτογραφεί σε onion και τα προωθεί στον πρώτο server της αλυσίδας. Κάθε server μετά με τη σειρά του, αφαιρεί ένα στρώμα κρυπτογράφησης από το onion, ανακατεύει τα μηνύματα- αιτήματα και τα προωθεί στον επόμενο server. Ο τελευταίος server αφαιρεί το τελευταίο στρώμα του onion που αποκαλύπτει το dead drop του κάθε αιτήματος και ανταλλάσσει (swap) τα περιεχόμενα των αιτημάτων που αντιστοιχούν στο ίδιο dead drop. Στη συνέχεια, κάθε server αρχίζοντας από τον τελευταίο και ως τον πρώτο, προσθέτει ένα στρώμα κρυπτογράφησης, εφαρμόζει την αντίστροφη μεταλλαγή (permutation) του ανακατέματος που έκανε στο forward μονοπάτι, και προωθεί το αίτημα στον επόμενο server. Έτσι τα αιτήματα επιστρέφουν στους χρήστες που τα έστειλα από τον ίδιο δρόμο από τον οποίο έφτασαν στον τελευταίο server, με τη διαφορά ότι τα περιεχόμενά τους έχουν αλλάξει. Τελικά οι χρήστες αφαιρούν όλα τα στρώματα της κρυπτογράφησης από τα αιτήματα που λαμβάνουν και έτσι ανακτούν το μήνυμα που προοριζόταν για αυτούς. Πρέπει να σημειωθεί ότι και το dialing πρωτόκολλο λειτουργεί με παρόμοιο τρόπο, μόνο που δε χρειάζεται την επιστροφή των αιτημάτων και έτσι αξιοποιεί μόνο το forward path, καθώς τα αιτήματα του dialing κατεβάζονται όλα μαζί από τους χρήστες.

2.3.5 Ιδιωτικότητα στο Vuvuzela

Το Vuvuzela επιτυγχάνει την ιδιωτικότητα μέσα από ένα συνδυασμό από πρωτόκολλα σταθερού bandwidth, mixnets και cover traffic. Οι χρήστες στέλνουν (και λαμβάνουν) ένα σταθερό αριθμό κρυπτογραφημένων αιτημάτων- μηνυμάτων σταθερού μεγέθους σε κάθε γύρο και έτσι είναι αδύνατο για έναν αντίπαλο να βγάλει συμπεράσματα παρατηρώντας την κίνηση του δικτύου. Τα mixnets, μαζί με το γεγονός ότι τα id's των dead drops (σημείων συνάντησης) είναι ψευδοτυχαία, χρησιμοποιούνται για να διαλύσουν τη σύνδεση μεταξύ του χρήστη και του μηνύματός του αρκεί να υπάρχει ένας τίμιος server στην αλυσίδα. Η μόνη παράμετρος που διαρρέει από το σύστημα, και αυτό μόνο στην περίπτωση που ο τελευταίος server ελέγχεται από τον αντίπαλο, είναι ο αριθμός των dead drops με μία πρόσβαση (m_1) και ο αριθμός των dead drop με δύο προσβάσεις (m_2). Με άλλα λόγια, η μόνη μεταβλητή που διαρρέει είναι ο αριθμός των χρηστών που βρίσκονται σε κάποια συνομιλία. Επειδή αυτό μπορεί να έχει ως αποτέλεσμα την παραβίαση της ιδιωτικότητας των χρηστών, απαιτείται η προσθήκη θορύβου, δηλαδή ψεύτικων μηνυμάτων ώστε να αποκρύπτεται η ακριβής τιμή των δύο μεταβλητών. Κάθε ένας από τους servers προσθέτει έναν αριθμό από απλά μηνύματα σε κάποια τυχαία dead drops και έναν αριθμό από διπλά μηνύματα σε κάποια τυχαία dead drops. Στην ανάλυση που ακολουθεί θα εστιάσουμε στο πρωτόκολλο συζήτησης, αλλά τα ίδια περίπου ισχύουν και για το πρωτόκολλο κλήσης.

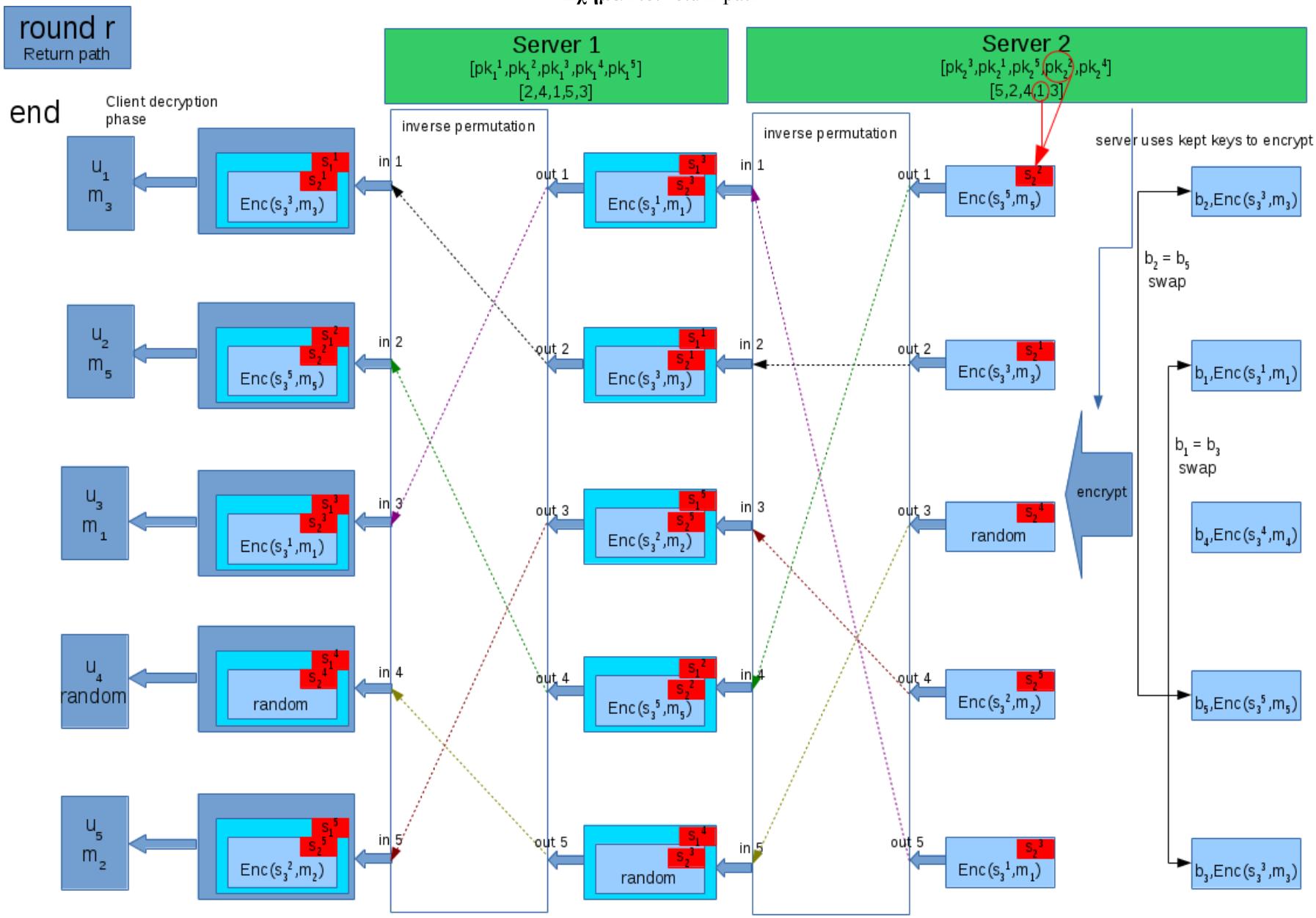
Ας θυμηθούμε, όμως πρώτα τον δυαδικό πίνακα κατάστασης 1.2:

Σε αυτό το μοντέλο, δύο σενάρια είναι γειτονικά (adjacent) εάν διαφέρουν μόνο στις πράξεις το πολύ ενός χρήστη, δηλαδή εάν οι πίνακες που τα περιγράφουν διαφέρουν σε μια μοναδική γραμμή. Είναι εύκολο να ορίσουμε τις μεταβλητές m_1 και m_2 ως συναρτήσεις πάνω σε πίνακες όπως ο 2.1. Συγκεκριμένα έχουμε:



Σχήμα 2.5: forward path

Σχήμα 2.6: return path



user_id	1	2	…	j	…	n
1	0	1	0	0	0	0
2	1	0	0	0	0	0
…						
i	0	0	0	1	0	0
…						
n	1	0	0	0	0	1

Πίνακας 2.1: the binary state matrix S

$$m_2 = \sum_{\substack{i=1\ldots n \\ j=i+1\ldots n}} |A[i,j] \cdot A[j,i]|$$

$$m_1 = n - 2m_2$$

Έτσι η συνάρτηση που αντιπροσωπεύει το μοναδικό query που μπορεί να κάνει ένας αντίπαλος στον παραπάνω πίνακα- βάση είναι η εξής: $f : S \to \mathbb{R}^2$, and for a given $x \in S$: $f(x) = (m_1, m_2)$.

Remark 2.1. Εδώ θα αποκλίνουμε λίγο από τον τρόπο ανάλυσης της ιδιωτικότητας στο paper του συστήματος [2] και θα παρουσιάσουμε μια πιο γενική προσέγγιση. Δηλαδή ενώ οι συγγραφείς θεωρούν τις δύο μεταβλητές ως ανεξάρτητες συναρτήσεις, εμείς θα τις θεωρήσουμε ως μία συνάρτηση με πεδίο τιμών στο $\mathbb{R}^2$. Η διαφορά που προκύπτει είναι μικρή σε σχέση με την ποσότητα θορύβου που απαιτείται και θα συγκρίνουμε τις δύο προσεγγίσεις στη συνέχεια του κεφαλαίου.

Είναι άμεση συνέπεια του παραπάνω ορισμού και του ορισμού B.5, ότι η ℓ_1-sensitivity της παραπάνω συνάρτησης είναι:

$$\Delta(f) = 3$$

Αυτό προέρχεται από την παρατήρηση ότι η αλλαγή στη συμπεριφορά ενός μόνο χρήστη μπορεί να αλλάξει το m_2 το πολύ κατά 1 και το m_1 κατά 2. Για παράδειγμα, αυτή η διαφορά παρατηρείται ανάμεσα σε ένα σενάριο στο οποίο ένας χρήστης είναι σε συζήτηση με κάποιον άλλο χρήστη και σε ένα σενάριο στο οποίο ο ίδιος χρήστης δε συμμετέχει σε κάποια συζήτηση. Στο δεύτερο σενάριο, σε σχέση με το πρώτο, ο αριθμός των dead drops με δύο προσβάσεις (m_2 μειώνεται κατά ένα και αντίστοιχα ο αριθμός των dead drops με μία πρόσβαση (m_1) αυξάνεται κατά δύο. Άρα η συνολική διαφορά στην ℓ_1 νόρμα είναι 3. Έτσι, είναι εύκολο να συμπεράνουμε σύμφωνα με το θεώρημα 2, ότι

προσθέτοντας θόρυβο από τη δειγματοληψία μιας κατανομής Laplace με παραμέτρους $b = \frac{3}{\varepsilon}$ σε κάθε μία από τις δύο μεταβλητές, μπορούμε να επιτύχουμε (ε, δ)-διαφορική ιδιωτικότητα. Βέβαια, τα μηνύματα έχουνε διακριτή φύση και έτσι στην πραγματικότητα κάθε server θα προσθέσει $\lceil Lap(\mu, b) \rceil$ σε κάθε μία από τις δύο μεταβλητές. Αυτό δεν επηρεάζει την ιδιωτικότητα, καθώς όπως ξέρουμε από τη θεωρία της διαφορικής ιδιωτικότητας, αυτή διατηρείται κατά το post-processing (απόδειξη στο [3]). Αυτό, όμως που επηρεάζει την ιδιωτικότητα είναι το γεγονός ότι τα μηνύματα δε μπορούν να διαγραφούν και οι servers μπορούν μόνο να προσθέσουν μηνύματα. Έτσι, στην πραγματικότητα, ο θόρυβος που προστίθεται σε κάθε άθροισμα είναι τυχαία μεταβλητή που ακολουθεί την κατανομή $D = \max(0, \lceil Lap(\mu, b) \rceil)$. Κάθε server δειγματοληπτεί τη συγκεκριμένη κατανομή δύο φορές και προσθέτει D_1 μηνύματα σε τυχαία dead drops και $2 \cdot D_2$ μηνύματα που πάνε ανά ζευγάρια σε ίδια τυχαία dead drops. Έτσι, ο συνολικός αριθμός μηνυμάτων θορύβου που προσθέτει ο κάθε server έχει μέση τιμή $3 \cdot \mu$. Το γεγονός ότι τα μηνύματα δε μπορούν να διαγραφούν είναι αυτό που κάνει το μηχανισμό (ε, δ)-dp αντί για $(\varepsilon, 0)$-dp. Και αυτό γιατί υπάρχουν περιπτώσεις στις οποίες η κατανομή μπορεί να δειγματοληπτηθεί κοντά στο 0 και ο θόρυβος που θα προστεθεί να μην είναι αρκετός για να προσφέρει διαφορική ιδιωτικότητα. Ένα τέτοιο παράδειγμα με 5 χρήστες φαίνεται παρακάτω:

Ας υποθέσουμε τους ακόλουθους γειτονικούς πίνακες x και y, οι οποίοι αναπαριστούν την κατάσταση του προβλήματος της επικοινωνίας, δηλαδή ποιος μιλάει με ποιον.

user_id	1	2	3	4	5
1	0	1	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	0	0	0	1	0
5	1	0	0	0	0

Πίνακας 2.2: state matrix x

user_id	1	2	3	4	5
1	0	1	0	0	0
2	0	1	0	0	0
3	1	0	0	0	0
4	0	0	0	1	0
5	1	0	0	0	0

Πίνακας 2.3: state matrix y

Για τους αριθμούς των dead drops με μία και με δύο προσβάσεις στις δύο περιπτώσει ισχύει αντίστοιχα $m_1^x = 3, m_2^x = 1$ και $m_1^y = 5, m_2^y = 0$. Τώρα ας υποθέσουμε ότι πίνακας x είναι η πραγματική κατάσταση και ότι η κατανομή Laplace δειγματοληπτείται στο 1 και για το m_1 και για το m_2. Η έξοδος που θα είναι ορατή στον αντίπαλο θα είναι $m_1' = 4$ και $m_2' = 2$. Έτσι, ο αντίπαλος μπορεί με σιγουριά να αποκλείσει την περίπτωση η πραγματική κατάσταση να είναι η y, παραβιάζοντας τον ορισμό της $(\varepsilon, 0)$-διαφορικής ιδιωτικότητας που απαιτεί κάθε παρατήρηση να μην είναι ικανή να διαχωρίσει ουσιαστικά δύο γειτονικές καταστάσεις.

Μετά από επιπλέον παρατήρηση, συμπεραίνουμε ότι τουλάχιστον 2 μηνύματα απαιτείται να προστεθούν σε κάθε μεταβλητή ώστε να έχουμε αυτή την εγγύηση. Η πιθανότητα να προστεθούν λιγότερα μηνύματα είναι η παράμετρος δ της (ε, δ)-dp που εγγυάται το σύστημα.

Αυτό προκύπτει όπως φαίνεται στη συνέχεια, υποθέτοντας το μηχανισμό της Vuvuzela $\mathcal{M}$ και δύο γειτονικές καταστάσεις x, y:

$$\begin{aligned} Pr[\mathcal{M}(x) \in \mathcal{S}] &= Pr[\mathcal{M}(x) \in \mathcal{S}|\text{enough noise}] + Pr[\mathcal{M}(x) \in \mathcal{S}|\text{not enough noise}] \\ &\leq e^{\varepsilon} Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta. \end{aligned}$$

Η παραπάνω έκφραση λέει ότι η Vuvuzela είναι (ε, δ)-διαφορικά ιδιωτική με παραμέτρους $\varepsilon = \frac{3}{b}$ και $\delta = exp(\frac{2-\mu}{b})$.

