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

Broadcast with Unknown Participants and Any Number of Corruptions

Διπλωματική Εργασία

Νικόλαος Άγγελος Σκουμιός

Επιβλέπων: Αριστείδης Παγουρζής
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων : Βασίλειος Ζήκας
Αν. Καθηγητής Georgia Tech

Αθήνα, Οκτώβριος 2024

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Λογικής και Επιστήμης Υπολογιστών

Broadcast with Unknown Participants and Any Number of Corruptions

Διπλωματική Εργασία

Νικόλαος Άγγελος Σκουμιός

Επιβλέπων: Αριστείδης Παγουρζής
Καθηγητής Ε.Μ.Π.

Συνεπιβλέπων : Βασίλειος Ζήκας
Αν. Καθηγητής Georgia Tech

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Οκτωβρίου, 2024.

.....	.....	.....
Αριστείδης Παγουρζής	Βασίλειος Ζήκας	Νικόλαος Λεονάρδος
Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Georgia Tech	Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2024

.....

Νικόλαος Άγγελος Σκουμιός
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Το πρόβλημα της βυζαντινής συμφωνίας αποτελεί ένα από τα πιο θεμελιώδη πεδία έρευνας στον τομέα των κατανεμημένων αλγορίθμων που είναι ανεκτικοί στην παρουσία σφαλμάτων. Παρότι το ενδιαφέρον της επιστημονικής κοινότητας έχει ενταθεί ιδιαίτερα τα τελευταία χρόνια για την επίλυση του προβλήματος λόγω της ανάπτυξης της τεχνολογίας του blockchain και των κρυπτονομισμάτων, η συντριπτική πλειοψηφία των προτεινόμενων λύσεων υποθέτει ένα δεδομένο σύνολο παικτών στο οποίο τα σφάλματα μειοψηφούν.

Στην παρούσα εργασία, παρουσιάζουμε την πρώτη, εξ όσων γνωρίζουμε, λύση του βυζαντινού Broadcast (όπου οι παίκτες καλούνται να συμφωνήσουν στην τιμή εισόδου ενός καθορισμένου αποστολέα) η οποία είναι ανεκτική σε οποιονδήποτε αριθμό σφαλμάτων στο μοντέλο ανώνυμων παικτών, στο οποίο υποθέτουμε ότι οι παίκτες δεν διαθέτουν καμία πληροφορία για το ακριβές σύνολο των παικτών ούτε για το πλήθος τους. Η λύση μας βασίζεται στην κεντρική ιδέα του αντίστοιχου πρωτοκόλλου στο μοντέλο του γνωστού συνόλου παικτών, όπου σε κάθε γύρο οι παίκτες αποδέχονται μία τιμή ως είσοδο του αποστολέα όταν λάβουν τόσες υπογραφές προς υποστήριξή της όσες και ο αριθμός του γύρου.

Συγκεκριμένα, τροποποιούμε κατάλληλα το πρωτόκολλο αυτό, ώστε σε κάθε γύρο οι παίκτες να αποδέχονται την ενεργή συμμετοχή οποιουδήποτε πιθανού παίκτη όταν λάβουν τόσες υπογραφές προς υποστήριξή του όσες και ο αριθμός του γύρου, αλλά από παίκτες που έχουν ήδη αποδεχθεί στους προηγούμενους γύρους. Έτσι, κατασκευάζουμε ένα νέο πρωτόκολλο στο μοντέλο ανώνυμων παικτών που επιτυγχάνει την συμφωνία των παικτών σε ένα υποσύνολό τους, το οποίο περιέχει όλους τους τίμιους παίκτες (δηλαδή εκείνους που δεν αποκλίνουν από το πρωτόκολλο). Επιπλέον, επεκτείνουμε το πρωτόκολλό αυτό ώστε οι παίκτες να συμφωνούν σε μία τιμή εισόδου για καθέναν από τους παίκτες αυτού του συμφωνημένου συνόλου. Έτσι, η λύση μας για το βυζαντινό Broadcast προκύπτει ως άμεση εφαρμογή του επεκτεταμένου πρωτοκόλλου.

Τέλος, βασιζόμενοι και πάλι σε αντίστοιχα αποτελέσματα στο μοντέλο γνωστών παικτών, αποδεικνύουμε ότι οποιοδήποτε πρωτόκολλο που λύνει το βυζαντινό Broadcast στο μοντέλο ανώνυμων παικτών με την ίδια ανεκτικότητα σε σφάλματα χρειάζεται στην χειρότερη περίπτωση τόσους γύρους επικοινωνίας μεταξύ των παικτών όσο και το πλήθος τους. Έτσι, δείχνουμε ότι η λύση που παρουσιάζουμε, η οποία απαιτεί τόσους γύρους όσο και το πλήθος των παικτών, είναι βέλτιστη ως προς την πολυπλοκότητα γύρων χειρότερης περίπτωσης.

Λέξεις κλειδιά: κατανεμημένοι αλγόριθμοι, βυζαντινή συμφωνία, ανώνυμοι παίκτες, οποιοσδήποτε αριθμός σφαλμάτων

Abstract

The byzantine agreement problem is one of the most fundamental research areas in the field of fault-tolerant distributed algorithms. While the problem has attracted the interest of the scientific community the last years due to the development of blockchain technologies and cryptocurrencies, the bulk of the literature studies the cases of static participation and honest majority.

In this work, we present the first, to our knowledge, byzantine Broadcast protocol (where the parties shall agree on the input value of designated sender), which tolerates any number of corruptions in the unknown participants model, where the parties have no clue about the set of actual participants nor for their number. Our solution is based on the main idea of the corresponding protocol in the known participants model, where in every round the parties accept an input value of the sender only if they receive as many signatures in support of it as the round number.

In particular, we modify appropriately this protocol, so then in every round the parties accept the active participation of some other party only it they receive as many signatures in support of it from as many parties as the round number, but only from parties that they have already accepted in the previous rounds. Hence, we we construct a new protocol in the unknown participants model which achieves agreement among the parties on a subset of them, which contains all honest parties (the ones which do not deviate from the protocol). In addition, we extend our protocol, so then the parties agree on an input value for each one of the parties of this commonly agreed party set. In this way, our solution of byzantine Broadcast is an immediate application of this extended protocol.

Finally, relying once again on corresponding results in the known participants model, we prove that any protocol that solves byzantine Broadcast in the unknown participants model with the same resilience requires as many communication rounds in the worst case as the number of active parties. Thus, we show that our protocol, which requires the same number of rounds as the number of parties, is optimal in terms of its worst-case round complexity.

Keywords: distributed algorithms, byzantine agreement, unknown participants, any number of corruptions

Ευχαριστίες

Ολοκληρώνοντας την διπλωματική μου εργασία, θα ήθελα να ευχαριστήσω πρώτα από όλα την οικογένεια μου που ήταν δίπλα μου κάθε στιγμή σε αυτήν μου την προσπάθεια. Θα ήθελα ακόμα να ευχαριστήσω ιδιαίτερα τον κύριο Παγουρτζή για την καθοδήγηση του σε όλο το χρονικό διάστημα που ερευνούσαμε το συγκεκριμένο πρόβλημα.

Επιπλέον, θα ήθελα να ευχαριστήσω τον Dr. Wonseok Choi με τον οποίο αφιερώσαμε αμέτρητες ώρες στην προσπάθεια να αντιληφθούμε αρχικά αν το πρόβλημα που ερευνούσανε ήταν επιλύσιμο ή όχι και στην συνέχεια, για να φέρουμε την λύση μας στην τελική της μορφή. Έβρισκε πάντα την όρεξη και την ενέργεια για να ανταλλάξουμε τις ιδέες και τις ανησυχίες μας για το συγκεκριμένο πρόβλημα και η συμβολή του ήταν καθοριστική στο να φτάσουν τα αποτελέσματα μας στο επιθυμητό επίπεδο.

Τέλος, θα ήθελα να ευχαριστήσω με όλη μου την καρδιά τον κύριο Ζήκα, πρωτίστως για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα ερευνητικό project υψηλών απαιτήσεων από προπτυχιακό κιόλας επίπεδο, χωρίς κανένα εχέγγυο παρά μόνο με την όρεξή μου για σκληρή δουλειά και την (όχι και τόσο εμφανή πολλές φορές) φιλοδοξία για ένα ερευνητικό αποτέλεσμα υψηλού επιπέδου. Προσωπικά, θεωρώ ότι το καλύτερο μέτρο της προσφοράς μας προς τους άλλους είναι ο χρόνος που τους αφιερώνουμε και με αυτό το σκεπτικό, θεωρώ ανεκτίμητο τον χρόνο που αφιέρωσε ο κύριος Ζήκας στις τακτικές συζητήσεις μας για την εξερεύνηση ενός προβλήματος που εκ πρώτης όψεως δεν άφηνε πολλά περιθώρια αισιοδοξίας για την επίλυσή του. Ειλικρινά, ευελπιστώ αν κάποια στιγμή φτάσω σε μια αντίστοιχη θέση να αντιμετωπίζω με τον ίδιο τρόπο τους άπειρους φοιτητές με όνειρα, φιλοδοξίες και όρεξη για δουλειά. Όπως και αν συνεχιστεί το ταξίδι μου στην έρευνα, θα θυμάμαι για πάντα τη βοήθειά του σε όλα τα επίπεδα και θα αισθάνομαι ότι ένα μεγάλο μέρος της ακαδημαϊκής μου εξέλιξης οφείλεται σε εκείνον.

Νίκος Σκουμιός
Οκτώβριος 2024

Contents

Κεφάλαιο 1

Εκτεταμένη Ελληνική Περίληψη

1.1 Εισαγωγή

Το πρόβλημα της *βυζαντινής συμφωνίας* (BA) [1, 2] είναι ένα από τα σημαντικότερα πεδία έρευνας στον τομέα των κατανεμημένων αλγορίθμων και το ενδιαφέρον της επιστημονικής κοινότητας για τη λύση του προβλήματος στην περίπτωση που οι παίκτες μπορούν να συμμετέχουν *δυναμικά* έχει ενταθεί ιδιαίτερα τα τελευταία χρόνια χάρη στην ανάπτυξη της τεχνολογίας του blockchain.