Η παράμετρος δ υπολογίζεται ως εξής:

$$\begin{aligned} Pr[\text{not enough noise}] &\leq Pr[Lap(\mu, b) \leq 2] + Pr[Lap(\mu, b) \leq 1] = \\ &= \frac{1}{2} exp(\frac{2-\mu}{b}) + \frac{1}{2} exp(\frac{1-\mu}{b}) \\ &\leq exp(\frac{2-\mu}{b}). \end{aligned}$$

Εδώ είναι το σημείο που πρέπει να αναφερθούμε στη διαφορά ανάμεσα στην ποσότητα του θορύβου που απαιτείται σύμφωνα με τη δικιά μας ανάλυση σε αντίθεση με την ανάλυση των δημιουργών του

μηχανισμού. Στην ανάλυση του paper, οι συγγραφείς θεωρούν δύο ξεχωριστούς μηχανισμούς με τον καθένα να αποκρύπτει μία μεταβλητή και στο τέλος τους συνδυάζουν για να λάβουν έναν μηχανισμό με παραμέτρους $\varepsilon = \frac{4}{b}$ και $\delta = exp(\frac{2-\mu}{b})$ προσθέτοντας $\max(0, \lceil Lap(\mu, b) \rceil)$ μονά μηνύματα θορύβου και $\max(0, \lceil Lap(\frac{\mu}{2}, \frac{b}{2}) \rceil)$ ζευγάρια μηνυμάτων. Η σύνθεση των δύο μηχανισμών σε έναν γίνεται βάσει του θεωρήματος 1. Συμπερασματικά, με αυτή την ανάλυση απαιτούνται κατά μέσο όρο $\frac{8ln\delta^{-1}}{\varepsilon}$ μηνύματα θορύβου συνολικά από κάθε server. Με βάση τη δική μας ανάλυση απαιτούνται $\frac{9ln\delta^{-1}}{\varepsilon}$ μηνύματα θορύβου από κάθε server. Αυτή η διαφορά παρατηρείται λόγω του γεγονότος ότι για την προσθήκη μιας μονάδας στο άθροισμα m_2 απαιτείται η χρήση **δύο** μηνυμάτων, και έτσι ενώ η ανάλυσή μας δίνει καλύτερα θεωρητικά αποτελέσματα στη γενικά περίπτωση, η δική τους δίνει τον ελάχιστο αριθμό μηνυμάτων για το συγκεκριμένο σύστημα.

2.3.6 Ιδιωτικότητα μετά από πολλούς γύρους

Όλη η συζήτηση από την αρχή του κεφαλαίου αφορά την ιδιωτικότητα για ένα γύρο του μηχανισμού Vuvuzela, δηλαδή για ένα και μοναδικό μήνυμα που ανταλλάσσεται από κάθε χρήστη. Όπως είναι λογικό, αφού έχουμε κάποια διαρροή πληροφορίας σε κάθε γύρο, για να επιτύχουμε τις ίδιες εγγυήσεις διαφορικής ιδιωτικότητας μετά από πολλούς γύρους θα πρέπει να προσθέσουμε περισσότερο θόρυβο. Το θεώρημα 3 μας δείχνει πως οι παράμετροι ε και δ εξασθενούν με το πέρασμα των γύρων. Με βάση αυτό το θεώρημα μπορούμε να επιλέξουμε τις παραμέτρους που επιθυμούμε να έχουμε μετά από ένα μεγάλο αριθμό γύρων και να λάβουμε τις παραμέτρους για τον κάθε γύρο. Τυπικές παράμετροι για τον αριθμό των γύρων που θέλουμε να εγγυηθούμε είναι γύρω στους 200.000 και για την ιδιωτικότητα που θέλουμε μετά από τόσους γύρους $\varepsilon \approx ln2$ και $\delta \approx 10^{-4}$. Για την επίτευξη αυτών των παραμέτρων, τυπικές τιμές για την κατανομή Laplace που πρέπει να δειγματοληπτήσουμε είναι $\mu \approx 150.000 - 450.000$ και $b \approx 7.300 - 20.000$. Πληροφοριακά, στην υλοποίηση του μηχανισμού επιλέχθηκε $\mu = 300.000$. Βλέπουμε λοιπόν ότι γενικά απαιτείται αρκετά μεγάλος αριθμός μηνυμάτων θορύβου αλλά αυτός ο αριθμός δεν εξαρτάται καθόλου από τον αριθμό των χρηστών του συστήματος.

2.3.7 Συνολικό κόστος επικοινωνίας του Vuvuzela

Υπενθυμίζοντας ότι ο κάθε server προσθέτει συνολικά $\frac{3\mu}{2}$ κατά μέσο όρο επιπλέον μηνύματα σε κάθε γύρο, ο συνολικό αριθμός των μηνυμάτων που στέλνονται σε έναν γύρο λειτουργίας του συστήματος είναι κατά μέσο όρο:

$$2(n \cdot s + \frac{3\mu s(s-1)}{4})$$

όπου n είναι ο αριθμός των χρηστών και s ο αριθμός των server που χρησιμοποιούνται. Είναι λοιπόν φανερό ότι το συνολικό κόστος επικοινωνίας του συστήματος κλιμακώνει γραμμικά σε σχέση με τον αριθμό των χρηστών και τετραγωνικά σε σχέση με τον αριθμό των server. Το δεύτερο δεν έχει πραγματικά σημασία γιατί ο αριθμός των server παραμένει μικρός σε κάθε περίπτωση. Για παράδειγμα, σε ένα σενάριο με $n = 1$ εκατομμύριο χρήστες και $s = 3$ servers με εγγυήσεις $(ln2, 10^4)$-διαφορικής ασφάλειας μετά από $k = 200.000$ γύρους (κάτι το οποίο δίνει $\mu \approx 400.000$), ο συνολικός αριθμός μηνυμάτων μου μεταδίδονται στο σύστημα είναι περίπου 10 εκατομμύρια. Πειραματικά δεδομένα δείχνουν ότι ένας τέτοιος όγκος μηνυμάτων μπορεί να εξυπηρετηθεί από το σύστημα προσφέροντας καθυστέρηση κάτω του ενός λεπτού στην αποστολή μηνυμάτων.

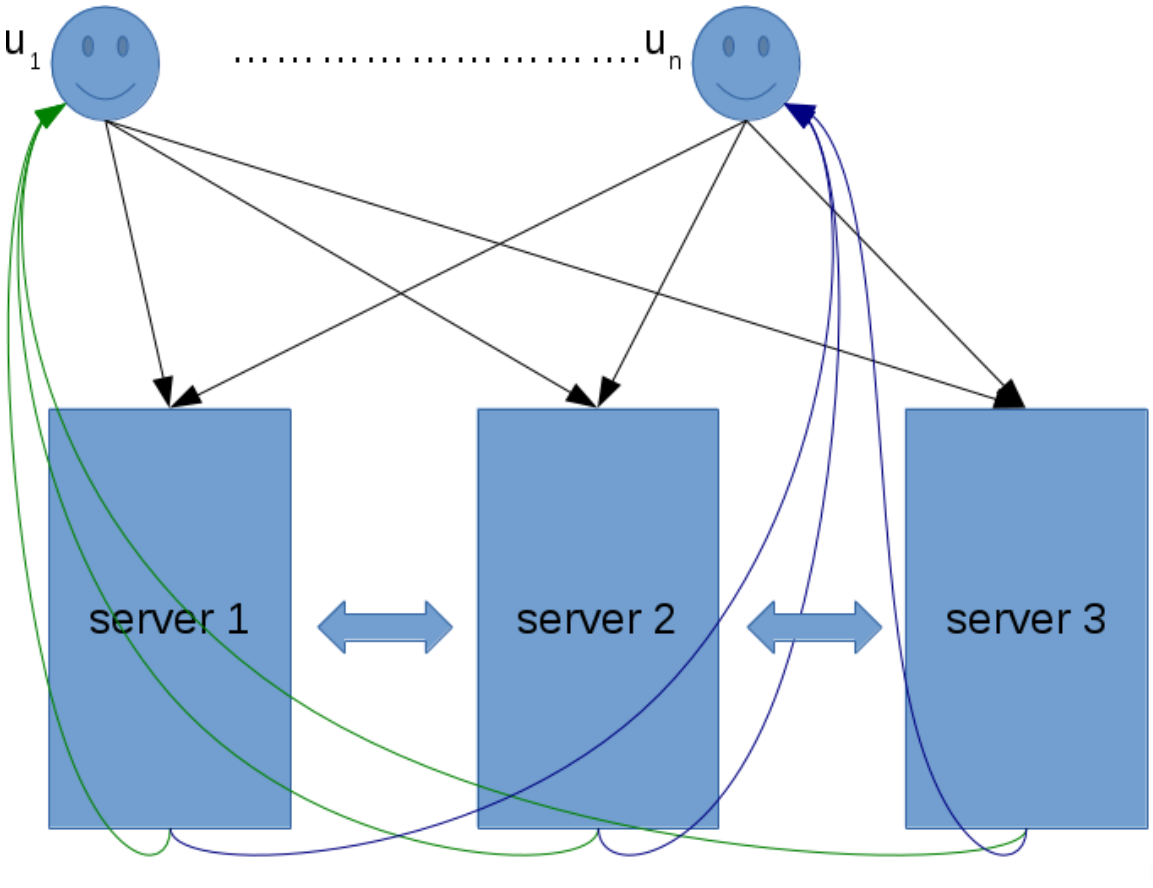
2.4 Σύγκριση των διαθέσιμων λύσεων

Για να συγκρίνουμε τις λύσεις του προβλήματος της ανώνυμης ανταλλαγής μηνυμάτων που αναφέραμε σε αυτό το κεφάλαιο θα λάβουμε υπόψιν μας κυρίως δύο παράγοντες. Ο πρώτος είναι η ασφάλεια που προσφέρει το σύστημα και ο δεύτερος ο αριθμός των χρηστών που μπορεί να εξυπηρετήσει, δηλαδή πώς κλιμακώνει μια λύση σε σχέση με τον αριθμό των χρηστών. Αρχικά, το Tor μπορεί να εξυπηρετήσει άνετα πάρα πολλούς χρήστες με πολύ μικρή καθυστέρηση μηνυμάτων, αλλά όπως αναφέραμε δε προσφέρει ουσιαστικά καμία ασφάλεια ενάντια σε έναν αντίπαλο που ελέγχει όλο το δίκτυο. Οι μηχανισμοί που βασίζονται σε DC-nets, από την άλλη μεριά προσφέρουν πολύ καλή ασφάλεια (κρυπτογραφική) με το λογικό επακόλουθο ότι δεν είναι κλιμακώσιμοι και μπορούν να εξυπηρετήσουν το πολύ λίγους χιλιάδες χρήστες. Πρωτόκολλα που βασίζονται αποκλειστικά σε mixnets είναι γενικά πιο κλιμακώσιμα αλλά και πάλι απαιτούν ακριβές αποδείξεις μηδενικής γνώσης ενώ είναι ευάλωτα σε κάποιες επιθέσεις που ο αντίπαλος μπορεί να μπλοκάρει μηνύματα από κάποιους χρήστες και να βγάλει συμπεράσματα. Ο μηχανισμός Vuvuzela χρησιμοποιώντας ψευδοτυχαία σημεία συνάντησης, mixnets και επιπλέον θόρυβο προσφέρει μια καλή και κλιμακώσιμη λύση στο πρόβλημα. Η ασφάλεια, όμως που προσφέρει δεν είναι απόλυτη και εκφράζεται μέσα από την διαφορική ιδιωτικότητα. Ιδανικά, θα θέλαμε ένα σύστημα που να προσφέρει απόλυτη, εάν είναι δυνατόν και information-theoretic ασφάλεια και να μπορεί να εξυπηρετήσει έναν αρκετά μεγάλο αριθμό χρηστών. Μια τέτοια λύση θα προσπαθήσουμε να προσεγγίσουμε στο υπόλοιπο της εργασίας.

Κεφάλαιο 3

Λύσεις μέσω του Secure Multiparty Computation

3.1 Εισαγωγή



Σχήμα 3.1: SMC 3 server mechanism

Σε αυτό το κεφάλαιο, θα παρουσιάσουμε το πρόβλημα της ανώνυμης ανταλλαγής μηνυμάτων ως μαθηματική συνάρτηση και θα προτείνουμε μία υλοποίηση αυτής της συνάρτησης χρησιμοποιώντας τεχνικές Secure Multiparty Computation (παράρτημα Α.4). Εδώ πρέπει να σημειώσουμε ότι θα ασχοληθούμε κυρίως με το πρωτόκολλο συζήτησης (conversation protocol), όπως αυτό ορίζεται στην ιδεατή λειτουργικότητα F του σχήματος 1.1. Δηλαδή θα υποθέσουμε πως οι δύο χρήστες που θέλουν να ανταλλάξουν μηνύματα έχουν συμφωνήσει με κάποιο τρόπο σε ένα κοινό μυστικό (shared secret), το οποίο χρησιμοποιούν ως σημείο-ραντεβού. Αυτή είναι και η χρησιμότητα των βημάτων DIAL και DIALCHECK της ιδεατής λειτουργικότητας F. Με τη σειρά τους αυτές οι λειτουργίες μπορούν να γίνουν σε πιο αραιά διαστήματα και με μεγαλύτερο latency και έτσι η έρευνά μας επικεντρώνεται στο

conversation protocol.

Θα προσπαθήσουμε λοιπόν να δώσουμε μια λύση στο πρόβλημα της ανταλλαγής μηνυμάτων μέσω ασφαλούς υπολογισμού μιας συνάρτησης από κάποιους servers (server-aided SMC). Φυσικά, ως πρώτο βήμα πρέπει να ορίσουμε την επιθυμητή συνάρτηση f. Στη συνέχεια, πρέπει να μετασχηματίσουμε τη συνάρτηση f σε ένα κύκλωμα C αποτελούμενο αποκλειστικά από δυαδικές (boolean) πύλες, καθώς αυτή είναι η αναγκαία είσοδος για τα πρωτόκολλα SMC. Γενικά, μπορούμε να σκεφτούμε τον μηχανισμό μας να αποτελείται από έναν αριθμό από servers (3 στο παράδειγμά μας) οι οποίοι θα δέχονται ένα μερίδιο (share) του input κάθε χρήστη σε κάθε γύρο r, στη συνέχεια θα υπολογίζουν τη συνάρτηση f με ασφαλή τρόπο, επικοινωνώντας μεταξύ τους όποτε χρειάζεται, και στο τέλος κάθε γύρου θα γυρίζουν ένα μερίδιο του output σε κάθε χρήστη. Ένα τέτοιο πρωτόκολλο φαίνεται στο σχήμα 3.1. Ο κάθε χρήστης συνδυάζοντας τα shares του τελικά θα ανακτά την επιθυμητή έξοδο της συνάρτησης, δηλαδή το μήνυμα που θα του αντιστοιχεί. Φυσικά, κάθε επικοινωνία μεταξύ των χρηστών και των servers, όσο και των servers μεταξύ τους θα γίνεται μέσω ασφαλών (κρυπτογραφημένων) καναλιών.

Ο στόχος αυτής της προσέγγισης είναι η στόχευση της υψηλής ασφάλειας που προσφέρουν τα πρωτόκολλα SMC στο πεδίο της ασφαλούς επικοινωνίας.

3.2 Το πρόβλημα ως συνάρτηση

Σύμφωνα με το υποπρωτόκολλο συζήτησης (conversation component) της ιδεατής λειτουργικότητας του σχήματος 1.1, η συνάρτησή μας f, θα πρέπει να δέχεται ως είσοδο n αιτήματα (requests) της μορφής (i, t_i, m_i), κάθε ένα από το χρήστη u_i, $i = 1, \ldots, n$ αντίστοιχα. Το πρώτο στοιχείο της τριάδας (i) δηλώνει τον αριθμό της εισόδου (input wire id) του χρήστη, το δεύτερο στοιχείο (t_i) είναι το σημείο ραντεβού στο οποίο θέλει να απευθυνθεί ο χρήστης u_i, και το τρίτο (m_i) το μήνυμα που θέλει να στείλει. Ως έξοδο, η συνάρτηση θα πρέπει να επιστρέφει n απαντήσεις της μορφής m_j, $j = 1 \ldots n$, μία για κάθε χρήστη u_j, όπως περιγράφεται παρακάτω:

$$f : (t_i, m_i)^n \rightarrow m_j^n$$

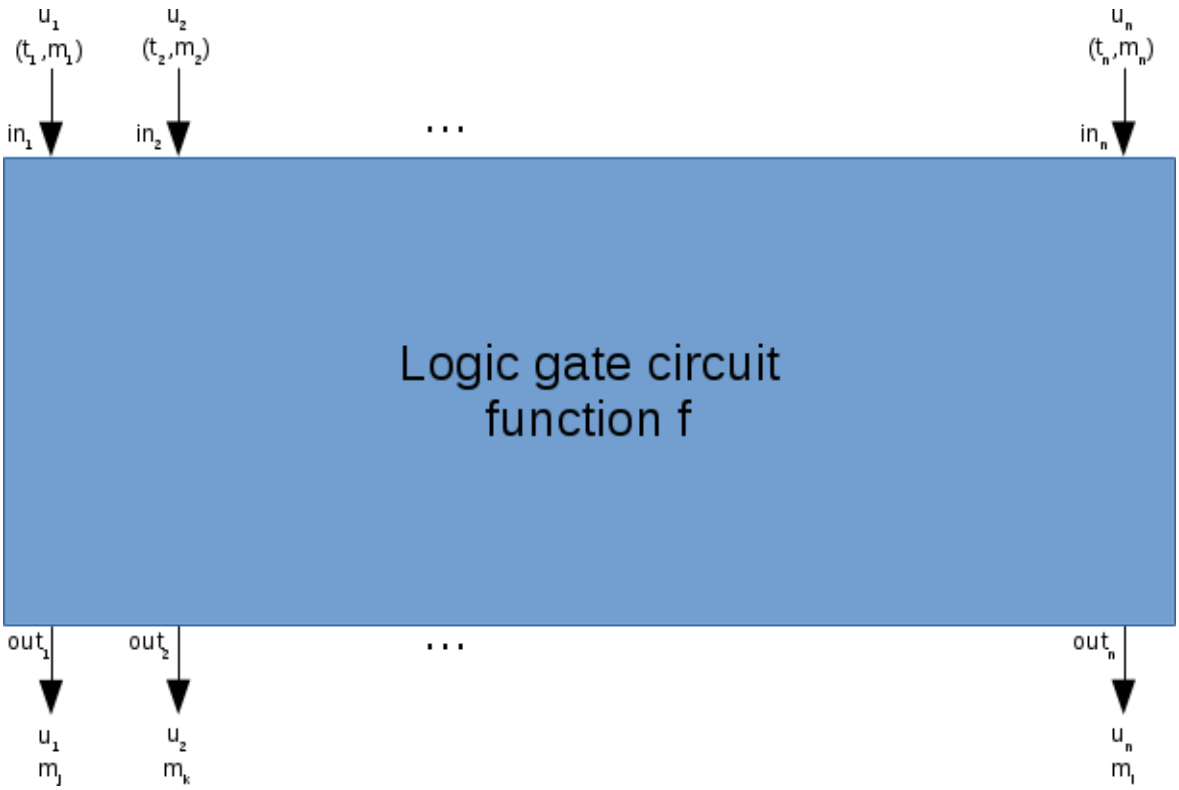
$$f = (f_1, \ldots, f_n)$$

$$f_i = \begin{cases} m_j & \text{if } \exists\, i \neq j : t_i = t_j \\ 0 & \text{else.} \end{cases}$$

Η παραπάνω συνάρτηση υλοποιεί την ιδέα της ανταλλαγής μηνυμάτων μεταξύ δύο χρηστών που έχουν συμφωνήσει σε μία μοιραζόμενη τιμή t_i. Ο τρόπος με τον οποίο οι δύο ενδιαφερόμενοι χρήστες αποκτούν αυτή τη μοιραζόμενη τιμή αποτελεί αντικείμενο του DIAL και DIALCHECK μέρους της ιδεατής λειτουργικότητας F και όπως έχουμε αναφέρει ξανά δεν αποτελεί βασικό σημείο της έρευνάς μας. Είναι επίσης σημαντικό να τονίσουμε ότι δύο τιμές t_i και t_j μπορεί να είναι ίσες μόνο εάν έχουν παραχθεί από κοινού από δύο χρήστες που θέλουν να επικοινωνήσουν. Με άλλα λόγια, είναι εξαιρετικά απίθανο να συμπέσουν τυχαία δύο σημεία συνάντησης. Αυτό μπορεί να εξασφαλιστεί δίνοντας τη δυνατότητα για μεγάλο εύρος πιθανόν σημείων συνάντησης (πχ 2^{128}).

Ο κάθε χρήστης συνεισφέρει μία είσοδο και δέχεται μία έξοδο από τη συνάρτηση. Εάν κάποιος online χρήστης δεν παίρνει μέρος σε κάποια συζήτηση, απλά στέλνει μηδενικά ή τυχαία μηνύματα και λαμβάνει μηδενικές απαντήσεις.

Όπως ξέρουμε από θεωρητικά αποτελέσματα [23], κάθε συνάρτηση μπορεί να υπολογιστεί με ασφαλή τρόπο χρησιμοποιώντας πρωτόκολλα SMC και σχήματα ασφαλούς διαμοιρασμού της εισόδου (secret sharing schemes). Όμως, αυτά τα πρωτόκολλα απαιτούν η συνάρτηση να είναι εκφρασμένη ως ένα κύκλωμα λογικών πυλών. Προς αυτή την κατεύθυνση, θα θέλαμε ένα κύκλωμα που να υλοποιεί τη



Σχήμα 3.2: circuit C of function f

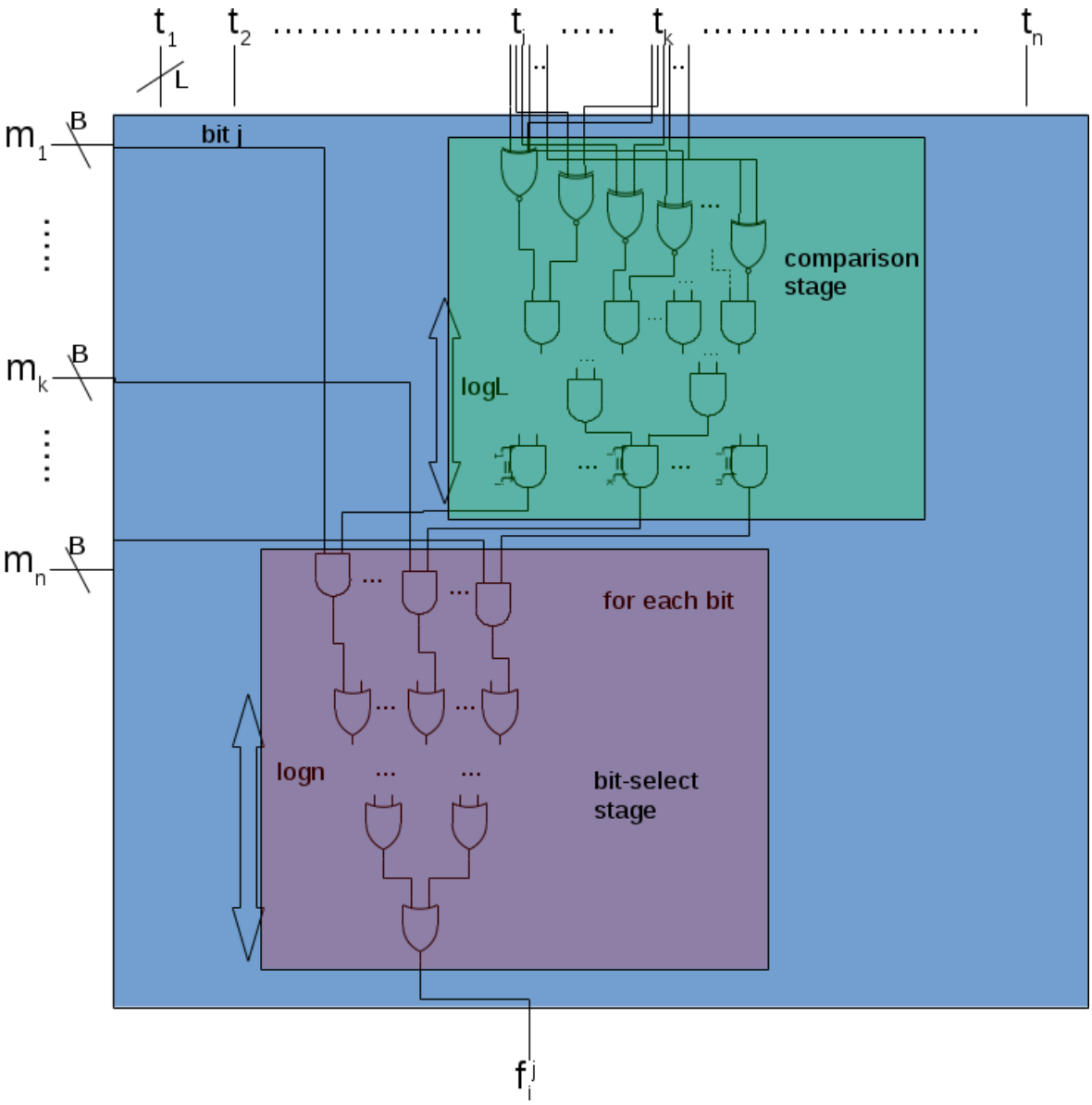
συνάρτηση f, όπως φαίνεται στο σχήμα 3.2. Στο κύκλωμα του σχήματος 3.2, έχουμε ακριβώς n καλώδια εισόδου (μπορούμε να πούμε και "πολύκλωνα" καλώδια καθώς κάθε ένα αντιπροσωπεύει έναν αριθμό από bits) και άλλα τόσα εξόδου. Κάθε ένα από αυτά τα καλώδια αντιπροσωπεύει την είσοδο και αντίστοιχα την έξοδο ενός από τους n χρήστες. Όπως ξέρουμε από τη λογική, η κατασκευή ενός δυαδικού κυκλώματος είναι εφικτή για κάθε υπολογίσιμη συνάρτηση, παρ' όλα αυτά η επίτευξη καλής πολυπλοκότητας δεν είναι καθόλου εύκολη. Στο επόμενο μέρος θα παρουσιάσουμε ένα κύκλωμα που υλοποιεί τη συνάρτηση f και ακολούθως θα παρουσιάσουμε πιο αποδοτικές εκδοχές του που μπορεί να οδηγήσουν σε ένα πρακτικό και υλοποιήσιμο σύστημα.

3.3 Ένα απλό κύκλωμα

Το πρώτο βήμα για την κατασκευή ενός κυκλώματος για οποιαδήποτε μαθηματική συνάρτηση είναι η έκφραση της συνάρτησης στην άλγεβρα Boole. Από αυτό το σημείο και στη συνέχεια της εργασίας, υποθέτουμε ότι οι τιμές των σημείων συνάντησης t_i έχουν μήκος L bits και τα μηνύματα m_i μήκος B bits, αντίστοιχα. Είναι αρκετά εύκολο να φτιάξουμε μια έκφραση άλγεβρας Boole για κάθε bit του διανύσματος εξόδου $f = (f_1, \ldots, f_n)$, όπου f_i η έξοδος που προορίζεται για τον χρήστη i, και t_i^j, f_i^j και m_i^j αναπαριστούν το j-οστό bit των t_i, f_i και m_i αντίστοιχα:

$$f_i^j = \bigvee_{k=1}^{n} [m_k^j \wedge \bigwedge_{l=1}^{L} (t_i^l \text{ XNOR } t_k^l)]$$

Η παραπάνω έκφραση δίνει το j-οστό bit της εξόδου f_i (απευθυνόμενη στον χρήστη i). Η έκφραση μπορεί να φαίνεται περίπλοκη αλλά είναι στην ουσία της απλή. Η εσωτερική έκφραση XOR, ελέγχει bit προς bit την ισότητα μεταξύ των bits των t_i και t_k. Η έκφραση AND που τρέχει πάνω στο μήκος



Σχήμα 3.3: κύκλωμα C_1

των t, δηλαδή εφαρμόζεται σε L bits, έχει ως είσοδο τους bit-wise ελέγχους της έκφρασης XOR και παράγει ως έξοδο 1 εάν $t_i = t_k$ και 0 σε κάθε άλλη περίπτωση. Τελικά, το j-οστό bit της εισόδου m_k προωθείται στην έξοδο εάν υπάρχει κάποιο k που να ικανοποιεί την εξίσωση $t_i = t_k$. Αυτό επιτυγχάνεται από την εξωτερική έκφραση OR. Είναι σημαντικό να υπογραμμίσουμε ότι η εσωτερική έκφραση AND υπολογίζεται μόνο μία φορά για όλα τα bits f_i^j, $j = 1, \ldots, B$ της εξόδου. Επίσης, πρέπει να τονίσουμε ότι είναι εξαιρετικά απίθανο να υπάρχουν περισσότερα από δύο ίδια t λόγω του μεγάλου τους μεγέθους και του τρόπου υπολογισμού τους. Εάν δεν υπάρχει k που να ικανοποιεί τη συνθήκη ισότητας, τότε η έξοδος είναι προφανώς 0.

Από την παραπάνω έκφραση μπορούμε να δημιουργήσουμε ένα απλό κύκλωμα, αποτελούμενο από δυαδικές πύλες, το οποίο να υλοποιεί τη συνάρτηση f. Αυτό παρουσιάζεται στο σχήμα 3.3.

Το κύκλωμα αυτό λύνει το πρόβλημα της υλοποίησης της συνάρτησης f και κατ'επέκταση το πρόβλημα της ανταλλαγής μηνυμάτων. Παρατηρώντας όμως το κύκλωμα, συμπεραίνουμε ότι απαιτούνται $\frac{n(n-1)}{2}$, δηλαδή $\mathcal{O}(n^2)$ μονάδες συγκριτών, κάτι που σημαίνει ότι το κύκλωμα C_1 δεν είναι

κατάλληλο για πρακτικές εφαρμογές. Παρ'όλα αυτά δίνει μία βάση πάνω στην οποία θα δουλέψουμε στην επόμενη ενότητα ώστε να παρουσιάσουμε ένα κύκλωμα με μικρότερη πολυπλοκότητα.

Σε αυτό τη σημείο είναι σημαντικό να εξηγήσουμε τι εννοούμε πολυπλοκότητα για ένα κύκλωμα στο πεδίο του SMC. Στα περισσότερα SMC πρωτόκολλα, ο υπολογισμός των λογικών πυλών XOR και INV (NOT) μπορεί να γίνει χωρίς ουσιαστικό κόστος (ανατρέξτε στο παράρτημα A.4 για περισσότερες λεπτομέρειες). Έτσι, στην ανάλυσή μας από εδώ και πέρα, θα μας ενδιαφέρει ο αριθμός των πυλών AND (υποθέτοντας ότι οι πύλες OR μπορούν να αναπαρασταθούν με πύλες AND). Επίσης, πύλες που βρίσκονται στο ίδιο επίπεδο σε ένα κύκλωμα, δηλαδή πύλες των οποίων οι είσοδοι είναι διαθέσιμες την ίδια λογική στιγμή, μπορούν να αποτιμηθούν παράλληλα. Έτσι, σημαντική μετρική για την απόδοση ενός κυκλώματος είναι το βάθος του, δηλαδή η μεγαλύτερη διαδρομή από μία είσοδό του προς μία έξοδό του, μετρώντας δυαδικές πύλες AND.