Παρ΄όλα αυτά, όλες οι λύσεις που έχουν παρουσιαστεί μέχρι σήμερα υποθέτουν τίμια πλειοψηφία των παικτών είτε ότι οι παίκτες γνωρίζουν το ακριβές σύνολο των παικτών καθ' όλη την εκτέλεση του πρωτοκόλλου. Στην παρούσα εργασία, παρουσιάζουμε ένα πρωτόκολλο ανεκτικό ενάντια σε οποιονδήποτε αριθμό σφαλμάτων στο μοντέλο των ανώνυμων παικτών.

1.2 Θεωρητικό Υπόβαθρο

1.2.1 Κρυπτογραφικό Υπόβαθρο

Αρχικά, εισάγουμε την έννοια των *αμελητέων* συναρτήσεων.

Ορισμός 1 (Αμελητέα Συνάρτηση)**.** *Μία συνάρτηση $f : \mathbb{N} \to \mathbb{R}_+$ θα αποκαλείται αμελητέα, αν για κάθε $c > 0$, υπάρχει $n_0 \in \mathbb{N}$, τέτοιο ώστε $f(n) < 1/n^c$, για κάθε $n > n_0$ (συμβολίζουμε με $f = \mathsf{negl}(\cdot)$).*

Στην συνέχεια, θα δώσουμε τον ορισμό του *σχήματος (ψηφιακών) υπογραφών* και θα περιγράψουμε τις επιθυμητές ιδιότητες του.

Ορισμός 2 (Σχήμα Ψηφιακών Υπογραφών)**.** *Ένα σχήμα ψηφιακών υπογραφών είναι μία τριάδα αλγορίθμων* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. *Ο αλγόριθμος* KeyGen *παίρνει ως είσοδο τη συμβολοσειρά 1^κ και δίνει στην έξοδο το ζεύγος* $(\mathsf{sk}, \mathsf{pk})$*, το οποίο αποτελείται από το μυστικό κλειδί* sk *και το δημόσιο κλειδί* pk*. Ο αλγόριθμος* Sign *παίρνει ως είσοδο το μυστικό κλειδί* sk *και ένα μήνυμα $m \in \{0,1\}^{\mathsf{poly}(\kappa)}$ και δίνει στην έξοδο την υπογραφή σ, δηλαδή $\sigma = \mathsf{Sign}_{\mathsf{sk}}(m)$. Ο αλγόριθμος* Verify *παίρνει ως είσοδο ένα δημόσιο κλειδί* pk*, ένα μήνυμα $m \in \{0,1\}^*$ και μία υπογραφή σ και δίνει στην έξοδο ένα bit $b \in \{0,1\}$, δηλαδή $b = \mathsf{Verify}_{\mathsf{pk}}(m, \sigma)$. Για την τριάδα* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *πρέπει να ισχύει ότι $\mathsf{Verify}_{\mathsf{pk}}(m, \mathsf{Sign}_{\mathsf{sk}}(m)) = 1$, για κάθε μήνυμα $m \in \{0,1\}^{\mathsf{poly}(\kappa)}$ και κάθε έξοδο* $(\mathsf{sk}, \mathsf{pk})$ *του* $\mathsf{KeyGen}(1^\kappa)$.

Στα επόμενα κεφάλαια, θα υποθέτουμε συχνά ότι ένα σύνολο παικτών $\mathcal{P} = \{P_1, \ldots, P_n\}$ μοιράζεται μία *υποδομή δημοσίου κλειδιού (PKI)*, όπως περιγράφεται παρακάτω.

Ορισμός 3 (Υποδομή Δημοσίου Κλειδιού (PKI))**.** *Θα λέμε ότι ένα σύνολο παικτών $\mathcal{P} = \{P_1, \ldots, P_n\}$ μοιράζεται μία υποδομή δημοσίου κλειδιού (PKI) αν για κάθε ζεύγος παικτών $P_i, P_j \in \mathcal{P}$, ο P_i έχει ένα μυστικό κλειδί sk_i που έχει παραχθεί από τον KeyGen και ο P_j έχει το αντίστοιχο δημόσιο κλειδί pk_i, τέτοιο ώστε $\mathsf{Verify}_{\mathsf{pk}_i}(m, \mathsf{Sign}_{\mathsf{sk}_i}(m)) = 1$, για κάθε μήνυμα $m \in \{0,1\}^{\mathsf{poly}(\kappa)}$.*

Για κάθε PKI, θέλουμε οποιοσδήποτε PPT αντίπαλος να μην μπορεί να *πλαστογραφεί* υπογραφές, δηλαδή να μην μπορεί να παράγει έγκυρες υπογραφές χωρίς να διαθέτει το αντίστοιχο μυστικό κλειδί.

Ορισμός 4 (Αδυναμία Πλαστογράφησης)**.** *Ένα PKI που μοιράζεται μεταξύ των παικτών ενός συνόλου $\mathcal{P} = \{P_1 \ldots, P_n\}$ ικανοποιεί την ιδιότητα αδυναμίας πλαστογράφησης ενάντια σε οποιονδήποτε PPT αντίπαλο αν η πιθανότητα ο αντίπαλος να βρει μία συμβολοσειρά σ, τέτοια ώστε $\mathsf{Verify}_{\mathsf{pk}_i}(b, \sigma) = 1$, για οποιονδήποτε παίκτη $P_i \in \mathcal{P}$ και οποιαδήποτε τιμή $b \in \{0,1\}$, χωρίς να γνωρίζει το sk_i, είναι αμελητέα ως προς την παράμετρο ασφαλείας κ.*

Θα θεωρούμε ότι αν $\mathsf{Verify}_{\mathsf{pk}_i}(b, \sigma) = 1$, για κάποια τιμή $b \in \{0,1\}$, τότε το σ έχει πράγματι παραχθεί από τον P_i με *συντριπτική* πιθανότητα, δηλαδή $1 - \mathsf{negl}(\kappa)$. Για ευκολία, θα συμβολίζουμε με $\mathsf{Sign}_p(b)$ την τριάδα $(i, b, \mathsf{Sign}_{\mathsf{sk}_i}(b))$.

1.2.2 Μοντέλο

Έστω $\mathcal{P} = \{P_1, \ldots, P_n\}$ ένα σύνολο n *παικτών*, δηλαδή PPT Μηχανών Turing. Ένα *κατανεμημένο πρωτόκολλο* π είναι μία n-πλειάδα PPT αλγορίθμων $(\pi_1, \ldots, \pi_n)$ που τρέχουν οι παίκτες του $\mathcal{P}$ για να επικοινωνούν μεταξύ τους, όπου κάθε παίκτης P_i εκτελεί τον αλγόριθμο π_i. Οι παίκτες επικοινωνούν μεταξύ τους μέσω διαύλων *σημείου-προς-σημείο*. Θεωρούμε ένα *σύγχρονο* δίκτυο, στο οποίο η επικοινωνία των παικτών γίνεται σε γύρους.

Αντίπαλος. Ο *αντίπαλος* είναι μία Μηχανή Turing που μπορεί να πάρει υπό τον έλεγχό του κάποιους παίκτες του $\mathcal{P}$ και να τους επιβάλλει να αποκλίνουν από την συμπεριφορά που υποδεικνύει το πρωτόκολλο. Οι παίκτες που βρίσκονται υπό τον έλεγχο του αντιπάλου θα αποκαλούνται *κακόβουλοι* παίκτες (ή αλλιώς *σφάλματα*), ενώ οι υπόλοιποι παίκτες θα αποκαλούνται *τίμιοι*. Θα ασχοληθούμε κυρίως με δύο *είδη σφαλμάτων*, δηλαδή απόκλισης από το πρωτόκολλο : τα *crash σφάλματα*, τα οποία απλώς σταματούν να στέλνουν μηνύματα από κάποιο σημείο της εκτέλεσης και έπειτα, και τα *βυζαντινά σφάλματα*, τα οποία μπορούν να αποκλίνον τελείως από τις οδηγίες του πρωτοκόλλου. Ένας t-*αντίπαλος* θεωρούμε ότι μπορεί να προκαλέσει μέχρι t βυζαντινά σφάλματα και ανάλογα με τις απαιτήσεις του εκάστοτε προβλήματος υποθέτουμε ότι είναι είτε PPT είτε *απεριόριστος*, δηλαδή ότι έχει απεριόριστη υπολογιστική ισχύ.

Μετρικές Πολυπλοκότητας. Θα ασχοληθούμε με τρεις μετρικές πολυπλοκότητας που χαρακτηρίζουν τα κατανεμημένα πρωτόκολλα που είναι ανεκτικά σε σφάλματα : την *πολυπλοκότητα γύρων* που είναι ο μέγιστος αριθμός γύρων για τον οποίο πρέπει να τρέχει κάποιος τίμιος παίκτης στην χειρότερη περίπτωση, την *πολυπλοκότητα επικοινωνίας* που ορίζεται ως ο συνολικός αριθμός μηνυμάτων / bit που στέλνουν οι τίμιοι παίκτες καθ' όλη τη διάρκεια της εκτέλεσης του πρωτοκόλλου στην χειρότερη περίπτωση και η *υπολογιστική πολυπλοκότητα* που ορίζεται ως ο μέγιστος χρόνος υπολογισμού που πρέπει να έχει στη διάθεση του κάποιος τίμιος παίκτης στην χειρότερη περίπτωση.