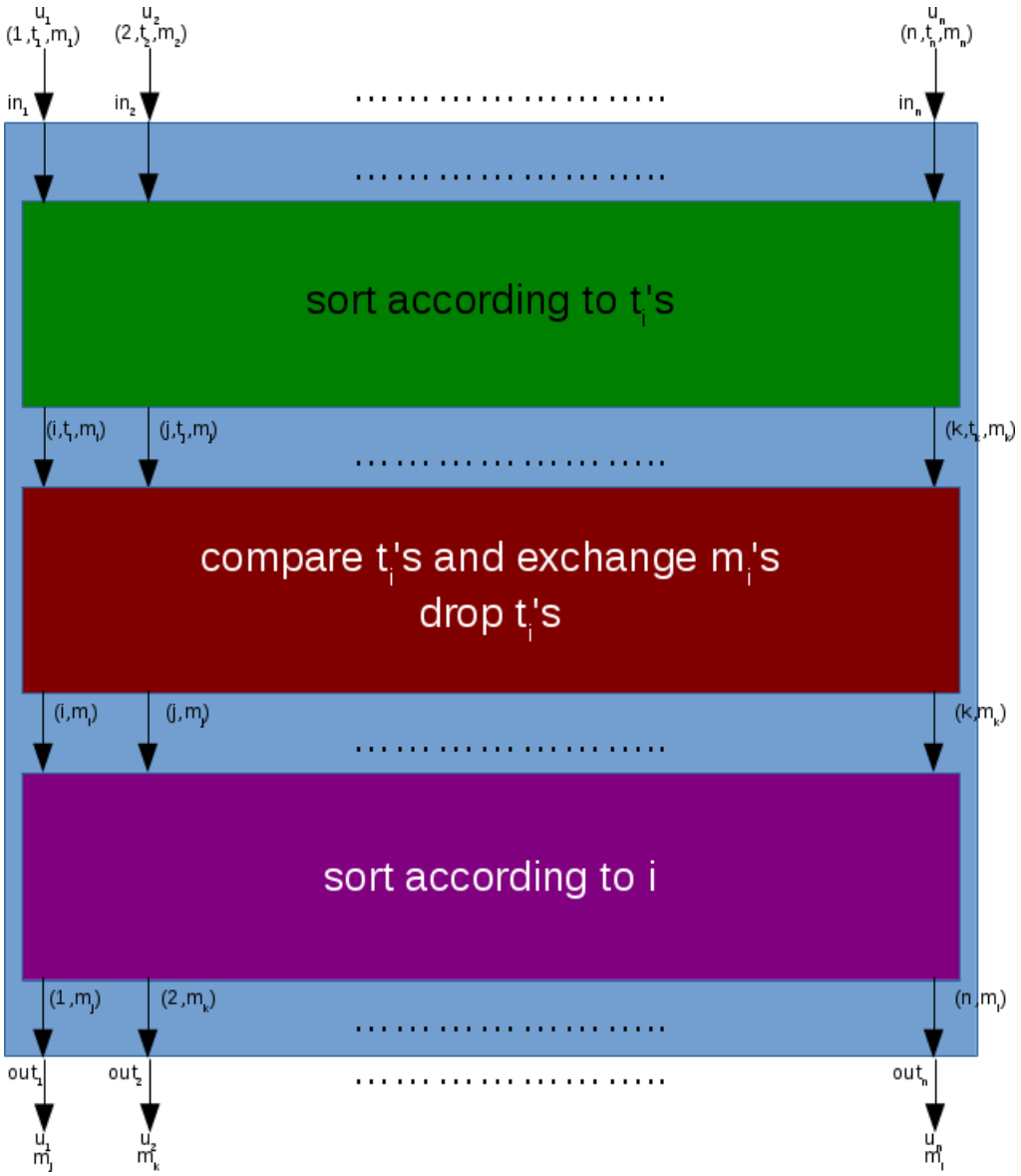
3.4 Μια πιο αποδοτική λύση

Για να συνθέσουμε ένα δίκτυο κατάλληλο για μια αποδοτική υλοποίηση, θα χρησιμοποιήσουμε προϋπάρχουσα έρευνα πάνω στα δίκτυα ταξινόμησης (παράρτημα A.5). Ένα δίκτυο ταξινόμησης λύνει το πρόβλημα της ταξινόμησης των τιμών που βρίσκονται στην είσοδό του χρησιμοποιώντας μόνο συγκριτές και ανταλλαγές τιμών. Οι αλγόριθμοι που υλοποιούνται σε τέτοια δίκτυα (κυκλώματα) διαφέρουν από τους κλασικούς αλγορίθμους που ξέρουμε (mergesort, quicksort) γιατί οι συγκρίσεις που πρέπει να εκτελέσουν δεν πρέπει να εξαρτώνται από το αποτέλεσμα προηγούμενων συγκρίσεων. Στο σχεδιασμό του κυκλώματός μας θα χρησιμοποιήσουμε το δίκτυο ταξινόμησης odd-even mergesort του Batcher [18], το οποίο έχει βάθος $\mathcal{O}(log^n)$ και συνολικό αριθμό συγκριτών $\mathcal{O}(nlog^n)$. Για να μιλάμε πιο συγκεκριμένα, για την ταξινόμηση n τιμών αυτό το κύκλωμα χρειάζεται $\frac{log^2 n}{2}$ επίπεδα με πλάτος $\frac{n}{2}$ το καθένα. Το κύκλωμα C_2 που προτείνουμε φαίνεται στο σχήμα 3.4.

Στο κύκλωμα του σχήματος 3.4, οι χρήστες υποβάλλουν ως εισόδους τριάδες της μορφής (i, t_i, m_i). Η πρώτη συντεταγμένη της τριάδας είναι το wire id του χρήστη, δηλαδή το καλώδιο που του αντιστοιχεί. Σημειώνεται ότι το wire id δεν είναι απαραίτητο να παίρνει τιμές από το $\{1, \ldots, n\}$ αλλά στη συνέχεια της εργασίας μας θα υποθέσουμε ότι το καλώδιο που αντιστοιχεί στο χρήστη u_i είναι το i. Επίσης, το wire id δεν είναι απαραίτητο να δίνεται στο κύκλωμα ως είσοδο από το χρήστη αλλά μπορεί να επιλέγεται από τους servers χρησιμοποιώντας κάποιο πρωτόκολλο ώστε να αποφεύγονται οι επιθέσεις denial of service. Η δεύτερη συντεταγμένη t_i είναι η τιμή του σημείου ραντεβού, και η τρίτη m_i είναι το (πιθανόν κρυπτογραφημένο) μήνυμα.

Το κύκλωμα λειτουργεί ως εξής. Αρχικά, οι τριάδες ταξινομούνται ανάλογα με τις t συντεταγμένες τους (πράσινο κομμάτι του κυκλώματος). Στη συνέχεια, οι γειτονικές τριάδες ελέγχονται ανά δύο και εάν διαπιστωθεί πως έχουν ίσες τιμές όσον αφορά τα t, ανταλλάσσονται (swap). Αυτή η λειτουργία αναπαριστάται από το κόκκινο κομμάτι του κυκλώματος. Από αυτό το σημείο και μετά, οι t συντεταγμένες των τριάδων δεν έχουν κάποιο λόγω ύπαρξης και έτσι αγνοούνται από το κύκλωμα. Στο τελευταίο στάδιο του κυκλώματος (μοβ), οι δυάδες πια της μορφής (i, m_j) ταξινομούνται ως προς την πρώτη τους συντεταγμένη (wire id) ώστε τα μηνύματα να φτάσουν στα καλώδια εξόδου που αντιστοιχούν στους επιθυμητούς τους παραλήπτες.

Όπως είναι φανερό, το παραπάνω κύκλωμα έχει περίπου την πολυπλοκότητα δύο δικτύων ταξινόμησης, τόσο όσον αφορά το βάθος αλλά και το συνολικό αριθμό πυλών. Το επόμενό μας βήμα θα είναι να ορίσουμε τις μονάδες compare-and-exchange που χρησιμοποιούνται σε κάθε ένα από τα τρία στάδια του κυκλώματος του σχήματος 3.4. Επίσης θα μετρήσουμε τον αριθμό πυλών (AND) που απαιτούν αυτές οι μονάδες, καθώς και το βάθος κάθε μίας από αυτές. Υπενθυμίζουμε ότι στον υπολογισμό των πυλών θα λάβουμε υπ'όψιν μόνο τις πύλες AND καθώς αυτές είναι που επιφέρουν επικοινωνιακό και υπολογιστικό κόστος στα περισσότερα πρωτόκολλα SMC.

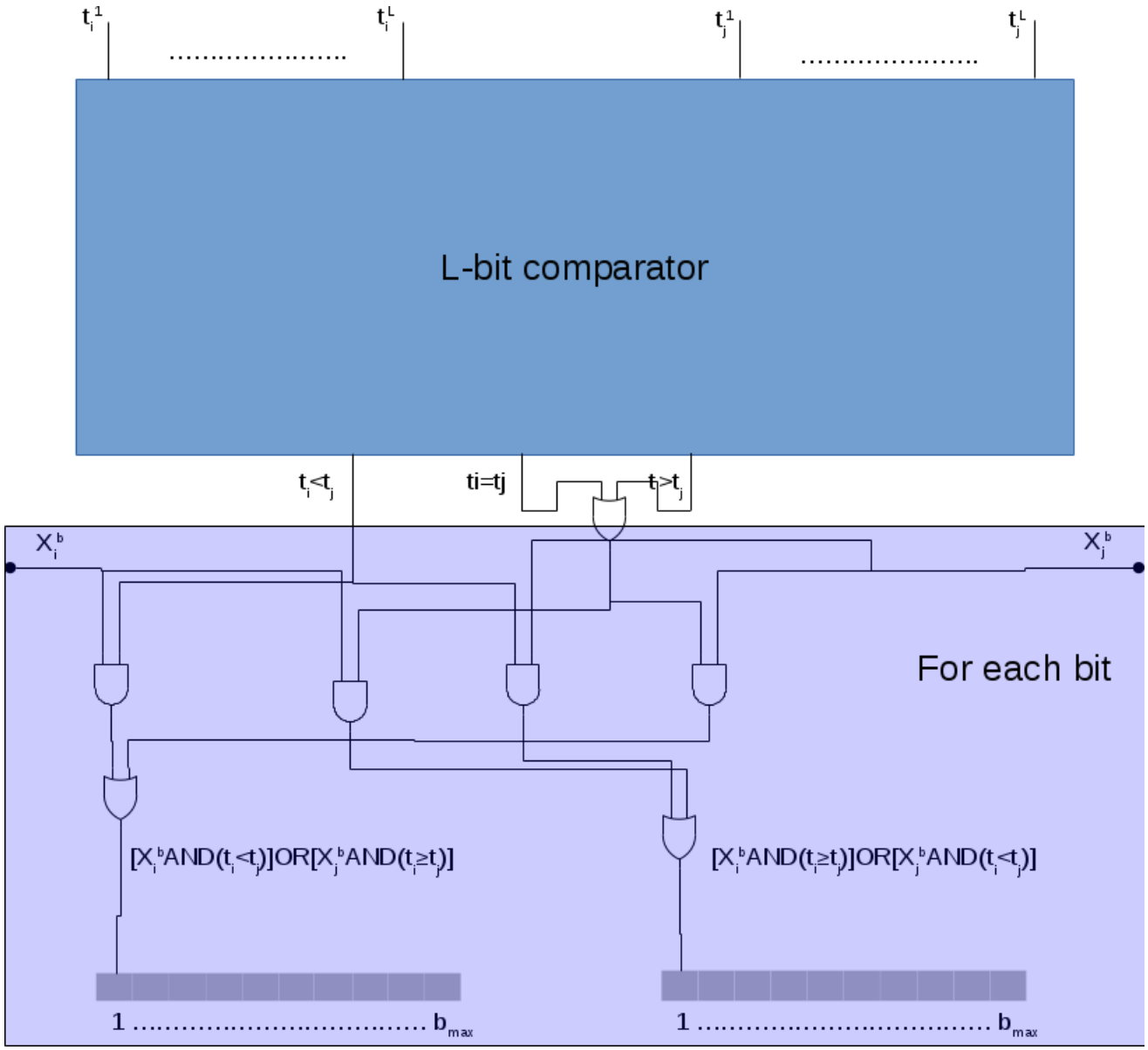


Σχήμα 3.4: Κύκλωμα C_2

3.4.1 Μονάδα compare-and-exchange A

Η πρώτη μονάδα που θα χρειαστούμε θα πρέπει να λαμβάνει ως είσοδο δύο τριάδες της μορφής (i, t_i, m_i) και (j, t_j, m_j) και θα προχωράει συγκρίνοντας τις t συντεταγμένες τους. Εάν $t_i > t_j$, τότε θα εναλλάσσει (swap) τις δύο τριάδες, όπως φαίνεται στο σχήμα 3.5. Η μονάδα του σχήματος αποτελείται από ένα συγκριτή των L bits, ο οποίος μπορεί εύκολα να παρασκευαστεί από απλούς συγκριτές των 4 bits, και από ένα μικρό υπο-κύκλωμα που υλοποιεί την ανταλλαγή των τιμών. Στο σχήμα 3.5, κάθε X_i^b αναπαριστά το b-οστό bit των t_i, m_i ή i's. Πιο αποδοτικές κατασκευές για τη μονάδα A είναι εφικτές και βασίζονται στη χρήση πιο αποδοτικών συγκριτών L bits, όπως θα δούμε στη συνέχεια. Σε αυτό το σημείο, πάντως, για την κατανόηση της λειτουργίας του κυκλώματος υποθέτουμε απλούς συγκριτές.

Το κόστος της μονάδας A είναι αυτό ενός συγκριτή L bits συν έξι πύλες που υλοποιούν την ανταλλαγή των τιμών για κάθε bit της εξόδου της μονάδας. Συνολικά, λοιπόν απαιτούνται $Comp + 6 \cdot L + 6 \cdot B + 6 \cdot logn$ πύλες. Το $logn$ είναι η υπόθεσή μας για το μέγεθος σε bits των wire id's. Μια τυπική



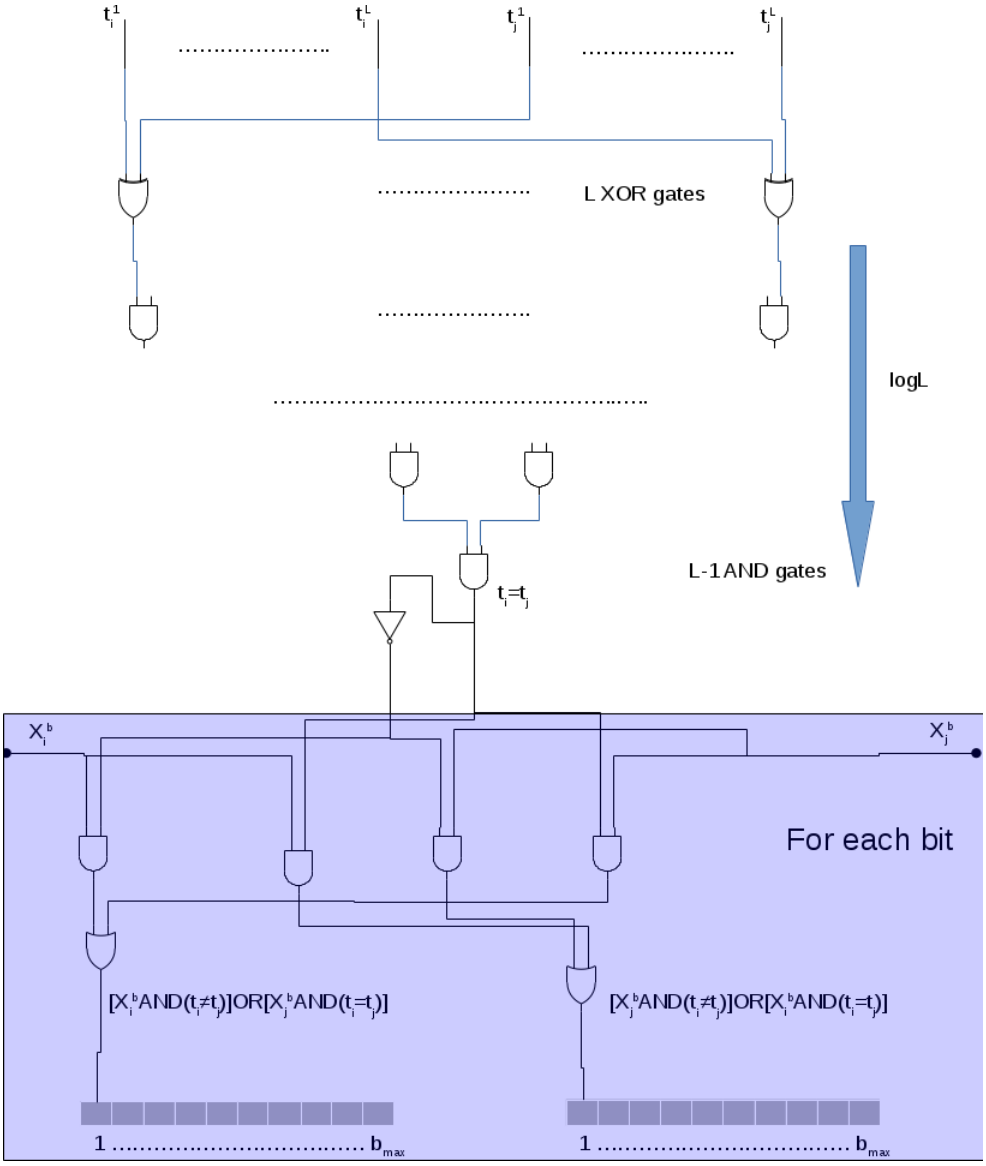
Σχήμα 3.5: compare and exchange module A

τιμή για τον αριθμό πυλών που απαιτούνται για ένα συγκριτή 128 bits, αποτελούμενο από Bristol συγκριτές 32 bits [34], είναι 621 πύλες AND με βάθος 25. Το βάθος έτσι της μονάδας διαμορφώνεται σε 27 πύλες AND. Όπως είπαμε και πριν αυτές οι τιμές δεν είναι βέλτιστες αλλά μας δίνουν μια πρώτη προσέγγιση στον αριθμό πυλών που απαιτούνται. Συνολικά, λοιπόν, υποθέτοντας $L = B = 128$ και $n_{max} = 2^{24}$, θα χρειαστούμε 2301 πύλες AND σε βάθος 27.

3.4.2 Μονάδα equality-test-and-exchange B

Η δεύτερη μονάδα που θα χρειαστούμε είναι απλούστερη από την πρώτη και παρουσιάζεται στο σχήμα 3.6. Αυτή η μονάδα παρέχει τη λειτουργία του ελέγχου ισότητας των t συντεταγμένων δύο τριάδων και την εναλλαγή των τριάδων σε περίπτωση επιτυχίας του ελέγχου. Ο ελεγκτής ισότητας κατασκευάζεται απλά και απαιτεί L πύλες AND σε βάθος $logL$. Επίσης, για την εναλλαγή των τιμών θέλουμε επιπλέον έξι πύλες για κάθε bit των wire id's και των m_i's (τα t_i's αγνοούνται). Έτσι η

μονάδα έχει βάθος $logL + 2$ και συνολικό αριθμό πυλών $L + 6 \cdot logn + 6 \cdot B$.



Σχήμα 3.6: check equality and exchange module B

3.4.3 Μονάδα compare-and-exchange C

Η μονάδα που απαιτείται για το τελευταίο στάδιο του κυκλώματός μας, δηλαδή για το στάδιο της ταξινόμησης των δυάδων της μορφής (i, m_j) με βάση την πρώτη τους συντεταγμένη (wire id), είναι παρόμοια με την μονάδα A, με τη διαφορά ότι εδώ απαιτείται ένας συγκριτής $log(n_{max})$ bits και επιπλέον απαιτείται η δυνατότητα εναλλαγής μόνο των m_i. Έτσι, το συνολικό κόστος της μονάδας, υποθέτοντας $n_{max} = 2^{32}$ και χρησιμοποιώντας απλούς συγκριτές των 32 bits, είναι $150 + 6B$ με βάθος 23 πύλες AND.

3.4.4 Συνολικός αριθμός πυλών και ένας πιο αποδοτικός συγκριτής

Ενώ οι αναπαραστάσεις των κυκλωμάτων στα προηγούμενα μέρη υποθέτουν τη χρήση απλών συγκριτών τεσσάρων bits, μπορούμε να έχουμε πιο αποδοτικά κυκλώματα χρησιμοποιώντας ειδικούς

συγκριτές όπως αυτόν του Garay που παρουσιάζεται στο [37] και για ακεραίους μεγέθους l bits, έχει λογαριθμικό βάθος $\lceil logl \rceil + 1$ και συνολικό μέγεθος $3l - \lceil logl \rceil - 2$ πύλες. Υπενθυμίζουμε ότι μας ενδιαφέρουν αποκλειστικά οι πύλες AND, καθώς οι πύλες XOR και INV (NOT) υπολογίζονται σχεδόν δωρεάν στα περισσότερα πρωτόκολλα SMC. Τώρα, υποθέτοντας μήκος σε bits $L = 128$ για τα t, ένας τέτοιος συγκριτής απαιτεί 375 πύλες συνολικά και έχει βάθος 8. Ένας τέτοιος συγκριτής θα αντικαταστήσει τον απλό συγκριτή των 128 bits της μονάδας A στους υπολογισμούς μας από εδώ και στη συνέχεια. Αντίστοιχα, για τον συγκριτή 32 bits που απαιτείται για τη μονάδα C, απαιτούνται 89 πύλες σε βάθος 6, χρησιμοποιώντας τον βελτιωμένο συγκριτή. Λαμβάνοντας υπ'όψιν όλα τα παραπάνω, παρουσιάζουμε το κόστος σε πύλες AND της κάθε μονάδας της κατασκευής μας στον παρακάτω πίνακα:

Module	AND gate depth	AND gate total
A	10	2103
B	9	1088
C	8	857

Πίνακας 3.1: Κόστος μονάδων

Για το συνολικό βάθος του κυκλώματος, υποθέτοντας μέγιστο αριθμό χρηστών- εισόδων $n < 2^{32}$, έχουμε:

$$D = \frac{log^2 n + logn}{2} \cdot 10 + 2 \cdot 9 + \frac{log^2 n + logn}{2} \cdot 8$$
$$D = 9log^2 n + 9logn + 18$$

Ο πρώτος όρος του αρχικού αθροίσματος αντιπροσωπεύει το βάθος του πρώτου σταδίου του κυκλώματὸς μας, που αποτελείται από $\frac{log^2 n + logn}{2}$ συγκριτές βάθους 10 ο καθένας. Ο δεύτερος μετράει το βάθος του δεύτερου σταδίου, με τους ελέγχους ισότητας να πρέπει να εκτελεστούν σε δύο στάδια, ώστε να μην έχουμε conflicts. Αντίστοιχα, ο τρίτος και τελευταίος όρος εκφράζει το βάθος του τρίτου σταδίου.

Για το συνολικό αριθμό πυλών AND που απαιτούνται έχουμε:

$$S = (n\frac{logn(logn-1)}{4} + n - 1) \cdot 2103 + (n-1) \cdot 1088 + (n\frac{logn(logn-1)}{4} + n - 1) \cdot 857$$
$$S = 740nlogn(logn - 1) + 4048n - 4048$$

Οι όροι του παραπάνω αθροίσματος προκύπτουν λόγω του γεγονότος ότι ένα δίκτυο ταξινόμησης n τιμών απαιτεί συνολικά $n\frac{logn(logn-1)}{4} + n - 1$ συγκριτές. Ως παράδειγμα, ας υποθέσουμε $n = 2^{16} = 65536$ χρήστες. Τότε, για το βάθος και το συνολικό μέγεθος του κυκλώματος, μετρώντας πύλες AND, έχουμε συνολικά $D = 2466$ και $S \approx 12$ δισεκατομμύρια πύλες αντίστοιχα.

3.5 Χρησιμοποιώντας κυκλώματα για secure multiparty computation

Έχοντας παρουσιάσει μια λύση στο πρόβλημα της ανταλλαγής μηνυμάτων, μέσω ενός δυαδικού κυκλώματος πολυπλοκότητας $\mathcal{O}(log^2 n)$ σε βάθος και $\mathcal{O}(nlog^2 n)$ σε συνολικό αριθμό πυλών, μπορούμε να χρησιμοποιήσουμε υπάρχοντα πρωτόκολλα SMC, όπως αυτά που αναφέρονται στο παράρτημα A.4 για να προχωρήσουμε προς μία υλοποίηση. Η μετατροπή των κυκλωμάτων σε ασφαλή πρωτόκολλα είναι ένα μεγάλο και πολύ ενεργό πεδίο έρευνας τα τελευταία χρόνια. Υπάρχει αρκετά

μεγάλη ποικιλία εργαλείων που υλοποιούν αυτή τη λειτουργία, δηλαδή δέχονται ως είσοδο μία αναπαράσταση του κυκλώματος, καθώς και τις εισόδους των χρηστών σε κάποια μοιρασμένη μορφή και παράγουν κώδικα που τρέχει στους servers. Κάποιοι τέτοιοι μηχανισμοί είναι το Sharemind [25], το fairplay [22], καθώς και το πιο πρόσφατο FRESCO [38]. Κάθε τέτοιος μηχανισμός προσφέρει τις δικές του μονάδες για σύγκριση, έλεγχο ισότητας και ίσως ακόμα και ταξινόμηση. Έτσι, το κύκλωμά μας, το οποίο βασίζεται κατά ένα μεγάλο μέρος στην ταξινόμηση, μπορεί εύκολα να αναπαρασταθεί σε οποιαδήποτε μορφή είναι συμβατή με το χρησιμοποιούμενο framework. Από πλευράς αποδοτικότητας, η πολυπλοκότητα της λύσης μας είναι παρόμοια με αυτή του προβλήματος της ιδιωτικής ταξινόμησης (oblivious sorting), και έχοντας δει αρκετά ενθαρρυντικά αποτελέσματα τα τελευταία χρόνια [39], είμαστε αισιόδοξοι ότι μια αρκετά αποδοτική υλοποίηση είναι εφικτή.

Συγκριτικά με τις υπάρχουσες λύσεις, η προσέγγισή μας εγγυάται κρυπτογραφική ασφάλεια, όταν υπάρχει τίμια πλειοψηφία (honest majority) για τους servers στο semi-honest μοντέλο, απέναντι σε κάθε αντίπαλο που έχει εποπτεία ολόκληρου του δικτύου, μπορεί να καθυστερήσει ή να μπλοκάρει όποια μηνύματα θέλει, και επιπλέον μπορεί να ελέγξει με οποιοδήποτε τρόπο όποιους χρήστες επιθυμεί εκτός από τους δύο που θέλουν να επικοινωνήσουν ανώνυμα.

3.6 Η αρχιτεκτονική ενός πραγματικού συστήματος

Αφού έχουμε παρουσιάσει τον τρόπο με τον οποίο επιτελείται η ανταλλαγή μηνυμάτων από ένα σύστημα που αποτελείται από ένα σύνολο από servers (3 στην περίπτωση που χρησιμοποιήσουμε το framework του sharemind) που τρέχουν ένα πρωτόκολλο ασφαλούς αποτίμησης της αντίστοιχης συνάρτησης, πρέπει να αποσαφηνίσουμε μερικά σημεία που αφορούν την υλοποίηση του συστήματος στο σύνολό του. Αρχικά μπορούμε να χρησιμοποιήσουμε ένα server εισόδου (entry server) στον οποίο θα στέλνουν οι χρήστες τα shares τους κρυπρογραφημένα με το κλειδί του κάθε SMC server για τον οποίο προορίζονται. Αυτός ο entry server θα συγκεντρώνει τα μηνύματα του κάθε γύρου, θα αναθέτει σε κάθε χρήστη ένα input wire και θα προωθεί τα shares στους servers για να υπολογίσουν τη συνάρτηση. Το input wire του κάθε χρήστη μπορεί να αποφασίζεται με βάση τη λεξικογραφική σειρά των ονομάτων των χρηστών ή με τη σειρά με την οποία έρχονται τα μηνύματα στον entry server. Εδώ σημειώνεται οτι αυτός ο server θα είναι επιφορτισμένος με τη δημιουργία των shares του input wire id. Αυτά τα shares μπορούν να είναι διαθέσιμα δημόσια σε plaintext χωρίς να επηρεάζεται η ασφάλεια του συστήματος, αφού στη συνέχεια και κατά την ταξινόμηση, αυτές οι τιμές θα τυχαιοποιηθούν.

Για να αποφύγουμε την κρυπτογράφηση δημοσίου κλειδιού για τα shares, μπορούμε να κρατάμε τα ονόματα των χρηστών που στέλνουν τα shares και ο κάθε SMC server να συμφωνεί μία φορά σε ένα μοιραζόμενο κλειδί με κάθε χρήστη όταν αυτός μπαίνει στην κατάσταση online. Έτσι, στη συνέχεια τα shares θα κρυπτογραφούνται με μεθόδους κρυπτογράφησης δημοσίου κλειδιού ώστε να επιτύχουμε καλύτερη αποδοτικότητα του συστήματος.

Μετά την ασφαλή αποτίμηση της συνάρτησης, οι SMC servers θα κρυπτογραφούν τα shares του αποτελέσματος με τα αντίστοιχα κλειδιά των χρηστών και θα τα επιστρέφουν στον entry server, ο οποίος με τη σειρά του θα τα προωθεί στους χρήστες στους οποίους αντιστοιχούν. Η όλη διαδικασία μπορεί να γίνει pipelined, δηλαδή ο entry server να δέχεται και να διαχειρίζεται τα αιτήματα των χρηστών προετοιμάζοντας την εκτέλεση του επόμενου γύρου επικοινωνίας ενώ οι SMC servers εκτελούν το πρωτόκολλο. Επίσης, οι SMC servers μπορούν να αποκρυπτογραφούν τα shares που δέχονται για χρήση στον επόμενο γύρο παράλληλα με την εκτέλεση του πρωτοκόλλου στον τρέχοντα γύρο.

Τέλος, όσον αφορά τον τρόπο με τον οποίο θα εκτελείται το dialing κομμάτι του πρωτοκόλλου, δηλαδή τον τρόπο με τον οποίο κάποιος χρήστης θα ειδοποιείται για την επιθυμία κάποιου άλλου χρήστη να αρχίσει μια συζήτηση μαζί του, προτείνουμε μια απλή λύση. Όλα τα αιτήματα τέτοιου είδους θα κρυπτογραφούνται από κάθε dialer με το δημόσιο κλειδί του dialee και θα δημοσιοποιούνται σε τακτά χρονικά διαστήματα (dialing rounds) σε ένα bulletin board (δημόσια βάση δεδομένων). Για την ανάκτηση των κλήσεων που απευθύνονται για αυτούς, οι χρήστες θα κατεβάζουν ολόκληρη

τη βάση και θα ελέγχουν εάν υπάρχουν αιτήματα που μπορούν να αποκρυπτογραφήσουν. Αυτός ο τρόπος είναι πολύ αργός αλλά δεδομένου ότι το dialing round μπορεί να είναι αρκετά μεγάλο σε χρονική έκταση, είναι μια αποδεκτή λύση. Άλλοι τρόποι εκτέλεσης του dialing, πιθανώς με τη χρήση τεχνικών PIR [41] (Private Information Retrieval) μπορούν να ερευνηθούν με στόχο τη μείωση του κόστους της συγκεκριμένης λειτουργίας.

Κεφάλαιο 4

Συμπεράσματα και μελλοντικές κατευθύνσεις

Σε αυτή την εργασία αφού μελετήσαμε το πρόβλημα της ανώνυμης επικοινωνίας με τη μορφή της ανταλλαγής μηνυμάτων και διάφορες πιθανές λύσεις του προτείναμε μια δική μας λύση που βασίζεται στον υπολογισμό μιας συνάρτησης μέσω πρωτοκόλλων Secure Multiparty Computation από έναν αριθμό από servers. Η λύση μας εκφράζεται ως ένα κύκλωμα δυαδικών πυλών βάθους $\mathcal{O}(log^2 n)$ και συνολικού μεγέθους $\mathcal{O}(nlog^2 n)$. Με τη χρήση SMC πρωτοκόλλων μπορούμε να λύσουμε το πρόβλημα με information-theoretic ασφάλεια στο semi-honest μοντέλο (αρχικά), κάτι που είναι σημείο υπεροχής της προσέγγισής μας σε σχέση με την υπάρχουσα κατάσταση. Σημαντικό ερώτημα όμως παραμένει η πρακτικότητα της λύσης μας, καθώς παραμένει ακόμα ένα θεωρητικό κατασκεύασμα. Προς αυτή την κατεύθυνση πρέπει να κινηθεί η έρευνα από εδώ και πέρα. Ενώ η λύση μας είναι σωστή είναι πολύ πιθανό να υπάρχουν βελτιστοποιήσεις σε διάφορα σημεία του κυκλώματος που να επιφέρουν ένα μικρότερο κύκλωμα και κατά συνέπεια ένα αποδοτικότερο πρωτόκολλο. Επίσης, προφανές επόμενο βήμα στην έρευνά μας είναι η υλοποίηση ενός τέτοιου πρωτοκόλλου σε κάποια πλατφόρμα SMC (πιθανώς την Sharemind) ώστε να λάβουμε πειραματικά αποτελέσματα που θα μας επιτρέψουν τη σύγκριση της λύσης μας με τις υπόλοιπες της βιβλιογραφίας. Μετά από αυτά τα βήματα θα είμαστε σε θέση να συμπεράνουμε εάν το κατασκεύασμά μας έχει τα απαραίτητα χαρακτηριστικά ώστε να οδηγήσει σε ένα πρακτικό σύστημα.

Παράρτημα Α

Χρήσιμα εργαλεία

Α.1 Μετρικές απόστασης

In this section, we will define some distance measures between distributions, that are going to be useful in the following chapters. The definitions are taken from [3].

Definition A.1. (KL-Divergence). The KL-Divergence, or Relative Entropy, between two random variables Y and Z taking values from the same domain is defined to be:

$$D(Y||Z) = \mathbb{E}_{y\sim Y}\left[ln\frac{Pr[Y=y]}{Pr[Z=y]}\right]$$

It is known that $D(Y||Z) \geq 0$, with equality if and only if Y and Z are identically distributed. However, D is not symmetric, does not satisfy the triangle inequality, and can even be infinite, specifically when $Supp(Y)$ is not contained in $Supp(Z)$.

Definition A.2. (Max Divergence). The Max Divergence between two random variables Y and Z taking values from the same domain is defined to be:

$$D_\infty(Y||Z) = \max_{S\subseteq Supp(Y)}\left[ln\frac{Pr[Y\in S]}{Pr[Z\in S]}\right]$$

The δ-Approximate Max Divergence between Y and Z is defined to be:

$$D_\infty^\delta(Y||Z) = \max_{S\subseteq Supp(Y):Pr[Y\in S]\geq\delta}\left[ln\frac{Pr[Y\in S]-\delta}{Pr[Z\in S]}\right]$$

Definition A.3. (Statistical Distance). The Statistical Distance between two random variables Y and Z is defined to be:

$$\Delta(Y,Z) = \max_S |Pr[Y\in S] - Pr[Z\in S]|$$

We say that Y and Z are δ-close if $\Delta(Y,Z)\leq\delta$.

Α.2 Μοντέλα αντιπάλου

When arguing about the security of a given protocol, it is very important to specify what an adversary can possibly do to break this security. In this thesis, all adversarial actions in a given protocol will be modeled as a single entity-adversary that can corrupt one or more parties, observe their input/output and/or control their behavior. That is, any coalition of dishonest parties that may collude in order to compromise the security of a given protocol, will be viewed as a single adversary.