1.2.3 Βυζαντινή Συμφωνία

Η *Βυζαντινή Συμφωνία* ([1, 2]) είναι το πρόβλημα στο οποίο οι παίκτες πρέπει να συμφωνήσουν στην τιμή που προτείνει ένας καθορισμένος παίκτης (*αποστολέας*) ακόμα και αν προκύψουν βυζαντινά σφάλματα. Από εδώ και στο εξής, θα αναφερόμαστε στο πρόβλημα αυτό ως *βυζαντινό Broadcast*, το οποίο τυπικά ορίζεται ως εξής :

Ορισμός 5 (Ντετερμινιστικό Βυζαντινό Broadcast)**.** *Έστω $\mathcal{P}$ ένα σύνολο n παικτών. Ένα πρωτόκολλο π που εκτελείται από τους παίκτες του $\mathcal{P}$, όπου ένας καθορισμένος αποστολέας $s \in \mathcal{P}$ παίρνει ως είσοδο μία αρχική τιμή $b_s \in \{0,1\}$, είναι ντετερμινιστικό βυζαντινό Broadcast πρωτόκολλο ανεκτικό ενάντια σε οποιονδήποτε (απεριόριστο) t-αντίπαλο, αν ικανοποιούνται οι παρακάτω συνθήκες :*

- ***Εγκυρότητα:*** *Αν ο αποστολέας s είναι τίμιος, τότε όλοι οι τίμιοι παίκτες δίνουν στην έξοδο την τιμή b_s.*
- ***Συμφωνία:*** *Όλοι οι τίμιοι παίκτες δίνουν στην έξοδο την ίδια τιμή $b \in \{0,1\}$.*
- ***Τερματισμός:*** *Υπάρχει ένας προκαθορισμένος αριθμός γύρων R τέτοιος ώστε το πρωτόκολλο εγγυημένα ολοκληρώνεται (δηλ. όλοι οι τίμιοι παίκτες δίνουν στην έξοδο κάποια τιμή) το πολύ σε R γύρους.*

Θα αναφερόμαστε στο *βυζαντινό Broadcast* ως το πρόβλημα που απαιτεί να ικανοποιούνται οι ιδιότητες του παραπάνω ορισμού με συντριπτική πιθανότητα και στο αντίστοιχα, στο *πιθανοτικό βυζαντινό Broadcast* όταν η πιθανότητα επιτυχίας πρέπει να είναι αυστηρά μεγαλύτερη από $1/2$.

Ορισμός 6 (Βυζαντινό Broadcast)**.** *Έστω $\mathcal{P}$ ένα σύνολο n παικτών. Ένα πρωτόκολλο π που εκτελείται από τους παίκτες του $\mathcal{P}$, όπου ένας καθορισμένος αποστολέας $s \in \mathcal{P}$ παίρνει ως είσοδο μία αρχική τιμή $b_s \in \{0,1\}$, είναι βυζαντινό Broadcast πρωτόκολλο ανεκτικό ενάντια σε οποιονδήποτε PPT t-αντίπαλο, αν οι ιδιότητες του Ορισμού 5 ικανοποιούνται με συντριπτική πιθανότητα, υποθέτοντας την ύπαρξη ενός PKI (Ορισμός 3).*

Ορισμός 7 (Πιθανοτικό Βυζαντινό Broadcast)**.** *Έστω $\mathcal{P}$ ένα σύνολο n παικτών και δ η πιθανότητα λάθους αυστηρά μικρότερη από $1/2$. Ένα πρωτόκολλο π που εκτελείται από τους παίκτες του $\mathcal{P}$, όπου ένας καθορισμένος αποστολέας $s \in \mathcal{P}$ παίρνει ως είσοδο μία αρχική τιμή $b_s \in \{0,1\}$, είναι πιθανοτικό βυζαντινό Broadcast πρωτόκολλο ανεκτικό ενάντια σε οποιονδήποτε PPT t-αντίπαλο, αν οι ιδιότητες του Ορισμού 5 ικανοποιούνται με πιθανότητα $1-\delta$, υποθέτοντας την ύπαρξη ενός PKI (Ορισμός 3).*

Ανεκτικότητα. Όταν ένα πρωτόκολλο επιτυγχάνει ακόμα και με την παρουσία $t < (1-\epsilon)n$ σφαλμάτων, για μία αυθαίρετα μικρή σταθερά $\epsilon > 0$, τότε θα λέμε ότι το πρωτόκολλο είναι ανεκτικό ενάντια σε *κακόβουλη πλειοψηφία*. Στην ειδική περίπτωση που το πρωτόκολλο επιτυγχάνει ακόμα και με την παρουσία t σφαλμάτων για οποιοδήποτε $t < n$ (e.g. ακόμα και αν $t = n - O(1)$), τότε θα λέμε ότι το πρωτόκολλο είναι ανεκτικό ενάντια σε *οποιονδήποτε αριθμό σφαλμάτων*.

1.3 Βυζαντινή Συμφωνία με Κακόβουλη Πλειοψηφία

1.3.1 Κάτω Όρια Πολυπλοκότητας

Πολυπλοκότητα Γύρων. Οι Fischer και Lynch [3] απέδειξαν ότι κάθε ντετερμινιστικό βυζαντινό Broadcast πρωτόκολλο που είναι ανεκτικό ενάντια σε t βυζαντινά σφάλματα χρειάζεται τουλάχιστον $t+1$ γύρους, ανεξάρτητα από την τιμή του n. Το κάτω όριο των $t+1$ γύρων αποδείχθηκε ότι ισχύει και στην περίπτωση που το πρωτόκολλο υποθέτει την ύπαρξη PKI [4, 5] και στην περίπτωση που ο αντίπαλος μπορεί να προκαλέσει μόνο crash σφάλματα [6, 7].

Θεώρημα 1. *Δεν υπάρχει ντετερμινιστικό βυζαντινό Broadcast πρωτόκολλο (Ορισμός 5) ανεκτικό ενάντια σε t crash σφάλματα που τερματίζει σε λιγότερους από $t+1$ γύρους, όπου $t \leq n-2$.*

Αργότερα, στο [8] παρουσιάστηκε το πρώτο κάτω όριο στην πολυπλοκότητα γύρων των βυζαντινών Broadcast πρωτοκόλλων που ισχύει ακόμα και για πιθανοτικά πρωτόκολλα με πιθανότητα επιτυχίας αυστηρά μεγαλύτερη από $1/2$ (Ορισμός 7). Συγκεκριμένα, χρησιμοποιώντας ένα πληροφοριο-θεωρητικό επιχείρημα απέδειξαν ότι δεν υπάρχει πιθανοτικό βυζαντινό Broadcast πρωτόκολλο, ανεκτικό ενάντια σε $t \leq n-2$ σφάλματα, που να τερματίζει πάντα σε λιγότερους από $2n/(n-t)-2$ γύρους.

Θεώρημα 2. *Δεν υπάρχει πιθανοτικό βυζαντινό Broadcast πρωτόκολλο (Ορισμός 7) που τερματίζει σε λιγότερους από $2n/(n-t)-2$ γύρους, όπου $t \leq n-2$.*

Πόρισμα 8. *Δεν υπάρχει πιθανοτικό βυζαντινό Broadcast πρωτόκολλο (Ορισμός 7) που τερματίζει σε λιγότερους από $n-1$ γύρους, όπου $t = n-2$.*

Πόρισμα 9. *Δεν υπάρχει πιθανοτικό βυζαντινό Broadcast πρωτόκολλο (Ορισμός 7) που τερματίζει σε λιγότερους από $\Omega(\frac{1}{\epsilon})$ γύρους, όπου $t = (1-\epsilon)n$.*

Πολυπλοκότητα Επικοινωνίας. Οι Dolev και Reischuk [9] παρουσίασαν ένα κάτω όριο στον αριθμό των μηνυμάτων που πρέπει να στείλουν οι τίμιοι παίκτες σε κάθε ντετερμινιστικό βυζαντινό Broadcast πρωτόκολλο που ικανοποιεί τις ιδιότητες του Ορισμού 5. Συγκεκριμένα, απέδειξαν ότι κάθε ντετερμινιστικό βυζαντινό Broadcast πρωτόκολλο πρέπει να έχει πολυπλοκότητα μηνυμάτων $\Omega(n+t^2)$, δείχνοντας ότι πάντα υπάρχει κάποια εκτέλεση του πρωτοκόλλου με $t/2$ σφάλματα στη οποία οι τίμιοι παίκτες πρέπει να στείλουν συλλογικά τουλάχιστον $t/2$ μηνύματα σε καθένα από τα σφάλματα. Έτσι, κατέληξαν στο παρακάτω θεώρημα :

Θεώρημα 3. *Έστω π ένα ντετερμινιστικό βυζαντινό Broadcast πρωτόκολλο (σύμφωνα με τον Ορισμό 5) ανεκτικό ενάντια σε $t \leq n-2$ σφάλματα, τότε υπάρχει εκτέλεση του π στην οποία οι τίμιοι παίκτες πρέπει συνολικά να στείλουν τουλάχιστον $\max\{(n-1)/2, t^2/4\}$ μηνύματα.*

Σχετικά πρόσφατα, το κάτω φράγμα $\Omega(t^2)$ στην πολυπλοκότητα μηνυμάτων των βυζαντινών Broadcast πρωτοκόλλων αποδείχθηκε ότι ισχύει και στην περίπτωση των πιθανοτικών πρωτοκόλλων με πιθανότητα επιτυχίας αυστηρά μεγαλύτερη από $3/4$. Όπως δείχθηκε στο [10], σε κάθε πιθανοτικό βυζαντινό Broadcast πρωτόκολλο (Ορισμός 7) με πιθανότητα λάθους $\delta \leq 1/4-\epsilon$, για κάποιο $\epsilon > 0$, η εκτιμώμενη τιμή των μηνυμάτων πρέπει να στείλουν συνολικά οι τίμιοι παίκτες είναι $(\epsilon t)^2$.