Generally, we can talk about two classes of adversarial behavior:

- A *passive* adversary can only observe the inputs, outputs and intermediate values of a corrupted party, however the party still follows the protocol. Such a party is often called *semi-honest* or *honest-but-curious*. An adversary following the semi-honest model will try to break the secrecy of a protocol by trying to extract information from the values that are available to him while executing the protocol without deviations.

- An *active* adversary totally controls the corrupted party and actively tries to manipulate the protocol in order to render it insecure.

The adversarial model used also depends on the computational resources available to him in terms of time and storage. There are *computationally-bounded* (most often polynomially-bounded) and unbounded adversaries. A classification also occurs between *static* and *adaptive* adversaries. The former can corrupt a given subset of parties at the beginning of the protocol and cannot change this subset during the execution, while the later can adaptively change the subset of corrupted parties while the protocol is underway.

Finally, in the context of network security, an adversary can be *local* or *global* depending on his view of the network. A local adversary can only view network traffic in a small portion of the network (e.g. a wi-fi access point), whereas a global adversary can view network traffic in a significantly larger portion of the network (e.g. ISP, governments).

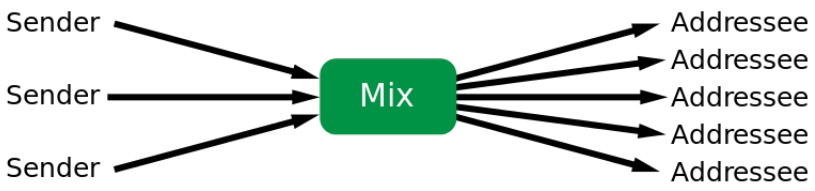
In this thesis, we will typically have to do with computationally bounded, static, global adversaries. In client-server models, clients can be actively corrupted, while servers can be semi-honest. This means that we will generally not encounter a server that wants to cause a failure in the protocol. However, even if a server does not follow the protocol, he cannot extract additional information, but can only cause the protocol to malfunction.

A.3 Mixnets

Mixnets or Mix networks, introduced by David Chaum in [4], are a very useful tool for achieving anonymity in many applications. In this section we will present the basic concepts of mixnets that will be useful in the next chapters.

A.3.1 Definition

A mixnet is a multistage system that uses cryptography and permutations to provide anonymity [5]. It accepts an input batch of quantities and produces an output batch containing the cryptographically transformed, permuted input batch.



Σχήμα A.1: Simple decryption mix net

Its aim is to hide the relation between the sender and the receiver of a given message.

A.3.2 Applications

Mixnets are used in a variety of setting including:

- E-voting
- Anonymous e-mail
- Anonymous Telecommunications
- Digital Currencies

A.3.3 Mixnet Properties

All mixnets should at some degree satisfy the Security, Performance and Implementation requirements listed below [5] :

A. Security requirements: Mixnets have as a primary purpose to offer certain security guarantees to the participating parties. These can be summarized by the points below.

1. *Anonymity* : The mixnet is designed to provide untraceability (unlinkability) as the basic property between senders and receivers [1]. No one should be able to trace any input through the stages of the mixnet. This notion of untraceability is captured by an anonymity set [7], [8], and the level of anonymity can be quantified by using entropy of the anonymity set probability distribution [9, 10].

2. *Integrity* : The messages from the senders have to be anonymized by the mixnet without any corruption of the data. An integrity check is provided in most mixnets by verification mechanisms and checksum techniques.

3. *Verifiability* : The mixnet must provide some means for verification of the correctness of the messages in the mixed output batch. This is most often accomplished by Zero Knowledge proofs or Randomized Partial Checking [6].

4. *Robustness against attacks* : The mixnet must be robust against passive, as well as active, attacks by an adversary. Various mechanisms are employed in a mixnet (both by the communicating sender and receiver, as well as the stages) to ensure robustness.Note that the verifiability property is related to robustness against some types of active attack, since verifiability ensures honest participation of each stage.

5. *Fault-tolerance* : The mixnet must be able to tolerate a certain number of faulty stages among the participating stages during its operation.

B. Performance requirements: Security and performance of a mixnet are naturally inversely proportional. The performance metrics used to evaluate mixnets can be listed as follows:

1. *Latency* : The processing at each stage and the verifiability and other robustness mechanisms in the mixnet take a finite amount of time, hence, adding to delay in the communications. Low latency is crucial for real-time applications requiring anonymity (like anonymous instant messaging). However, most of the security properties contradict latency, especially under low-traffic conditions.

2. *Throughput* : This is a measure of the number of actual sender messages a mixnet can output per unit of time. It provides an estimate of the overhead due to the mixnet. The overhead can be mainly due to any traffic padding or dummy messages employed for robustness by the mixnet.

C. Implementation Requirements: While providing security and performance guarantees, the mixnet must also be implementable. More specifically the following requirements must be addressed.

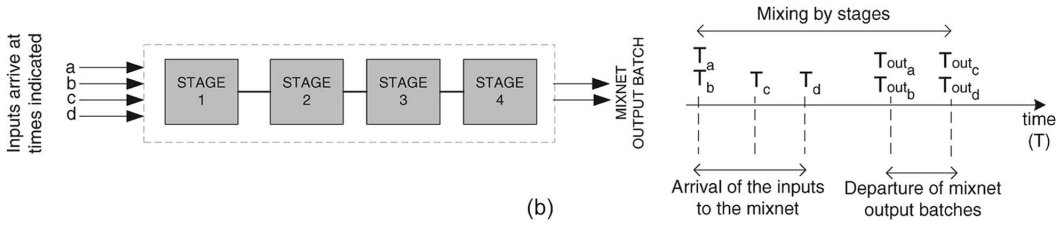
1. *Scalability* : In a mixnet, the level of anonymity can be enhanced by increasing the number of participants (increase in the batch size) and/or increasing the number of stages in the anonymous path. However, scalability of the mixnet, with respect to the number of inputs and stages, is mainly limited by the latency requirement.

2. *Efficiency* : The mix network protocol must minimize complex computations and communications. The computational complexity is measured mainly in terms of modular exponentiations or number of cryptographic operations required in the protocol computations. Communication complexity is often measured in terms of the number of interactions needed between entities participating in the protocol.

It is important to notice that for each application, some of the above requirements are much more important than others. For example, in the e-voting setting, very strong security and robustness requirements must be guaranteed but the latency of the system is of secondary concern. On the other hand, in the implementation of a real-time messaging service, low latency is necessary, sometimes with a relaxation of security and robustness guarantees.

A.3.4 Mixnet Topologies

Mixnets generally consist of several mix servers that process data sequentially. For the interest of this thesis, we will consider only the *cascade* topology, where messages are grouped together to form batches, and all messages in a batch follow the same (agreed) path along a fixed number of mix nodes with the same timing. In the remainder of this thesis we will assume that by referring to a mixnet we refer to this topology. Mixnets that follow this topology are considered secure if one of the mix servers follows the protocol.



Σχήμα A.2: The cascade topology

A.3.5 Mixnet Types

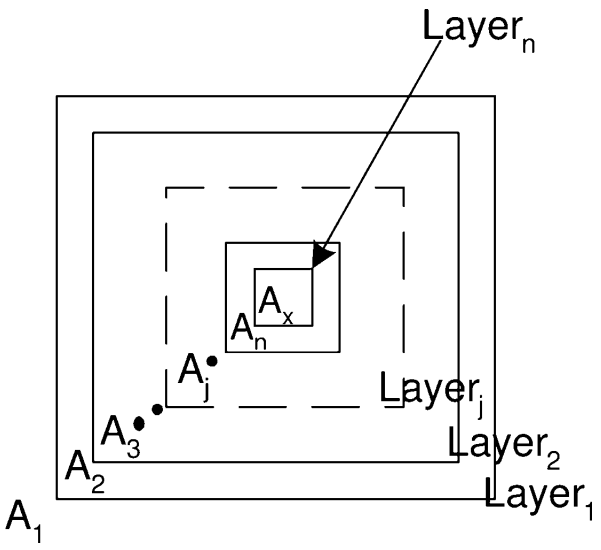
A. Decryption Mixnet:

Decryption mixnets were introduced in [4] and are the most popular mixnets with many applications. In decryption mixnets, the sender is required to encrypt the message with the keys of the stages, thus forming an onion.

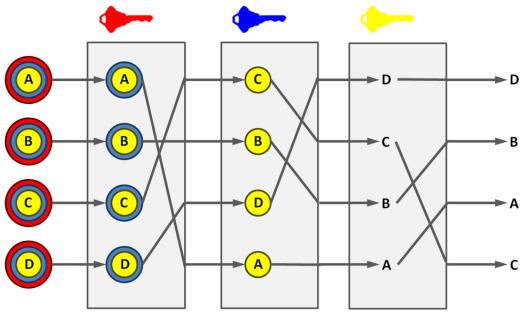
Hence, a stage can change the appearance of its inputs by decrypting with its key. This can be accomplished with the use of public key cryptosystems such as RSA, ElGamal.

B. Hybrid Mixnet:

The Hybrid mixnet [14, 15] is a more efficient variant of the Decryption mixnet. The functionality is generally the same, except that in the Hybrid mixnet, public key encryption (e.g. Diffie-Helman [13]) is used only to share a symmetric key with which the remaining payload of the message is encrypted.



Σχήμα A.3: A decryption mixnet onion



Σχήμα A.4: A decryption mixnet

Thus, the computational burden of public key encryption is drastically limited.

C. Reencryption Mixnet:
As opposed to the work done by a mix server in a decryption mixnet, the role of which is to partially decrypt and permute its inputs, that of a mix server in a re-encryption based mixnet [12] differs as it now re-encrypts and permutes its input ciphertexts. The El Gamal cryptosystem, among others, offers this nice re-encryption property. In contrast to a decryption mixnet, where the input messages are successively encrypted using the public keys of the individual mix servers, here, the inputs are encrypted under the joint public key of the mix servers and then submitted to the mixnet.

D. Variations:
Variations of the above basic mixnet concepts, such as the Universal Re-encryption Mixnet [16], exist but are out of the scope of this thesis.

A.4 Secure Multiparty Computation

A.4.1 What it is

Secure Multiparty Computation (SMC or MPC), is an area of cryptography concerned with methods and protocols that enable a set of users $u_1, \ldots, u_n$ with private data $d_1, \ldots, d_n$ to compute the result of a public function $F(d_1, \ldots, d_n)$, without revealing their private inputs. Secure computation was

formally introduced as secure two-party computation (2PC) in 1982 by Andrew Yao,[21] the first recipient of the Knuth Prize, and was soon expanded to the multi-party setting. There exist several generic SMC constructions that receive as input a description of an algorithm (in some form) and the distribution of the inputs among the data owners and produce as output the description of a secure protocol that implements the algorithm in a privacy-preserving manner. In most cases, some form of secret sharing of the inputs, such as additive or Shamir secret sharing [24] is used and the protocol proceeds to produce a sharing of the output. Most known SMC frameworks, such as Fairplay [22], VIFF [27], Sharemind [25] etc. need the function as a circuit, either made up of boolean gates or as an arithmetic circuit over a sufficiently large field $GF(p)$. This is a highly non-trivial matter as most useful functions use loops or recursion. Generally, each implementation follows either the Yao paradigm, the GMW [23] or some combination of those. The main difference of these two approaches is that Yao's approach required the generation of a garbled circuit and the evaluation of the function on it, whereas the GMW approach requires communication during the evaluation of any multiplication (or binary AND) gate.

A.4.2 Security in SMC

Generally SMC protocols can be information-theoretically secure in the passive (semi-honest) adversary model for $t < n/2$ corrupted parties. That is, for unconditional security an honest majority is needed. On the other hand, using cryptographic security assumptions protocols can be constructed that can tolerate any number of semi-honest adversaries. Concerning active adversaries, SMC protocols can be constructed that can provide cryptographic security against $t < n/2$ corrupted parties.

A.4.3 The GMW protocol simplified

The GMW protocol, introduced by Goldreich,Micali and Wigderson in 1987 is an SMC protocol enabling the joint computation of any boolean circuit and therefore any boolean function. Concerning security, GMW was originally built to tolerate only passive adversaries but was soon expanded in order to function against malicious adversaries as well. The protocol works by sharing each input bit into n shares, where n is the number of the protocol's participants. This sharing can be a simple additive secret sharing $mod2$ (more on this in the example below). Then, the parties involved compute the gates at each level of the boolean circuit jointly. It has to be noted that gates at the same level of the circuit (all their input wires are known at the same round of the protocol) can be computed in parallel given adequate communication bandwidth between the servers, thus making the depth of the circuit the most important performance parameter. As we will see in our example, XOR and NOT gates can be computed basically for free, as they are linear functions on the input shares and therefore the output shares will remain a valid sharing of the result of the gate. Only AND gates require a special protocol to run in order to be evaluated. This sub-protocol is named 1-4 Oblivious Transfer [35] and makes it possible for a receiver to receive one of four values held a sender without the sender knowing which value has been sent and without the receiver knowing the other three values which were not sent. It has to be noted that OR gates aren't a separate concern, as they can easily be constructed using AND and NOT gates. At the end of the execution of the protocol, each of the final output wires will contain a share of the function's output. These shares can be sent to the desired parties that will simply add them ($mod2$) and reconstruct the output of the function.

A.4.4 A simple 2-party example

Let us assume that we have two parties P_1 and P_2 who are in possession of two inputs x and y respectively and want to securely compute a function of these inputs. The parties may not necessarily be in possession of the inputs x and y, but can alternatively be in possession of their a and b shares.

The desired function has been represented as a boolean circuit C consisting of XOR,AND and NOT gates. The protocol proceeds as follows:

- **Initialization:** $x \to \{x_1, \ldots, x_k\}$, where x_i denotes the i-th bit of x. Similarly $y \to \{y_1, \ldots, y_k\}$. Both parties start by splitting their input bits into a and b shares, with the former held by P_1 and the latter by P_2. For every input wire w, its value is denoted by $p_w = a_w \oplus b_w$. The splitting can be done the following way:
  $\forall w \in \{w_1, \ldots, w_k\}$
  Sample $a^x_{w_i} \leftarrow \{0, 1\}$
  $b^x_{w_i} = x_i \oplus a^x_{w_i}$
  Similarly, P_2 splits y into $a^y_{w_i}$ and $b^y_{w_i}$, where $i \in \{1, \ldots, k\}$. After the split is done, P_1 keeps all of the a_w shares and P_2 keeps the b_w shares. At this stage, $x_i = a^x_{w_i} \oplus b^x_{w_i}$ and $y_i = a^y_{w_i} \oplus b^y_{w_i}$.

- **Execution:** Given a gate G, we want to evaluate $G(x_i, y_i)$ without P_1 learning the value of y_i and P_2 learning the value of x_i.

  XOR gate
  To evaluate $x_i \oplus y_i$, we can simply have the parties compute their shares separately because $x_i \oplus y_i = (a^x_{w_i} \oplus b^x_{w_i}) \oplus (a^y_{w_i} \oplus b^y_{w_i}) = (a^x_{w_i} \oplus a^y_{w_i}) \oplus (b^x_{w_i} \oplus b^y_{w_i})$.

  NOT gate
  For the NOT gate, we wish to evaluate the inverse of an input $w_i = a_{w_i} \oplus b_{w_i}$. In order to achieve this, we can simply flip the output bit held by one of the two players. This player can be decided in advance and be the same for all NOT gates.

  AND gate
  In this situation, we need to compute $x_i \wedge y_i = (a^x_{w_i} \oplus b^x_{w_i}) \wedge (a^y_{w_i} \oplus b^y_{w_i})$. We would like each party to finish the evaluation with one bit each, but in the meantime learning nothing about the other party's initial share. In order to achieve this, we will make use of the 1-4 OT protocol.
  Let $T = x_i \wedge y_i$. P_1 randomly samples $\sigma \leftarrow \{0, 1\}$. Then, the following table (A.1) is constructed privately by P_1:

b_1	b_2	$T = (a^x_{w_i} \oplus b_1) \wedge (a^y_{w_i} \oplus b_2)$	
0	0	α_0	$s_0 = \sigma \oplus \alpha_0$
0	1	α_1	$s_1 = \sigma \oplus \alpha_1$
1	0	α_2	$s_2 = \sigma \oplus \alpha_2$
1	1	α_3	$s_3 = \sigma \oplus \alpha_3$

Πίνακας A.1: 1-4 OT matrix

  Then, P_2 uses 1-4 OT to retrieve the correct result, that is the result that satisfies $b_1 = b^x_{w_i}$ and $b_2 = b^y_{w_i}$. The details of the 1-4 OT protocol will not be presented as part of this thesis. However, it is possible to make use of preprocessing and make the 1-4 OT protocol very efficient.