Θεώρημα 4. *Έστω π ένα πιθανοτικό βυζαντινό Broadcast πρωτόκολλο (σύμφωνα με τον Ορισμό 7) ανεκτικό ενάντια σε $t \leq n-2$ σφάλματα και με πιθανότητα επιτυχίας $3/4+\epsilon$, τότε σε κάθε εκτέλεση του π οι τίμιοι παίκτες πρέπει συνολικά να στείλουν τουλάχιστον $(\epsilon t)^2$ μηνύματα κατά εκτιμώμενη τιμή.*

Υπολογιστική Πολυπλοκότητα. Όπως αποδείχθηκε στο [2] κάθε βυζαντινό Broadcast πρωτόκολλο ανεκτικό σε κακόβουλη πλειοψηφία απαιτεί την ύπαρξη PKI. Έστω, λοιπόν, $\kappa \in \mathbb{N}$ η παράμετρος ασφαλείας του PKI που απαιτείται για την επίλυση του βυζαντινού Broadcast, τότε ο χρόνος που δίνεται στην διάθεση των παικτών (οπότε και του αντίπαλου) πρέπει να είναι πολυωνυμικός ως προς κ, διαφορετικά ο αντίπαλος θα έχει την δυνατότητα να "σπάσει" την ασφάλεια του PKI. Για αυτό, συχνά θα θεωρούμε ότι η παράμετρος κ είναι πολυωνυμική ως προς το πλήθος των παικτών n και θα απαιτούμε τα πρωτόκολλα να χρειάζονται πολυωνυμικό χρόνο ως προς n.

1.3.2 Το Πρωτόκολλο Dolev-Strong

Το διάσημο πρωτόκολλο Dolev-Strong [4] λύνει το πρόβλημα του βυζαντινού Broadcast ενάντια σε οποιονδήποτε PPT t-αντίπαλο όταν $t < n$. Το πρωτόκολλο λειτουργεί ως εξής : ο αποστολέας s στέλνει την υπογραφή του για την τιμή b_s στον γύρο 0 και σε κάθε επόμενο i-οστό γύρο, όπου $i \in [1, t]$, οι τίμιοι παίκτες αποδέχονται την τιμή $b \in \{0, 1\}$ αν και μόνο αν λάβουν i διαφορετικές υπογραφές για την b, συμπεριλαμβανομένης και της υπογραφής του αποστολέα s. Όταν κάποιος παίκτης αποδεχτεί κάποια τιμή b, την υπογράφει και στέλνει όλες τις υπογραφές που έχει λάβει για αυτήν σε όλους τους παίκτες, ώστε στον γύρο $t + 1$, κάθε τίμιος παίκτης να δίνει στην έξοδο 1 αν και μόνο αν το 1 είναι η μοναδική τιμή που έχει αποδεχτεί ως τότε, διαφορετικά να δίνει στην έξοδο 0.

Για την τυπική περιγραφή του πρωτοκόλλου Dolev-Strong (Σχήμα 1.3.1) θα χρειαστεί να ορίσουμε τα r-*έγκυρα πακέτα* υπογραφών για κάποια τιμή $b \in \{0, 1\}$ ως τα σύνολα της μορφής $\{\mathsf{Sign}_v(b)\}_{v \in V}$, για κάποιο $V \subseteq \mathcal{P}$ με $|V| \geq r$ και $s \in V$, και τα *καλά* μηνύματα ως τα μηνύματα της μορφής $\{B\}$, όπου B είναι ένα έγκυρο *πακέτο* υπογραφών για κάποια τιμή $b \in \{0, 1\}$ ή της μορφής $\{B_0, B_1\}$, όπου B_0, B_1 είναι έγκυρα *πακέτα* υπογραφών για τις τιμές 0, 1, αντίστοιχα.

Πρωτόκολλο π_{ds}

Γύρος 0: Κάθε παίκτης $p \in \mathcal{P}$ αρχικοποιεί $A_p^0 = C_{p,0} = C_{p,1} = \emptyset$. Ο αποστολέας s στέλνει σε όλους τους παίκτες το μήνυμα $M_s^0 = \{\{\mathsf{Sign}_s(b_s)\}\}$.

Βρόγχος: Για $r \in [1, t]$, κάθε παίκτης $p \in \mathcal{P}$ εκτελεί τα ακόλουθα βήματα :

1. Αρχικοποιεί $A_p^r = A_p^{r-1}$.
2. Όταν λάβει ένα *καλό* μήνυμα M, για κάθε $B \in M$, κάνει τα ακόλουθα : αν το B είναι ένα r-έγκυρο πακέτο υπογραφών για κάποια τιμή $b \notin A_p^{r-1}$, τότε θέτει $A_p^r \leftarrow A_p^r \cup \{b\}$ και $C_{p,b} \leftarrow C_{p,b} \cup B$.
3. Στέλνει σε όλους τους παίκτες το μήνυμα $M_p^r = \{C_{p,b} \cup \{\mathsf{Sign}_p(b)\}\}_{b \in A_p^r \setminus A_p^{r-1}}$.

Γύρος t + 1: Κάθε παίκτης $p \in \mathcal{P}$ εκτελεί τα ακόλουθα βήματα :

1. Αρχικοποιεί $A_p^{t+1} = A_p^t$.
2. Όταν λάβει ένα *καλό* μήνυμα M, για κάθε $B \in M$, κάνει τα ακόλουθα : αν B είναι ένα $(t + 1)$-έγκυρο πακέτο υπογραφών για κάποια τιμή $b \notin A_p^t$, τότε θέτει $A_p^{t+1} \leftarrow A_p^{t+1} \cup \{b\}$.
3. Τερματίζει και δίνει στην έξοδο 1, αν $A_p^{t+1} = \{1\}$, αλλιώς δίνει στην έξοδο 0.

Σχήμα 1.3.1: Πρωτόκολλο Dolev-Strong

Θεώρημα 5. *Το πρωτόκολλο Dolev-Strong π_{ds} (Σχήμα 1.3.1) είναι ένα βυζαντινό Broadcast πρωτόκολλο (σύμφωνο με τον Ορισμό 6) το οποίο είναι ασφαλές ενάντια σε οποιονδήποτε PPT t-αντίπαλο, όπου $t < n$. Επιπλέον, το πρωτόκολλο απαιτεί $t + 1$ γύρους και έχει $O(n^2)$ πολυπλοκότητα μηνυμάτων με $O(n^3)|\sigma|$ bits, όπου $|\sigma|$ είναι το μήκος κάθε υπογραφής του PKI.*

Η ορθότητα του πρωτοκόλλου βασίζεται στο γεγονός πως αν κάποιος παίκτης λάβει ένα r-έγκυρο πακέτο υπογραφών για κάποια τιμή $b \in \{0, 1\}$ στον γύρο $r \leq t$, τότε μπορεί να δημιουργήσει ένα $(r + 1)$-έγκυρο πακέτο υπογραφών για την b προσθέτοντας την δική του υπογραφή. Μάλιστα, το πρωτόκολλο διατηρεί την ορθότητα του όταν τρέχει για οποιονδήποτε αριθμό γύρων μεγαλύτερο του t, π.χ. για $n - 1$ γύρους ακόμα και αν δεν είναι γνωστή η τιμή του t.

1.3.3 Πρωτόκολλο με Υπογραμμικό Αριθμό Γύρων

Οι Chan, Pass και Shi [11] κατασκεύασαν ένα πιθανοτικό βυζαντινό Broadcast πρωτόκολλο ανεκτικό ενάντια σε $t = (1-\epsilon)n$ σφάλματα που απαιτεί υπογραμμικό αριθμό γύρων ως προς το n. Για την ακρίβεια, το πρωτόκολλο τερματίζει σε $O(\log(1/\delta)/\epsilon)$ γύρους και αποτυγχάνει με πιθανότητα το πολύ $\delta + \mathsf{negl}(\kappa)$, για οποιοδήποτε $\delta > 0$. Εύκολα διαπιστώνει κανείς ότι αυτή η πολυπλοκότητα γύρων είναι υπογραμμική ως προς n όταν $\epsilon = \omega(1/n)$ και ότι απέχει από το κάτω όριο του Πορίσματος 9 κατά έναν παράγοντα $\log(1/\delta)$.

Το πρωτόκολλο βασίζεται στην εκλογή μίας τυχαίας *επιτροπής* με $\log(1/\delta)$ παίκτες ώστε με πιθανότητα $O(\delta)$ να περιέχει τουλάχιστον ένα τίμιο μέλος. Η ιδέα είναι ότι μόνο τα μέλη της *επιτροπής* (καθώς και ο αποστολέας s) θα έχουν το προνόμιο να υπογράφουν τιμές από το $\{0,1\}$ και ότι υπογραφές από όλους τους υπόλοιπους παίκτες του $\mathcal{P}$ θα δεν λαμβάνονται από κανέναν τίμιο παίκτη του $\mathcal{P}$.

Το πρωτόκολλο αποτελεί μία παραλλαγή του Dolev-Strong, στην οποία κάθε γύρος του Dolev-Strong αντιστοιχίζεται σε ένα *στάδιο* του νέου πρωτοκόλλου που αποτελείται από δύο διαδοχικούς γύρους. Στον πρώτο γύρο κάθε σταδίου, κάθε τίμιος παίκτης απλώς προωθεί όλες τις υπογραφές που λαμβάνει στους υπόλοιπους παίκτες, ενώ στον στον δεύτερο γύρο, οι τίμιοι παίκτες της επιτροπής (και μόνο αυτοί) προωθούν όλες τις όλες τις υπογραφές που έχουν λάβει για τις τιμές που έχουν λάβει, προσθέτοντας και την δική τους υπογραφή στις τιμές αυτές.

Για να είναι το πρωτόκολλο ανεκτικό ενάντια σε οποιονδήποτε PPT t-αντίπαλο πρέπει να υποθέσουμε την ύπαρξη της λειτουργίας $\mathcal{F}_{\mathsf{mine}}$ από την οποία θα καθορίζεται αν κάποιος παίκτης έχει το προνόμιο να υπογράψει την τιμή b (δηλαδή είναι στην b-επιτροπή) ή όχι.

Λειτουργία $\mathcal{F}_{\mathsf{mine}}$

Εξόρυξη: Αν κάποιος παίκτης $p \in \mathcal{P}$ καλέσει την $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$ για πρώτη φορά για κάποια τιμή $b \in \{0,1\}$, τότε η $\mathcal{F}_{\mathsf{mine}}$ στρίβει ένα κέρμα το οποίο καθορίζει αν ο p είναι στην b-επιτροπή. Επιπλέον, η $\mathcal{F}_{\mathsf{mine}}$ απαντάει στον p αναλόγως και αποθηκεύει την απάντηση που του δίνει, ώστε αν ο p ξανακαλέσει την $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$, η $\mathcal{F}_{\mathsf{mine}}$ να του δώσει την ίδια απάντηση.