The example described above gives us a rough guide as to how SMC protocols work. On the part of the circuit designer, efficiency generally depends on the depth of the circuit counting only AND gates and to a lesser extent on the total number of AND gates.

A.5 Δίκτυα ταξινόμησης

A.5.1 Introduction

Sorting networks are circuits that solve the sorting problem on any set with an order relation. Without loss of generality, we shall only consider sorting of integers. Sorting networks are devices

built up only of wires carrying values and comparator modules that connect pairs of wires and that swap these values if they are not in the desired order (according to a given order relation). What sets sorting networks apart from general comparison sorts is that their sequence of comparisons is set in advance, regardless of the outcome of previous comparisons. They were first introduced by Armstrong, Nelson and O'Connor in the 1950's [17] and surprisingly research has not stopped since, intensified by their use in modern parallel hardware architectures and Secure Multiparty computation. Various algorithms exist to construct simple and efficient networks of depth $\mathcal{O}(log^2 n)$ and size $\mathcal{O}(nlog^2 n)$. The three more used ones are Batcher's odd-even mergesort and Bitonic sort [18] and Shellsort [19]. All three of these networks are simple in principle and efficient. Sorting networks that achieve the theoretically optimal $\mathcal{O}(log^n)$ and $\mathcal{O}(nlog^n)$ complexity in depth and total number of comparisons, such as the AKS-network [20] exist, but the constants involved are so large that make them impractical for use.

A.5.2 Batcher's odd-even mergesort

Batcher's odd-even mergesort algorithm, in contrast to the traditional mergesort is not data-dependent, that is the same comparisons are performed regardless of the input. The algorithm *odd-even mergesort* and the called procedure *odd-even merge* is described below:

Algorithm 1 Odd-even merge

Input: sequence $a_0, \ldots, a_{n-1}$ whose two halves $a_0, \ldots, a_{n/2-1}$ and $a_{n/2}, \ldots, a_{n-1}$ are sorted (n is a power of 2)
Output: the sorted sequence of length n

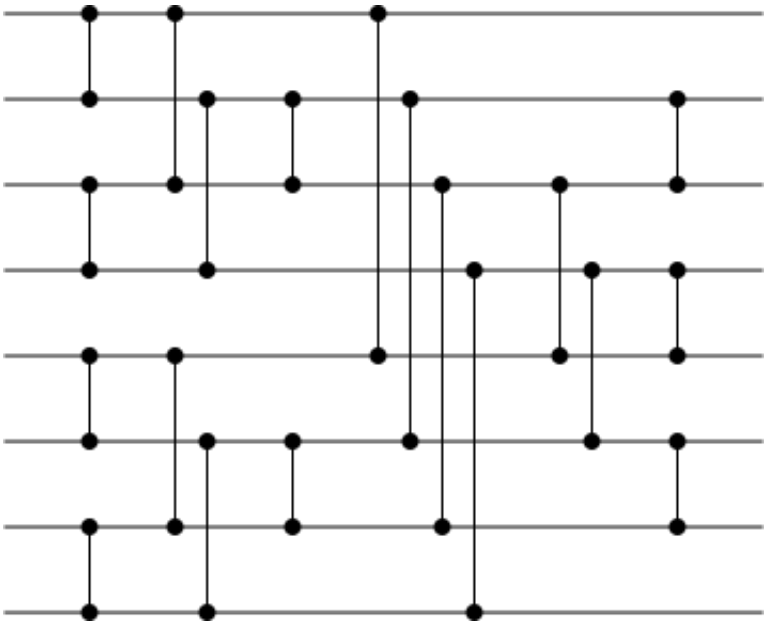
if $n > 2$ **then**
1. apply *odd-even merge (n/2)* recursively to the even subsequence $a_0, a_2, \ldots, a_{n-2}$ and the odd subsequence $a_1, a_3, \ldots, a_{n-1}$
2. comparison [i:$i+1$] for all $i \in \{1, 3, 5, 7, \ldots, n-3\}$
else
comparison [0:1]
end if
return sequence $a_0, \ldots, a_{n-1}$

Algorithm 2 Odd-even mergesort

Input: sequence $a_0, \ldots, a_{n-1}$, n a power of 2.
Output: the sorted sequence of length n

if $n > 1$ **then**
1. apply *odd-even mergesort(n/2)* recursively to the two halves $a_0, \ldots, a_{n/2-1}$ and $a_{n/2}, \ldots, a_{n-1}$ of the sequence
2. *odd-even merge (n)*
end if
return sequence $a_0, \ldots, a_{n-1}$

The exact number of comparators needed to sort n elements is: $n - 1 + \frac{nlogn(logn-1)}{4}$ with depth $\frac{logn(logn+1)}{2}$. We could say that this merge sort network needs $\frac{log^2 n}{2}$ steps of $\frac{n}{2}$ comparators each. This makes Batcher's network the most efficient of the three proposed. An example of the network for 8 inputs is shown in figure A.5.



Σχήμα A.5: example of Batcher's mergesort network

A.5.3 Correctness of Batcher's algorithm

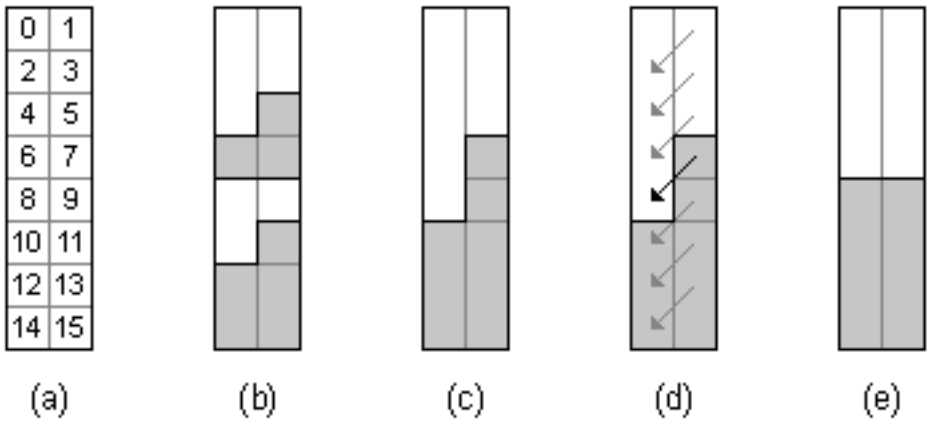
Correctness of Batcher's sorting network can be proven based on the 0-1 principle of Knuth (proof in [36]) that states that if a network can sort every sequence of 0's and 1's, then it sorts every arbitrary sequence of values. We will prove the correctness of the merge algorithm by induction (proof presented as in [30]).

If $n = 2^1$, the sequence is sorted by the comparison [0:1]. Now for $n = 2^k$, $k > 1$ we assume that the algorithm is correct for all smaller k (induction hypothesis).

Now consider the 0-1 sequence $\alpha = \alpha_0, \ldots, \alpha_n - 1$ to be arranged in rows of an array with two columns. The corresponding mapping of the index positions is shown in figure A.6(a), for $n = 16$. Then, figure A.6(b) shows a possible situation with a 0-1 sequence. Each of its two sorted halves starts with some 0's (white) and ends with some 1's (gray). In the left column, the even subsequence is found, that is all the a_i with i even, namely $a_0, a_2, a_4, \ldots$. In the right column the odd subsequence is found, that is all a_i with i odd, namely $a_1, a_3, a_5, \ldots$. Just like the original sequence, the even as well as the odd subsequence consists of two sorted halves.

By induction hypothesis, the left and the right column are sorted by recursive application of *odd-even merge(n/2)* in step 1 of the algorithm. The right column can have at most two more 1's than the left column (figure A.6(c)).

Finally, after performing the comparisons of step 2 of the algorithm (figure A.6(d)), in each case the array is sorted (figure A.6(e)).



Σχήμα Α.6: sorting in steps

Παράρτημα Β

Διαφορική ιδιωτικότητα (Differential Privacy)

Β.1 Τι μπορούμε να κατορθώσουμε

In an anonymous messaging protocol, achieving strong-semantic privacy comes at the cost of large communication and computation overheads. (see broadcast-clique based protocols). In order to make such a system near real- time, we need an alternative, relaxed notion of privacy. One such notion, that intuitively fits the problem, is Differential Privacy [3]. In this chapter, we define and study differential privacy in the message-exchange setting.

Β.2 Plausible deniability.

Before formalizing the notion of Differential Privacy, let us see an example of what plausible deniability can offer taken from [3]. Randomized response is a technique developed in the social sciences to collect statistical information about embarrassing or illegal behavior, captured by having a property P. Study participants are told to report whether or not they have property P as follows:

1. Flip a coin.
2. If **tails**, then respond truthfully.
3. If **heads**, then flip a second coin and respond "Yes" if heads and "No", if tails.

In the above setting, Privacy comes from the *plausible deniability* of any outcome. In particular, if having property P corresponds engaging in illegal behavior, even a "Yes" answer is not incriminating, since this answer occurs with probability $1/4$ whether or not the respondent actually has property P. In the context of anonymous communication - message exchange, *plausible deniability* would mean that a user participating in such a protocol, will have plausible cover stories for her actions. This is what Differential Privacy comes to formalize.

Β.3 Η Διαφορική Ιδιωτικότητα τυπικά

Differential Privacy is not a new privacy notion. It has been used for years by database curators to assure people that statistical analysis of their records cannot expose any individual's private data. However, the use of Differential Privacy in the context of anonymous communication is a fairly new idea. That is why we will try to view the problem of message exchange as a problem of keeping the state matrix S private, while facilitating the exchange of messages.

Definition B.1. (Probability Simplex): Given a discreet set B, the *probability simplex* over B, denoted $\Delta(B)$, is defined to be:

$$\Delta(B) = \left\{ x \in \mathbb{R}^{|B|} : x_i \geq 0 \text{ for all i and } \sum_{i=1}^{|B|} = 1 \right\}$$

Definition B.2. (Randomized Algorithm): A *randomized algorithm* $\mathcal{M}$ with domain A and discreet range B is associated with a mapping $M : A \to \Delta(B)$. On input $\alpha \in A$, the algorithm $\mathcal{M}$ outputs $\mathcal{M}(\alpha) = b$ with probability $(M(\alpha))_b$ for each $b \in B$. The probability space is over the coin flips of the algorithm $\mathcal{M}$.

In the setting of anonymous messaging, we will think of databases x as instances of the binary state matrix S, as in Table 1.2 . From here on, we will refer to S as the set of all action matrices. Our *universe* $\mathcal{X}$ will be the set of all possible database entries, which is all possible rows of our binary action matrix S. It represents all possible actions a single user can perform. In the setting of n users, $\mathcal{X}$ is the set of the n combinations of $n-1$ zeroes and one 1.

Definition B.3. (Distance Between Databases): The distance between two databases $x, y \in S$ is defined to be:

$$||x - y|| = \frac{1}{2} \sum_{i=1,\ldots,n}^{j=1,\ldots,n} |A[i,j] - B[i,j]|$$

This distance shows how many users perform different actions in the two databases. For example, if $||x - y|| = 1$, only one user's action differs.

Definition B.4. (Differential Privacy): A randomized algorithm $\mathcal{M}$ with domain S is (ε, δ)-differentially private if for all $\mathcal{S} \subseteq Range(\mathcal{M})$ and for all $x, y \in S$ such that $||x - y|| \leq 1$:

$$Pr[\mathcal{M}(x) \in \mathcal{S}] \leq exp(\varepsilon) Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta$$

In the setting of anonymous communication, what differential privacy ensures is that given a run of the mechanism $\mathcal{M}(x)$, the output observed is (almost) equally likely to be observed on *every* neighboring database, *simultaneously*, with probability δ. Typically we are interested in very low values for δ, because a high value for δ will make it possible, given an output $\xi \sim \mathcal{M}(x)$, to find a database y, such that ξ is much more likely to be produced when the database is y than when the database is x.

Remark B.1. Viewing this definition along with the definitions of section A.1, we easily have that a mechanism $\mathcal{M}$ is:

1. ε-differentially private if and only if on every two neighboring databases x and y, $D_\infty(\mathcal{M}(x)||\mathcal{M}(y)) \leq \varepsilon$ and $D_\infty(\mathcal{M}(y)||\mathcal{M}(x)) \leq \varepsilon$.

2. (ε−δ)-differentially private if and only if on every two neighboring databases x and y, $D_\infty^\delta(\mathcal{M}(x)||\mathcal{M}(y)) \leq \varepsilon$ and $D_\infty^\delta(\mathcal{M}(y)||\mathcal{M}(x)) \leq \varepsilon$.

Finally, it is easy to prove the following combination theorem:

Θεώρημα 1. Let $\mathcal{M}_i : S \to R_i$ be an $(\varepsilon_i, \delta_i)$-differentially private algorithm for $i \in [k]$. Then if $\mathcal{M}_{[k]} : S \to \prod_{i=1}^k$ is defined to be $\mathcal{M}_{[k]}(x) = (\mathcal{M}_1(x), \ldots, M_k(x))$, then $\mathcal{M}_{[k]}$ is $(\sum_{i=1}^k \varepsilon_i, \sum_{i=1}^k \delta_i)$-differentially private.

B.4 Ο μηχανισμός Laplace

Differential privacy assumes the use of randomness on the output of mechanism $\mathcal{M}$. The Laplace mechanism uses randomness drawn from a Laplace distribution to achieve differential privacy. But before introducing the mechanism, we need to make some definitions.
We will view the information f_1, f_2 leaked to the adversary, as numeric queries on a database $x \in S$. Numeric queries are functions $f : S \to \mathbb{R}^k$. These queries map databases to k real numbers.

Definition B.5. (ℓ_1-sensitivity): The ℓ_1-*sensitivity* of a function $f : S \to \mathbb{R}^k$ is:

$$\Delta f = \max_{\substack{x,y \in S \\ ||x-y||=1}} ||f(x) - f(y)||.$$

The ℓ_1-sensitivity of a function-query f captures the magnitude by which a single individual's data can change the function f in the worst case. In our setting, it expresses the maximum difference of the information leaked to the adversary, between two scenarios differing only in the actions of a single user. In other words, it is a metric of how much the actions of a single user influence the observable variables of the system. Intuitively, we can assume that the higher the ℓ_1-sensitivity of a function, the more noise is needed to achieve a given degree of privacy.

Definition B.6. (The Laplace Distribution) : *The Laplace Distribution* centered at μ with scale b is the distribution with probability density function:

$$Lap(x|\mu, b) = \tfrac{1}{2b} exp(-\tfrac{|x-\mu|}{b})$$

The variance of this distribution is $\sigma^2 = 2b^2$. In most cases we will write $Lap(\mu, b)$ or even $Lap(b)$ when μ is zero.

We will now define the *Laplace Mechanism*. The mechanism introduces laplacian noise to each coordinate of the function f. The scale of the noise is calibrated to the sensitivity of f as will be shown below.

Definition B.7. (The Laplace Mechanism) : Given any function $f : S \to \mathbb{R}^k$, the *Laplace Mechanism* is defined as:

$$\mathcal{M}_L(x, f(.), \varepsilon) = f(x) + (Y_1, \ldots, Y_k)$$

where Y_i are random variables drawn from $Lap(\mu, \frac{\Delta f}{\varepsilon})$.

Θεώρημα 2. The Laplace Mechanism preserves (ε,0)-differential privacy.

Proof: Let $x, y \in S$ such that $||x - y|| \leq 1$, and let $f(.)$ be a function $f : S \to \mathbb{R}^k$. Let $\mathcal{M}_L(x, f, \varepsilon)$ denote the output of the Laplace mechanism for input database x, and $\mathcal{M}_L(y, f, \varepsilon)$ the output of the Laplace mechanism for input database y. We compare the two at some arbitrary point $z = (z_1, \ldots, z_k) \in \mathbb{R}^k$.

$$\frac{Pr[\mathcal{M}_L(x,f,\varepsilon)=z]}{Pr[\mathcal{M}_L(y,f,\varepsilon)=z]} = \prod_{i=1}^{k} \frac{Pr[f(x)_i + Y_i = z_i]}{Pr[f(y)_i + Y_i' = z_i]} = \prod_{i=1}^{k} \frac{Pr[Y_i = z_i - f(x)_i]}{Pr[Y_i' = z_i - f(y)_i]} =$$

$$\prod_{i=1}^{k} \left(\frac{exp(-\frac{\varepsilon|z_i - f(x)_i - \mu|}{\Delta f})}{exp(-\frac{\varepsilon|z_i - f(y)_i - \mu|}{\Delta f})} \right) = \prod_{i=1}^{k} \left(\frac{exp(-\frac{\varepsilon|z_i - (f(x)_i + \mu)|}{\Delta f})}{exp(-\frac{\varepsilon|z_i - (f(y)_i + \mu)|}{\Delta f})} \right) =$$

$$\prod_{i=1}^{k} exp\left(\varepsilon \frac{|z_i - (f(y)_i + \mu)| - |z_i - (f(x)_i + \mu)|}{\Delta f} \right) \leq$$

$$\prod_{i=1}^{k} exp\left(\varepsilon \frac{|z_i - (f(y)_i + \mu) - z_i + (f(x)_i + \mu)|}{\Delta f} \right) = \prod_{i=1}^{k} exp\left(\varepsilon \frac{|(f(x)_i - f(y)_i|}{\Delta f} \right) =$$

$$exp\left(\varepsilon \frac{||f(x) - f(y)||}{\Delta f} \right) \leq exp(\varepsilon).$$

The first inequality follows from the triangle inequality, and the last follows from the definition of sensitivity and the fact that $||x - y|| \leq 1$.