Επιβεβαίωση: Αν κάποιος παίκτης $p \in \mathcal{P}$ καλέσει την $\mathcal{F}_{\mathsf{mine}}.\mathsf{verify}(b,q)$ για κάποιο ζεύγος $(b,q) \in \{0,1\} \times \mathcal{P}$ για να ελέγξει αν ο q είναι στην b-επιτροπή, τότε η $\mathcal{F}_{\mathsf{mine}}$ απαντάει καταφατικά ανν ο q έχει ήδη καλέσει επιτυχώς την $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$, διαφορετικά απαντάει αρνητικά.

Σχήμα 1.3.2: Η Λειτουργία $\mathcal{F}_{\mathsf{mine}}$

Για την τυπική περιγραφή του πρωτοκόλλου που φαίνεται στο Σχήμα 1.3.3, θα πρέπει να ορίσουμε εκ νέου τα *r-έγκυρα πακέτα* υπογραφών για κάποια τιμή $b \in \{0,1\}$ ως τα σύνολα της μορφής $\{\mathsf{Sign}_v(b)\}_{v \in V}$, για κάποιο $V \subseteq \mathcal{P}$ με $|V| \geq r$, $s \in V$ και τις κλήσεις $\mathcal{F}_{\mathsf{mine}}.\mathsf{verify}(b,v)$ να είναι επιτυχείς για τουλάχιστον $r-1$ παίκτες του $V \backslash \{s\}$.

Εύκολα διαπιστώνει κανείς ότι το πρωτόκολλο του Σχήματος 1.3.3 ικανοποιεί τις ιδιότητες της *εγκυρότητας* και του *τερματισμού* του βυζαντινού Broadcast (Ορισμός 6) με πιθανότητα 1. Επιπλέον, παρατηρούμε ότι αν δεν συμβεί κανένα γεγονός ενός από τους παρακάτω τύπους γεγονότων, τότε το πρωτόκολλο εξασφαλίζει και την ιδιότητα της *συμφωνίας* :

– Τύπος A : κάποιος τίμιο παίκτη $p \in \mathcal{P}$ αποδέχεται κάποια τιμή b σε κάποιο στάδιο $r \in [1,R]$, αλλά

Πρωτόκολλο π_{sb}

Έστω $R = \lceil \frac{3}{\epsilon} \cdot \ln \frac{2}{\delta} \rceil$ και ότι η $\mathcal{F}_{\mathsf{mine}}$ υλοποιείται με $p = \frac{1}{\epsilon n} \ln \frac{2}{\delta}$.

Στάδιο 0: Κάθε παίκτης $p \in \mathcal{P}$ αρχικοποιεί $\hat{A}_p^0 = C_{p,0} = C_{p,1} = \emptyset$. Ο αποστολέας s στέλνει σε όλους τους παίκτες το μήνυμα $M_s^0 = \{\{\mathsf{Sign}_s(b_s)\}\}$.

Βρόγχος: Κάθε στάδιο $r \in [1, R]$ αποτελείται από 2 γύρους.

1. Στον πρώτο γύρο, κάθε παίκτης $p \in \mathcal{P}$ εκτελεί τα ακόλουθα βήματα :
   - Αρχικοποιεί $A_p^r = \hat{A}_p^{r-1}$.
   - Όταν λάβει ένα *καλό* μήνυμα M, για κάθε $B \in M$, κάνει τα ακόλουθα : αν το B είναι ένα r-έγκυρο πακέτο υπογραφών για κάποια τιμή $b \notin \hat{A}_p^{r-1}$, τότε θέτει $A_p^r \leftarrow A_p^r \cup \{b\}$ και $C_{p,b} \leftarrow C_{p,b} \cup B$.
   - Στέλνει σε όλους τους παίκτες το μήνυμα $M_p^r = \{C_{p,b}\}_{b \in A_p^r \setminus \hat{A}_p^{r-1}}$.
2. Στον δεύτερο γύρο, κάθε παίκτης $p \in \mathcal{P}$ εκτελεί τα ακόλουθα βήματα :
   - Αρχικοποιεί $\hat{A}_p^r = A_p^r$.
   - Όταν λάβει ένα *καλό* μήνυμα M, για κάθε $B \in M$, κάνει τα ακόλουθα : αν το B είναι ένα r-έγκυρο πακέτο υπογραφών για κάποια τιμή $b \notin \hat{A}_p^{r-1}$ και η κλήση της $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$ είναι επιτυχής, τότε θέτει $\hat{A}_p^r \leftarrow \hat{A}_p^r \cup \{b\}$ και $C_{p,b} \leftarrow C_{p,b} \cup B$.
   - Αν $\hat{A}_p^r \neq A_p^r$, τότε στέλνει σε όλους τους παίκτες $\hat{M}_p^r = \{C_{p,b} \cup \{\mathsf{Sign}_p(b)\}\}_{b \in \hat{A}_p^r \setminus \hat{A}_p^{r-1}}$.

Στάδιο R + 1: Κάθε παίκτης $p \in \mathcal{P}$ εκτελεί τα ακόλουθα βήματα :

1. Αρχικοποιεί $A_p^{R+1} = \hat{A}_p^R$.
2. Όταν λάβει ένα *καλό* μήνυμα M, για κάθε $B \in M$, κάνει τα ακόλουθα : αν το B είναι ένα $(R+1)$-έγκυρο πακέτο υπογραφών για κάποια τιμή $b \notin \hat{A}_p^R$, τότε θέτει $A_p^{R+1} \leftarrow A_p^{R+1} \cup \{b\}$.
3. Τερματίζει και δίνει στην έξοδο 1, αν $A_p^{R+1} = \{1\}$, αλλιώς δίνει στην έξοδο 0.

Σχήμα 1.3.3: Βυζαντινό Broadcast Πρωτόκολλο με Υπογραμμικό Αριθμό Γύρων

κάποιος άλλος τίμιος παίκτης $q \in \mathcal{P}$ δεν αποδέχεται ποτέ την b.

- Τύπος B : κάποιος τίμιο παίκτη $p \in \mathcal{P}$ αποδέχεται κάποια τιμή b στο στάδιο $R+1$, αλλά κάποιος άλλος τίμιος παίκτης $q \in \mathcal{P}$ δεν αποδέχεται ποτέ την b.

Αποδεικνύεται ότι ένα γεγονός τύπου A ή B συμβαίνει με πιθανότητα το πολύ $\delta/2 + \mathsf{negl}(\kappa)$, οπότε έχουμε το παρακάτω θεώρημα :

Θεώρημα 6. *Για κάθε $\epsilon, \delta \in (0,1)$, με $\delta > 2e^{-\epsilon n}$, το πρωτόκολλο π_{sb} (Σχήμα 1.3.3) ένα πιθανοτικό βυζαντινό Broadcast πρωτόκολλο (σύμφωνα με τον Ορισμό 7), ανεκτικό ενάντια σε οποιονδήποτε PPT t-αντίπαλο, όταν $t < (1-\epsilon)n$, το οποίο τερματίζει σε $2\lceil \frac{3}{\epsilon} \cdot \ln \frac{2}{\delta} \rceil + 1$ γύρους και επιτυγχάνει με πιθανότητα $1 - \delta - \mathsf{negl}(\kappa)$. Επιπλέον, έχει πολυπλοκότητα μηνυμάτων $O(n^2)$ με $O(n^2 \ln(1/\delta)/\epsilon)|\sigma|$ bits κατά εκτιμώμενη τιμή, όπου $|\sigma|$ είναι το μήκος κάθε υπογραφής του PKI.*

1.4 Ανώνυμο Broadcast για Οποιονδήποτε Αριθμό Κακόβουλων Παικτών

1.4.1 Το Μοντέλο Ανώνυμων Παικτών

Στο *μοντέλο ανώνυμων παικτών*, θεωρούμε ένα (μεγάλο) σύνολο πιθανών παικτών, το οποίο αποκαλούμε *σύμπαν* $\mathcal{U}$ και υποθέτουμε ότι ένα μικρό ποσοστό των παικτών του $\mathcal{U}$ θα συμμετάσχει ενεργά στην εκτέλεση του πρωτοκόλλου. Επιπλέον, θεωρούμε ότι οι πραγματικά *ενεργοί* παίκτες (συμβολίζουμε με $\mathcal{P} \subseteq \mathcal{U}$), δηλαδή αυτοί που συμμετέχουν στην εκτέλεση, δεν έχουν καμία πληροφορία ο ένας για τον άλλον, ούτε γνωρίζουν κάποιο άνω όριο του πραγματικού αριθμού των ενεργών παικτών.

Εφόσον θα ασχοληθούμε με την περίπτωση των κακόβουλων πλειοψηφιών, θα πρέπει να υποθέσουμε την ύπαρξη ενός σχήματος υπογραφών. Συγκεκριμένα, θα θεωρήσουμε ένα σχήμα υπογραφών με παράμετρο ασφαλείας $\kappa \in \mathbb{N}$ και θα υποθέσουμε ότι το σύμπαν περιέχει $\omega(\mathsf{poly}(\kappa))$ παίκτες, ενώ το $\mathcal{P}$ περιέχει $\mathsf{poly}(\kappa)$ παίκτες. Οι παίκτες του $\mathcal{P}$ θα δημιουργούν μόνοι τους το ζευγάρι ιδιωτικού-δημοσίου κλειδιού τους και θα στέλνουν το δημόσιο κλειδί τους σε μία *αρχή πιστοποίησης (CA)*, η οποία θα τους επιστρέφει ένα πιστοποιητικό εγκυρότητας του κλειδιού τους. Μόλις λάβουν το πιστοποιητικό, οι παίκτες θα μπορούν να παράγουν έγκυρες υπογραφές.