B.5 Τι σημαίνει πρακτικά η Διαφορική Ιδιωτικότητα

Differential privacy, as defined in Definition B.4, provides a bound to the additional knowledge any adversary can gain by looking at the output of the system.At first sight, the mathematical definition doesn't seem to translate into a real-world privacy assurance. Let's take as an example a scenario described my a matrix such as in table 1.2.

user_id	1	2	…	i	…	j	…	n
1	0	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
…								
i	0	0	0	?	0	?	0	0
…								
j	0	0	0	?	0	?	0	0
…								
n	1	0	0	0	0	1	0	0

Πίνακας B.1: the binary state matrix $State$

Suppose the adversary wants to extract information about whether user u_i is talking (sending messages) to user u_j ($T_i(j)$) or not ($!T_i(j)$). The adversary A, may know all rows of the above matrix except the i^{th} and the j^{th}. This is equivalent to A controlling all other users participating in the protocol, except these two. Furthermore, A has a prior belief $0 \leq P_{prior} \leq 1$ that the specified communication is taking place. Additionally, let $x \in State$ stand for the instances of the above matrix where user u_i is talking to u_j, and $y \in State$, where she is not.

Now lets suppose that a mechanism M implements the functionality of message exchange as defined in the ideal functionality of figure 1.1, while leaking information $f_1 = M(z), z \in State$, to A (z here stands for the reality as depicted in a binary state matrix). $M(z)$ can be viewed as a vector of numerical values ($M(z) \in \mathbb{R}^n$) and it is all the adversary learns after the execution of the mechanism. Finally, let $S \in 2^{\mathbb{R}^n}$ stand for the set of possible adversary observations which incriminate user u_i.

That is, if the adversary observes an output belonging to S, then he strongly believes that the users are communicating. We will see what happens when the adversary observes such an output.

$$
\begin{gathered}
Pr[T_i(j)|M(z) \in S] = \\
= \frac{Pr[M(z) \in S|T_i(j)] \cdot Pr[T_i(j)]}{Pr[M(z) \in S]} = \\
= \frac{Pr[M(x) \in S] \cdot Pr[T_i(j)]}{Pr[M(z) \in S]} \\
= \frac{Pr[M(x) \in S] \cdot Pr[T_i(j)]}{Pr[M(z) \in S|T_i(j)] \cdot Pr[T_i(j)] + Pr[M(z) \in S|!T_i(j)] \cdot Pr[!T_i(j)]} = \\
= \frac{Pr[M(x) \in S] \cdot Pr[T_i(j)]}{Pr[M(x) \in S] \cdot Pr[T_i(j)] + Pr[M(y) \in S] \cdot Pr[!T_i(j)]}
\end{gathered}
$$

The above quantity can be written as:

$$
\frac{\Pi_x \cdot \mu}{\Pi_x \cdot \mu + \Pi_y \cdot (1 - \mu)} \tag{B.1}
$$

where:

$$
\begin{gathered}
\Pi_x := Pr[M(x) \in S] \\
\Pi_y := Pr[M(y) \in S] \\
\mu := Pr[T_i(j)] = P_{prior}
\end{gathered}
$$

Now let's define the adversaries advantage α:

Definition B.8. The advantage α gained by the adversary after observing the output of the protocol is defined as the ratio of the probability that user u_i is talking to u_j after the adversary has made his observations, over his prior belief that this communication is taking place. That is:

$$
\alpha = \frac{Pr[T_i(j)|M(z) \in S]}{\mu}
$$

We would like to show that this advantage α is bounded by some value.

Now for a mechanism that is $(\varepsilon, 0)-$ differentially private, (instances where we have $(\varepsilon, \delta)-$ differentially private mechanisms will be discussed later) we have easily from definition B.4 that:

$$
\frac{Pr[M(x) \in S]}{Pr[M(y) \in S]} \leq e^{\varepsilon} \iff \frac{\Pi_x}{\Pi_y} \leq e^{\varepsilon} \iff \frac{\Pi_y}{\Pi_x} \geq e^{-\varepsilon} \tag{B.2}
$$

Thus, definition B.8, along with equation B.1 yields:

$$
\begin{gathered}
\alpha = \frac{Pr[T_i(j)|M(z) \in S]}{\mu} = \\
= \frac{\Pi_x}{\Pi_x \cdot \mu + \Pi_y \cdot (1 - \mu)} =
\end{gathered}
$$

$$= \frac{1}{\mu + \frac{\Pi_y}{\Pi_x} \cdot (1-\mu)} \leq$$
$$\leq \frac{1}{\mu + e^{-\varepsilon}(1-\mu)} =$$
$$= \frac{e^{\varepsilon}}{1-(1-e^{\varepsilon})\mu}$$

The inequality comes from equation B.2 and the fact that $\mu \leq 1$, given μ is a probability. Given the above, we have for the adversary's advantage:

$$\alpha \leq \frac{e^{\varepsilon}}{1-(1-e^{\varepsilon})\mu} \tag{B.3}$$

For a practical example, lets assume the adversary A has prior belief $P_{prior} = \mu = 0.5$ that user u_i is talking to user u_j. If the mechanism provides $(ln2, 0)$ - differential privacy, then equation B.3 yields $\alpha \leq \frac{4}{3}$, that is $Pr[T(i,j)|M(z) \in S] \leq \frac{2}{3}$.

This means that the adversary, who had a prior belief of 50% that the users were communicating, now is at most 67% sure they are talking.

Equation B.3 bounds the advantage achieved by the adversary for any set S, and thus for any possible algorithm the adversary can use to extract knowledge from the leaked information. For any observation made by the adversary, there is a substantial possibility that either scenario (x, y) corresponds to reality. This is how differential privacy provides plausible deniability for the users.

Having presented a concrete example, it is easier to understand what differential privacy promises in the context of anonymous communications. However, it is still somewhat difficult to quantify the belief of an adversary for a given scenario.

B.6 Σύνθεση της Διαφορικής Ιδιωτικότητας

For a mechanism that offers a given degree of (differential) privacy in any given round, it is very important to study how privacy degrades as time progresses. For example, a conversation that spans over several rounds becomes more and more easy for the adversary to discern. Thus, we present the known results on composition of differentially private mechanisms, after we have defined what composition means.

B.6.1 Defining Composition

As explained in [3], we would like our definition to cover active adversaries that can alter the databases (action matrices) being input to future mechanisms. Thus, we model composition through the experiment presented in [3] and described below. Let F be a family of database access mechanisms. For a probabilistic adversary A, we consider two experiments, Experiment 0 and Experiment 1, defined as follows.

Experiment b for family F and adversary A:

For $i = 1, \ldots, k$:

1. A outputs two adjacent databases x_i^0 and x_i^1, a mechanism $\mathcal{M}_i \in F$, and parameters w_i.

2. A receives $y_i \in_{\mathbb{R}} \mathcal{M}_i(w_i, x_i^b)$.

We allow the adversary A above to be stateful throughout the experiment, and thus he may choose the databases, mechanisms, and the parameters adaptively depending on the outputs of previous mechanisms. We define A's *view* of the experiment to be A's coin tosses and all of the mechanism outputs $(y_1, \ldots, y_k)$. (The x_i^j's , $\mathcal{M}_i$'s and w_i's can all be reconstructed from these.) To understand the experiment better, consider an adversary who always chooses x_i^0 to contain the row that Alice is sending messages to Bob, and x_i^1 to differ only at the specified row (Alice is doing something else). Then, Experiment 0 can be thought of as the reality and Experiment 1 as Alice's cover story. The intuition is that Alice wants to have a guarantee that the adversary can't tell apart, given the output of all k mechanisms (after k rounds), whether or not Alice ever sent a message to Bob. The formal definition follows.

B.6.2 Differential privacy under k-fold adaptive composition

Definition B.9. We say that the family F of database access mechanisms satisfies ε-differential privacy under k-fold adaptive composition if for every adversary A, we have $D_\infty(V^0||V^1) \leq \varepsilon$, where V^b denotes the view of A in k-fold Composition Experiment b, above. (ε, δ)-differential privacy under k-fold adaptive composition instead requires that $D_\infty^\delta(V^0||V^1) \leq \varepsilon$. The statistical distances D_∞ and D_∞^δ are as defined in section A.1.

Now, we would like to know how the differential privacy provided by a mechanism in one round, degrades over k rounds. To this end, we have the following theorem, proven in [3]:

Θεώρημα 3. (Advanced Composition). For all ε, δ, $\delta' \geq 0$, the class of (ε, δ)-differentially private mechanisms satisfies $(\varepsilon', k\delta + \delta'))$-differential privacy under k-fold adaptive composition for:

$$\varepsilon' = \sqrt{2kln(1/\delta')\varepsilon} + k\varepsilon(e^\varepsilon - 1).$$

This gives as the following corollary:

Corollary B.1. Given target privacy parameters $0 < \varepsilon' < 1$ and $\delta' > 0$, to ensure $(\varepsilon', k\delta + \delta')$ cumulative privacy loss over k rounds, it suffices that for each round the mechanism is (ε, δ)- differentially private, where

$$\varepsilon = \frac{\varepsilon'}{2\sqrt{2kln(1/\delta')}}$$

The above corollary gives a rough guide for how to set ε to get the desired privacy parameters under composition. Obviously, theorem 3 gives tighter bounds and should be used with actual implementations. From all the above, we see that the factor ε degrades with $\sqrt{k}$, and δ with k, where k is the number of rounds. We will return to this topic, with specific examples in the next chapter.

Βιβλιογραφία

[1] Pfitzmann, A. and Hansen, M., 2005. Anonymity, unlinkability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology.

[2] Van Den Hooff, J., Lazar, D., Zaharia, M. and Zeldovich, N., 2015, October. Vuvuzela: Scalable private messaging resistant to traffic analysis. In Proceedings of the 25th Symposium on Operating Systems Principles (pp. 137-152). ACM.

[3] Dwork, C. and Roth, A., 2014. The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science, 9(3-4), pp.211-407.

[4] Chaum, D.L., 1981. Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM, 24(2), pp.84-90.

[5] Sampigethaya, K. and Poovendran, R., 2006. A survey on mix networks and their secure applications. Proceedings of the IEEE, 94(12), pp.2142-2181.

[6] Jakobsson, M., Juels, A. and Rivest, R.L., 2002, August. Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking. In USENIX security symposium (pp. 339-353).

[7] Chaum, D., 1988. The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of cryptology, 1(1), pp.65-75.

[8] Pfitzmann, A. and Waidner, M., 1985, April. Networks without user observability—design options. In Advances in Cryptology—EUROCRYPT'85 (pp. 245-253). Springer Berlin Heidelberg.

[9] Serjantov, A. and Danezis, G., 2002, April. Towards an information theoretic metric for anonymity. In Privacy Enhancing Technologies (pp. 41-53). Springer Berlin Heidelberg.

[10] Diaz, C., Seys, S., Claessens, J. and Preneel, B., 2002, April. Towards measuring anonymity. In Privacy Enhancing Technologies (pp. 54-68). Springer Berlin Heidelberg.

[11] Dingledine, R., Mathewson, N. and Syverson, P., 2004. Tor: The second-generation onion router. Naval Research Lab Washington DC.

[12] Park, C., Itoh, K. and Kurosawa, K., 1993, May. Efficient anonymous channel and all/nothing election scheme. In Advances in Cryptology—EUROCRYPT'93 (pp. 248-259). Springer Berlin Heidelberg.

[13] Diffie, W. and Hellman, M.E., 1976. New directions in cryptography. Information Theory, IEEE Transactions on, 22(6), pp.644-654.

[14] Jakobsson, M. and Juels, A., 2001, August. An optimally robust hybrid mix network. In Proceedings of the twentieth annual ACM symposium on Principles of distributed computing (pp. 284-292). ACM.

[15] Moeller, B. Provably secure public-key encryption for length-preserving Chaumian mixes, in Proc. CT-RSA. New York: Springer-Verlag, 2003, pp. 244

[16] Golle, P., Jakobsson, M., Juels, A. and Syverson, P., 2004. Universal re-encryption for mixnets. In Topics in Cryptology–CT-RSA 2004 (pp. 163-178). Springer Berlin Heidelberg.

[17] Nelson, R.J., Nelson Raymond J, 1986. One level sorting network. U.S. Patent 4,628,483.

[18] Batcher, K.E., 1968, April. Sorting networks and their applications. In Proceedings of the April 30–May 2, 1968, spring joint computer conference (pp. 307-314). ACM.

[19] Shell, D.L., 1959. A high-speed sorting procedure. Communications of the ACM, 2(7), pp.30-32.

[20] Ajtai, M., Komlós, J. and Szemerédi, E., 1983, December. An 0 (n log n) sorting network. In Proceedings of the fifteenth annual ACM symposium on Theory of computing (pp. 1-9). ACM.

[21] Yao, A.C., 1982, November. Protocols for secure computations. In Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on (pp. 160-164). IEEE.

[22] Ben-David, A., Nisan, N. and Pinkas, B., 2008, October. FairplayMP: a system for secure multi-party computation. In Proceedings of the 15th ACM conference on Computer and communications security (pp. 257-266). ACM.

[23] Goldreich, O., Micali, S. and Wigderson, A., 1987, January. How to play any mental game. In Proceedings of the nineteenth annual ACM symposium on Theory of computing (pp. 218-229). ACM.

[24] Shamir, A., 1979. How to share a secret. Communications of the ACM, 22(11), pp.612-613.

[25] Bogdanov, D., Laur, S. and Willemson, J., 2008. Sharemind: A framework for fast privacy-preserving computations. In Computer Security-ESORICS 2008 (pp. 192-206). Springer Berlin Heidelberg.

[26] Burkhart, M., Strasser, M., Many, D. and Dimitropoulos, X., 2010. SEPIA: Privacy-preserving aggregation of multi-domain network events and statistics. Network, 1, p.101101.

[27] Damgård, I., Geisler, M., Krøigaard, M. and Nielsen, J.B., 2009. Asynchronous multiparty computation: Theory and implementation. In Public Key Cryptography–PKC 2009 (pp. 160-179). Springer Berlin Heidelberg.

[28] Dingledine, R., Mathewson, N. and Syverson, P., 2004. Tor: The second-generation onion router. Naval Research Lab Washington DC.

[29] https://www.torproject.org/

[30] http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/networks/oemen.htm

[31] Chaum, D., 1988. The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of cryptology, 1(1), pp.65-75.

[32] Wolinsky, D.I., Corrigan-Gibbs, H., Ford, B. and Johnson, A., 2012. Dissent in numbers: Making strong anonymity scale. In Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12) (pp. 179-182).

[33] Robson, M., Polte, M., Goel, S. and Sirer, E., 2003. Herbivore: A scalable and efficient protocol for anonymous communication. Cornell University.

[34] https://www.cs.bris.ac.uk/Research/CryptographySecurity/MPC/

[35] Rabin, M.O., 2005. How To Exchange Secrets with Oblivious Transfer. IACR Cryptology ePrint Archive, 2005, p.187.

[36] Knuth, D.E., 1998. The art of computer programming: sorting and searching (Vol. 3). Pearson Education.

[37] Garay, J., Schoenmakers, B. and Villegas, J., 2007. Practical and secure solutions for integer comparison. In Public Key Cryptography–PKC 2007 (pp. 330-342). Springer Berlin Heidelberg.

[38] https://github.com/aicis/fresco

[39] Jónsson, K.V., Kreitz, G. and Uddin, M., 2011. Secure Multi-Party Sorting and Applications. IACR Cryptology ePrint Archive, 2011, p.122.

[40] Murdoch, S.J. and Danezis, G., 2005, May. Low-cost traffic analysis of Tor. In Security and Privacy, 2005 IEEE Symposium on (pp. 183-195). IEEE.

[41] Chor, B., Kushilevitz, E., Goldreich, O. and Sudan, M., 1998. Private information retrieval. Journal of the ACM (JACM), 45(6), pp.965-981.