Η επικοινωνία των παικτών του $\mathcal{U}$ θα γίνεται μέσω ενός δικτύου διάχυσης, στο οποίο η παράδοση των μηνυμάτων από τίμιους παίκτες είναι εγγυημένη. Έτσι, θεωρούμε ότι οι παίκτες μπορούν να καλούν την λειτουργία $\mathcal{F}_{\mathsf{diffuse}}$, η οποία θα επιτελεί τον ανάλογο ρόλο της εντολής ``στείλε σε όλους τους παίκτες`` του κλασικού μοντέλου.

Τέλος, ο αντίπαλος θα μπορεί να είναι ισχυρά προσαρμοστικός, αλλά και να ενεργοποιεί $\mathsf{poly}(\kappa)$ παίκτες που δεν προτίθεντο να συμμετάσχουν και να τους επιβάλλει να ακολουθούν τις εντολές του. Μάλιστα, μπορεί εκτός από το να καλεί την λειτουργία $\mathcal{F}_{\mathsf{diffuse}}$ να στέλνει επιλεκτικά μηνύματα στους παίκτες.

1.4.2 Broadcast στο Μοντέλο Ανώνυμων Παικτών

Εύκολα μπορεί να αντιληφθεί κανείς ότι το κάθε πρωτόκολλο στο μοντέλο ανώνυμων παικτών πρέπει να ολοκληρώνεται το πολύ σε $\mathsf{poly}(\kappa)$ γύρους, διαφορετικά ο αντίπαλος μπορεί να πλαστογραφήσει υπογραφές. Εφόσον, λοιπόν, όπως γνωρίζουμε από το κλασικό μοντέλο κάθε Broadcast πρωτόκολλο ανεκτικό σε οποιονδήποτε αριθμό σφαλμάτων διαρκεί στη χειρότερη περίπτωση τόσους γύρους όσος και αριθμός των παικτών, θα πρέπει να υποθέσουμε ότι υπάρχει κάποιος γύρος R, στον οποίον ο αντίπαλος σταματάει να ενεργοποιεί νέους παίκτες.

Έτσι, έχουμε τον παρακάτω ορισμό για το *ανώνυμο Broadcast*:

Ορισμός 10 (Ανώνυμο Broadcast)**.** *Έστω $\mathcal{U}$ ένα σύνολο παικτών. Ένα πρωτόκολλο π είναι πρωτόκολλο ανώνυμου Broadcast, όπου ένας καθορισμένος αποστολέας $s \in \mathcal{U}$ παίρνει ως είσοδο μία αρχική τιμή $b_s \in \{0,1\}$, αν για οποιονδήποτε PPT αντίπαλο και οποιοδήποτε σύνολο $\mathcal{P} \subseteq \mathcal{U}$, οι παρακάτω ιδιότητες ικανοποιούνται με συντριπτική πιθανότητα:*

- ***Εγκυρότητα:*** *Αν ο αποστολέας s είναι τίμιος, τότε όλοι οι τίμιοι παίκτες δίνουν στην έξοδο την τιμή b_s.*
- ***Συμφωνία:*** *Όλοι οι τίμιοι παίκτες δίνουν στην έξοδο την ίδια τιμή $b \in \{0,1\}$.*
- ***Τερματισμός:*** *Όλοι οι τίμιοι παίκτες τερματίζουν και δίνουν στην έξοδο μία τιμή σε $\mathsf{poly}(\kappa)$ γύρους.*

Η δυσκολία του προβλήματος έγκειται στο γεγονός ότι οι παίκτες δεν μπορούν να είναι σίγουροι (σε αντίθεση με το Dolev-Strong) ότι όταν αποδέχονται μία τιμή του αποστολέα, τότε όλοι οι υπόλοιποι παίκτες είτε έχουν ήδη δεχθεί αυτήν την τιμή είτε θα συμμετέχουν στο επόμενο γύρο για να μπορούν πειστούν για την αποδοχή της πριν τερματίσουν.

Η λύση μας βασίζεται στην ανάπτυξη ενός πρωτοκόλλου το οποίο επιτρέπει στους παίκτες να συμφωνούν σε ένα υποσύνολό τους, το οποίο θα περιέχει πάντα όλους τους τίμιους παίκτες.

1.4.3 Συμφωνία Ενεργών Παικτών

Παρουσιάζουμε ένα πρωτόκολλο *συμφωνίας ενεργών παικτών* (APA), όπου οι παίκτες συμφωνούν σε ένα σύνολο ενεργών παικτών $S \subseteq \mathcal{P}$, τέτοιο ώστε κάθε τίμιος παίκτης να περιλαμβάνεται στο S. Ο τυπικός ορισμός του APA είναι ο εξής:

Ορισμός 11 (Συμφωνία Ενεργών Παικτών (APA))**.** *Έστω $\mathcal{U}$ ένα σύνολο παικτών. Ένα πρωτόκολλο π είναι πρωτόκολλο συμφωνίας ενεργών παικτών, αν για οποιονδήποτε PPT αντίπαλο και οποιοδήποτε σύνολο $\mathcal{P} \subseteq \mathcal{U}$, οι παρακάτω ιδιότητες ικανοποιούνται με συντριπτική πιθανότητα:*

- ***Ορθότητα:*** *Αν ένας τίμιος παίκτης p δώσει στην έξοδο το S, τότε $S \subseteq \mathcal{P}$.*
- ***Εγκυρότητα:*** *Αν ένας τίμιος παίκτης p δώσει στην έξοδο το S, τότε $p \in S$.*
- ***Συμφωνία:*** *Όλοι οι τίμιοι παίκτες δίνουν στην έξοδο το ίδιο σύνολο.*
- ***Τερματισμός:*** *Όλοι οι τίμιοι παίκτες τερματίζουν και δίνουν στην έξοδο ένα σύνολο σε $\mathsf{poly}(\kappa)$ γύρους.*

Για την τυπική περιγραφή του APA πρωτοκόλλου (Σχήμα 1.4.1) θα χρειαστεί να ορίσουμε τα (p, r)-*έγκυρα πακέτα* υπογραφών για κάποιον παίκτη $u \in \mathcal{U}$ ως τα σύνολα της μορφής $\{\mathsf{Sign}_v(\mathsf{id}_u)\}_{v \in V}$, για κάποιο $V \subseteq \mathcal{P}$ με $u \in V$ και $|V \cap S_p^{r-1}| \geq r - 1$, και τα *καλά* μηνύματα ως τα μηνύματα της μορφής $\{B_i\}_{i=1}^{l}$, για κάποιο $l \geq 1$, όπου B_i είναι έγκυρα *πακέτα* υπογραφών για κάποιο παίκτη $u_i \in \mathcal{U}$, τέτοια ώστε $B_i = \{\mathsf{Sign}_v(\mathsf{id}_{u_i})\}_{v \in V_i}$, για κάποιο $V_i \subseteq \mathcal{U}$ με $u_i \in V_i$, για κάθε $i \in [1, l]$.

Θεώρημα 7. *Το πρωτόκολλο π_{apa} (Σχήμα 1.4.1) είναι ένα πρωτόκολλο συμφωνίας ενεργών παικτών (σύμφωνα με τον Ορισμό 11) το οποίο είναι ασφαλές ενάντια σε οποιονδήποτε PPT. Επιπλέον, έστω S το κοινό σύνολο εξόδου των παικτών, τότε το πρωτόκολλο τερματίζει σε $|S| \subseteq |\mathcal{P}|$ γύρους..*

Η ορθότητα του πρωτοκόλλου βασίζεται στο γεγονός πως αν ο παίκτης p λάβει ένα (p, r)-έγκυρο πακέτο υπογραφών για κάποιον παίκτη $u \in \mathcal{U}$ στον γύρο r, τότε μπορεί να δημιουργήσει ένα $(q, r + 1)$-έγκυρο πακέτο υπογραφών για τον u, για κάθε q, προσθέτοντας την δική του υπογραφή. Επιπλέον, αποδεικνύεται επαγωγικά ότι κάθε παίκτης q που δεν έχει αποδεχθεί τον u μέχρι τον γύρο r, θα προχωρήσει στον γύρο $r + 1$, οπότε και θα αποδεχθεί τον u πριν τερματίσει.

Εύκολα, μπορούμε να παρατηρήσουμε ότι το παραπάνω πρωτόκολλο λύνει το ανώνυμο Broadcast ως εξής: ο αποστολέας συμμετέχει ενεργά στο APA ανν $b_s = 1$ και οι παίκτες δίνουν στην έξοδο 1 αν και μόνο αν ο s περιλαμβάνεται στην έξοδο του APA.

1.4.4 Ανώνυμο Interactive Consistency

Παρουσιάζουμε ένα *ανώνυμο interactive consistency* πρωτόκολλο, όπου κάθε παίκτης $p \in \mathcal{P}$ παίρνει ως είσοδο μία τιμή $b_p \in \{0, 1\}$ και δίνει στην έξοδο ένα σύνολο ζευγαριών παίκτη τιμής.

Πρωτόκολλο π_{apa}

Αρχικοποίηση: Κάθε παίκτης $p \in \mathcal{U}$ που επιθυμεί να συμμετάσχει, δημιουργεί το $(\mathsf{sk}_p, \mathsf{pk}_p) \leftarrow \mathsf{KeyGen}(1^\kappa)$ και $\mathsf{salt}_p \xleftarrow{\$} \{0,1\}^\kappa$. Έπειτα, στέλνει $\mathsf{id}_p = \mathsf{pk}_p || \mathsf{salt}_p$ στο CA και λαμβάνει το cert_p.

Γύρος 0: Κάθε παίκτης $p \in \mathcal{P}^0$ αρχικοποιεί $S_p^0 = \{p\}$ και καλεί $\mathcal{F}_{\mathsf{diffuse}}(\{\{\mathsf{Sign}_p(\mathsf{id}_p)\}\})$.

Βρόγχος: Για $r = 1, 2, \ldots$, κάθε παίκτης $p \in \mathcal{P}$ εκτελεί τα ακόλουθα βήματα:

1. Αρχικοποιεί $S_p^r = S_p^{r-1}$.
2. Όταν λάβει ένα *καλό* μήνυμα M, για κάθε $B \in M$, κάνει τα ακόλουθα: αν το B είναι ένα (p,r)-έγκυρο πακέτο υπογραφών για κάποιον παίκτη $u \notin S_p^{r-1}$, τότε:
   - αν $u \notin S_p^r$, αρχικοποιεί $C_{p,u}^r = \emptyset$
   - θέτει $S_p^r \leftarrow S_p^r \cup \{u\}$ και $C_{p,u}^r \leftarrow C_{p,u}^r \cup B$.
3. Αν $|S_p^r| \leq r$, τότε τερματίζει και δίνει στην έξοδο το S_p^r, αλλιώς καλεί την $\mathcal{F}_{\mathsf{diffuse}}(\{C_{p,u}^r \cup \{\mathsf{Sign}_p(\mathsf{id}_u)\}\}_{u \in S_p^r \setminus S_p^{r-1}})$ και προχωράει στον γύρο $r+1$.

Σχήμα 1.4.1: Πρωτόκολλο Συμφωνίας Ενεργών Παικτών

Ορισμός 12 (Ανώνυμο Interactive Consistency)**.** *Έστω $\mathcal{U}$ ένα σύνολο παικτών. Ένα πρωτόκολλο π είναι Ανώνυμο interactive consistency πρωτόκολλο, όπου κάθε παίκτης $p \in \mathcal{P}$ παίρνει ως είσοδο μία τιμή $b_p \in \{0,1\}$, αν για οποιονδήποτε PPT αντίπαλο και οποιοδήποτε σύνολο $\mathcal{P} \subseteq \mathcal{U}$, οι παρακάτω ιδιότητες ικανοποιούνται με συντριπτική πιθανότητα:*

- ***Ορθότητα:*** *Αν ένας τίμιος παίκτης p δώσει στην έξοδο το , τότε $A \subseteq \mathcal{P} \times \{0,1\}$.*
- ***Εγκυρότητα:*** *Αν ένας τίμιος παίκτης p δώσει στην έξοδο το A, τότε $(p, b_p) \in A$.*
- ***Συμφωνία:*** *Όλοι οι τίμιοι παίκτες δίνουν στην έξοδο το ίδιο σύνολο.*
- ***Τερματισμός:*** *Όλοι οι τίμιοι παίκτες τερματίζουν και δίνουν στην έξοδο ένα σύνολο σε $\mathsf{poly}(\kappa)$ γύρους.*

Για την τυπική περιγραφή του ανώνυμου interactive consistency πρωτοκόλλου (Σχήμα 1.4.2) θα χρειαστεί να ορίσουμε τα (p,r)-*έγκυρα πακέτα* υπογραφών για κάποιον ζευγάρι $(u,) \in \mathcal{U} \times \{0,1\}$ ως τα σύνολα της μορφής $\{\mathsf{Sign}_v(\mathsf{id}_u || b)\}_{v \in V}$, για κάποιο $V \subseteq \mathcal{P}$ με $u \in V$ και $|V \cap S_p^{r-1}| \geq r-1$, και τα *καλά* μηνύματα ως τα μηνύματα της μορφής $\{B_i\}_{i=1}^l$, για κάποιο $l \geq 1$, όπου B_i είναι έγκυρα *πακέτα* υπογραφών για διακριτά ζευγάρια (u_i, b_i), τέτοια ώστε $B_i = \{\mathsf{Sign}_v(\mathsf{id}_{u_i})\}_{v \in V_i}$, για κάποιο $V_i \subseteq \mathcal{U}$ με $u_i \in V_i$, για κάθε $i \in [1,l]$.

Θεώρημα 8. *Το πρωτόκολλο π_{ic} (Σχήμα 1.4.2) είναι ένα ανώνυμο interactive consistency πρωτόκολλο (σύμφωνα με τον Ορισμό 12) το οποίο είναι ασφαλές ενάντια σε οποιονδήποτε PPT. Επιπλέον, έστω A το κοινό σύνολο εξόδου των παικτών, τότε το πρωτόκολλο τερματίζει σε $|A| \subseteq |\mathcal{P}|$ γύρους..*

Η ορθότητα του πρωτοκόλλου αποδεικνύεται με παρόμοιο τρόπο με εκείνη του APA πρωτοκόλλου. Μάλιστα, από το παραπάνω πρωτόκολλο προκύπτει άμεσα ένα ανώνυμο Broadcast πρωτόκολλο ως εξής: ο s συμμετέχει στο ανώνυμο interactive consistency με είσοδο b_s, ενώ όλοι οι υπόλοιποι παίκτες συμμετέχουν με είσοδο 0 και τελικά, οι παίκτες δίνουν στην έξοδο 1 ανν το ζευγάρι $(s,1)$ περιλαμβάνεται στο σύνολο εξόδου του ανώνυμου interactive consistency.

Πρωτόκολλο π_{ic}

Αρχικοποίηση: Κάθε παίκτης $p \in \mathcal{U}$ που επιθυμεί να συμμετάσχει, δημιουργεί το $(\mathsf{sk}_p, \mathsf{pk}_p) \leftarrow \mathsf{KeyGen}(1^\kappa)$ και $\mathsf{salt}_p \xleftarrow{\$} \{0,1\}^\kappa$. Έπειτα, στέλνει $\mathsf{id}_p = \mathsf{pk}_p||\mathsf{salt}_p$ στο CA και λαμβάνει το cert_p.

Γύρος 0: Κάθε παίκτης $p \in \mathcal{P}^0$ αρχικοποιεί $S_p^0 = \{p\}$ and $A_p^0 = \{(p, b_p)\}$ και καλεί $\mathcal{F}_{\mathsf{diffuse}}(\{\{\mathsf{Sign}_p(\mathsf{id}_p||b_p)\}\})$.

Βρόγχος: Για $r = 1, 2, \ldots$, κάθε παίκτης $p \in \mathcal{P}$ εκτελεί τα ακόλουθα βήματα:

1. Αρχικοποιεί $S_p^r = S_p^{r-1}$ και $A_p^r = A_p^{r-1}$.
2. Όταν λάβει ένα *καλό* μήνυμα M, για κάθε $B \in M$, κάνει τα ακόλουθα: αν το B είναι ένα (p, r)-έγκυρο πακέτο υπογραφών για κάποιο ζευγάρι $(u, b) \notin A_p^{r-1}$, τότε:
   - αν $(u, b) \notin {}^r_p$, αρχικοποιεί $C^r_{p,u,b} = \emptyset$
   - θέτει $S_p^r \leftarrow S_p^r \cup \{u\}$, $A_p^r \leftarrow A_p^r \cup \{(u, b)\}$ και $C^r_{p,u,b} \leftarrow C^r_{p,u,b} \cup B$.
3. Αν $|S_p^r| \leq r$, τότε τερματίζει και δίνει στην έξοδο $\mathsf{Out}(A_p^r)$, αλλιώς καλεί την $\mathcal{F}_{\mathsf{diffuse}}(\{C^r_{p,u,b} \cup \{\mathsf{Sign}_p(\mathsf{id}_u||b)\}\}_{(u,b)\in A_p^r \setminus A_p^{r-1}})$ και προχωράει στον γύρο $r + 1$.

Σχήμα 1.4.2: Ανώνυμο Interactive Consistency Πρωτόκολλο

1.4.5 Κάτω φράγμα πολυπλοκότητας γύρων

Βασιζόμενοι στο κάτω φράγμα του [8] αποδεικνύουμε ότι δεν υπάρχει πιθανοτικό ανώνυμο Broadcast πρωτόκολλο, ανεκτικό ενάντια σε οποιονδήποτε αριθμό σφαλμάτων, που να τερματίζει πάντα σε λιγότερους από $\mathcal{P}$ γύρους.

Θεώρημα 9. *Δεν υπάρχει πιθανοτικό ανώνυμο Broadcast πρωτόκολλο (Ορισμός 10) που τερματίζει σε λιγότερους από $|\mathcal{P}|$ γύρους.*

Άμεση συνέπεια του παραπάνω θεωρήματος είναι ότι το ίδιο φράγμα ισχύει και για τα πρωτόκολλα συμφωνίας ενεργών παικτών και interactive consistency. Συνεπώς, όλα τα πρωτόκολλα που παρουσιάστηκαν παραπάνω στο μοντέλο ανώνυμων παικτών είναι βέλτιστα ως προς την πολυπλοκότητα γύρων χειρότερης περίπτωσης.

1.5 Συμπεράσματα

Το πρόβλημα της βυζαντινής συμφωνίας έχει απασχολήσει ιδιαίτερα την επιστημονική κοινότητα, ωστόσο, μέχρι στιγμής όλες οι υπάρχοντα αποτελέσματα αφορούν το κλασσικό (στατικό) μοντέλο της βυζαντινής συμφωνίας ή υποθέτουν την πλειοψηφία των τίμιων παικτών. Στην παρούσα εργασία, αναπτύσσουμε πρωτόκολλα που λύνουν το πρόβλημα του βυζαντινού Broadcast στο ανώνυμο μοντέλο, κατασκευάζοντας πρωτόκολλα που επιτυγχάνουν την συμφωνία των παικτών σε ένα σύνολο ενεργών παικτών και ένα σύνολο ζευγαριών ενεργών παικτών-τιμών, που περιλαμβάνει τα αντίστοιχα στοιχεία για κάθε τίμιο παίκτη. Μάλιστα, αποδεικνύουμε ότι τα πρωτόκολλά μας επιτυγχάνουν βέλτιστη πολυπλοκότητα γύρων χειρότερης περίπτωσης, η οποία ταυτίζεται με το πλήθος των ενεργών παικτών σε κάθε εκτέλεση.

Chapter 2

Introduction

The problem of *byzantine agreement* (BA) [1, 2] is one of the most fundamental primitives in fault-tolerant distributed computing. It requires a set of parties to reach agreement on a value in the presence of an *adversary*, i.e., a malicious attacker that takes control over some of the parties and who is trying to prevent them from reaching their goal. BA comes in two flavors, *consensus* and *Broadcast*.

In consensus, every party has an input value and the goal is to reach *agreement* on the output, such that if all *honest* (i.e., uncorrupted) parties have the same input value, then they agree upon their common input at the end of the protocol (*validity*). In Broadcast, on the other hand, only a single designated party, called the *sender*, has an input value and the goal is that all honest parties agree on the same value (*agreement*), such that this values equals the sender's input if the sender is honest throughout the execution of the protocol (*validity*).

The bulk of the literature on BA assumes a known set of n parties which communicate via a complete network of point-to-point channels. In this setting, BA can be achieved if and only if at most $t < n/3$ of the parties are malicious, unless a trusted PKI setup for digital signatures is assumed [2, 12]. On the other hand, assuming a secure PKI for existentially unforgeable signatures, one can achieve Broadcast with arbitrarily many ($t < n$) corrupted parties, the so-called *any number of corruptions* [4].

In the last ten years, the advent of blockchains has opened up several new horizons and highlighted the significance of a novel realization of distributed systems. In particular, a common assumption in what is frequently called the *permissionless* model, is that there is no known exact upper bound on the number of participants in the protocol, and that parties might join and leave during the execution of the protocol. This property is often referred to as *dynamic participation* (e.g., [13, 14, 15]), and has become mainstream in the blockchain literature (cf. [16]).

The common assumption in such blockchain systems is that the majority of some resource is in honest hands, drawing the attention of research primarily on BA protocols that assume an honest majority of parties. Hence, the question of whether BA (in particular Broadcast) is possible in the unknown participants (UP) model, where the set of parties is not known in advance, without assuming an honest majority of parties has not yet been addressed. This question naturally raises

the challenge of how can the parties decide that they have sufficiently many messages, so then they can have a consistent output. This may lead one to believe that achieving Broadcast under these assumptions has very few chances.

Surprisingly, in this work, after describing an appropriate definition for Broadcast in the UP model, we answer this question in the affirmative and provide protocols with tight worst-case round complexity. Our protocols make minimal cryptographic assumptions and assume a universe of exponentially[1] many potential parties, out of which only a subset of parties with polynomial[1] size actually participate. We then prove that while our protocols do not require any sources of randomness available to the parties, the worst-case round complexity of our protocols, coincide a lower bound that holds even for randomized protocols in the UP model.

Since our approach is mostly inspired by existing solutions for BA in the classic (known participants model) that tolerate any number of corruptions, before presenting our new results, we choose to do an overview of inherent limitations and relevant BA solutions under the dishonest majority assumption ($t < (1-\epsilon)n$) in the classic setting.

2.1 Related Work

There are plentiful papers studying byzantine agreement in the anonymous setting, i.e., where the nodes do not know each other. Okun and Barak [17] were the first to consider the Byzantine problem in this setting, assuming a set of n static anonymous parties out of which at most t get corrupted. Augustine *et al.* [18] design scalable randomized byzantine protocols in the same setting. Khanchandani and Wattenhofer [19] give BA protocols where the participants have unique but not necessarily consecutive identifiers, where n and t are also unknown but static. Besides, all these papers assume the honest majority and static participants. On the contrary, Momose and Ren [20] solve BA with dynamic participation assuming that the participation level does not fluctuate widely.

A comparison of these results is shown in Figure 2.1.1.

Protocol	Resilience	Round Complexity	Participation	Randomized	PKI
[17]	$t < n/3$	$\Theta(\frac{(n-t)t}{n-2t})$	static		
[18]	$t < (1/4-\epsilon)n$	$O(\mathsf{polylog}(n))$	static	$\surd$	
[19]	$t < n/3$	$O(t)$	static		
[20]	$t < n/2$	$O(1)$	dynamic	$\surd$	$\surd$

Figure 2.1.1: Comparison between BA protocols with unknown/anonymous participants

Most studies on BA under dishonest majority assume static participants [8, 21, 22, 23, 11, 24]. Dolev Strong [4] first provided a protocol that solves BA for any number of corruptions. Garay *et al.* [8] presented a randomized expected $\Theta((t-n/2)^2)$ round protocol and their result was later improved by Fitzi and Nielsen [25] to expected $\Theta(t-n/2)$. Chan, Pass and Shi [11] reduced the round complexity even further to $O(\log(\frac{1}{\delta})\frac{n-t}{n})$, where δ is the error probability of their protocol.

[1] in our security parameter

Subsequent works by Wan *et al.* [23, 22] eradicate the $\log(\frac{1}{\delta})$ term by constructions which retain security under the decisional linear assumption in suitable bilinear groups.

A comparison between the main protocols that tolerate dishonest majorities in the classic model is shown in Figure 2.1.2.

Protocol	Resilience	Round Complexity	Randomized
[4]	$t < n$	$\min\{t+1, n-1\}$	
[8]	$t < n/2 + k$	$\Theta(k^2)$ (expected)	$\surd$
[25]	$t < n$	$\Theta(t - n/2)$ (expected)	$\surd$
[11]	$t < (1-\epsilon)n$	$O(\log(\frac{1}{\delta})\frac{n-t}{n})$	$\surd$

Figure 2.1.2: Comparison between classic BA protocols resilient to dishonest majorities

Regarding the round complexity of BA, Fischer and Lynch [3] showed that any deterministic BA protocol that tolerates t corruptions requires at least $t+1$ rounds. Garay *et al.* [8] showed that no randomized Broadcast protocol tolerating t malicious parties can always terminate in $2n/(n-t)-2$ or fewer rounds, indicating a $\Omega(1/\epsilon)$ lower bound for any protocol that tolerates $t < (1-\epsilon)n$ corruptions. Chan, Pass and Shi [26] finally showed that whenver $t/n = O(1)$ and $\delta = 1/\mathsf{poly}(n)$, then no (even randomized) protocol that completes in worst-case $o(\log(1/\delta)/\log\log(1/\delta))$ rounds can achieve BA with $1-\delta$ probability.

About message complexity limitations, Fischer, Lynch and Merritt [9] proved that in every deterministic BA protocol the honest parties collectively need to send $\Omega(n+t^2)$ messages. This bound was extended by Chan *et al.* [10], in order to prove that if a protocol solves BA with $3/4+\epsilon$ probability against a strongly adaptive adversary, then in expectation, honest parties collectively need to send at least $(\epsilon t)^2$ messages.

2.2 Our contribution

In this work, we provide a byzantine Broadcast protocol that tolerates any number of corruptions in the unknown participants model (UP-Broadcast). In our way towards solving Broadcast in this setting, we first construct two other protocols with the same resilience, the active parties agreement protocol (APA) and an interactive consistency protocol in the unknown participants model (UP-Interactive Consistency). For each one of these protocols we provide formal security proofs and we analyse their worst-case round complexity. Then, we prove that their worst-case round complexity is optimal, by showing that any protocol providing the same guarantees could not always require fewer rounds that our protocols.

Chapter 3

Background

3.1 Cryptographic primitives

First, we introduce the notion of *negligibility*, which is the cornerstone for capturing the security definitions of most cryptographic schemes.

Definition 13 (Negligibility)**.** *A function $f : \mathbb{N} \to \mathbb{R}_+$ is called* negligible*, if for every $c > 0$, there exists some $n_0 \in \mathbb{N}$, such that $f(n) < 1/n^c$, for every $n > n_0$.*

Let $\kappa \in \mathbb{N}$ be the security parameter of a cryptographic scheme, then typically we want the probability of the adversary to violate the security of the scheme, let it be $q(\kappa)$, to be negligible in κ and then, we write $q(n) = \mathsf{negl}(\kappa)$. In particular, throughout this work we will be interested in the security of *digital signature schemes*.

Definition 14 (Signature Scheme)**.** *A (digital) signature scheme is a triple of algorithms* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*. The* KeyGen *algorithm takes as input the 1^κ string and outputs a pair* $(\mathsf{sk}, \mathsf{pk})$ *consisting of a secret key* sk *and a public key* pk*. The* Sign *algorithm takes as input a secret key* sk *and a message $m \in \{0,1\}^{\mathsf{poly}(\kappa)}$ and outputs a signature σ, i.e., $\sigma = \mathsf{Sign}_{\mathsf{sk}}(m)$. The* Verify *algorithm takes as input a public key* pk*, a message $m \in \{0,1\}^{\mathsf{poly}(\kappa)}$ and a signature σ and outputs a bit $b \in \{0,1\}$, i.e., $b = \mathsf{Verify}_{\mathsf{pk}}(m, \sigma)$. For the triple* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *it must hold that $\mathsf{Verify}_{\mathsf{pk}}(m, \mathsf{Sign}_{\mathsf{sk}}(m)) = 1$, for every message $m \in \{0,1\}^*$ and every output* $(\mathsf{sk}, \mathsf{pk})$ *of $\mathsf{KeyGen}(1^\kappa)$.*

Note that for some party to *verify* a pair (m, σ) it must hold a public key pk that corresponds to a secret key sk, such that $\sigma = \mathsf{Sign}_{\mathsf{sk}}(m)$. So, consider a set of $n = \mathsf{poly}(\kappa)$ parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where each party $P_i \in \mathcal{P}$ executes the triple of algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ of some signature scheme, as described in Definition 14, then we will say that $\mathcal{P}$ shares a *public key infrastructure (PKI)*, if every party $P_i \in \mathcal{P}$ is able to verify every pair (m, σ) generated by another party $P_j \in \mathcal{P}$. More, precisely :

Definition 15 (Public Key Infrastructure (PKI))**.** *We will say that a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ shares a public key infrastructure (PKI) if for every pair of parties $P_i, P_j \in \mathcal{P}$, P_i holds a secret key sk_i output by* KeyGen *and P_j holds the corresponding public key pk_i, such that $\mathsf{Verify}_{\mathsf{pk}_i}(m, \mathsf{Sign}_{\mathsf{sk}_i}(m)) = 1$, for every message $m \in \{0,1\}^{\mathsf{poly}(\kappa)}$.*

On the other side, we want that whenever some party P_j manages to verify a pair (m, σ) using the public key pk_i of some party P_i, then this shall imply that the signature σ was generated by P_i, since no other party could be able to generate σ efficiently without the knowledge of sk_i. To make our statement more clear, we will be interested in PKIs such that no PPT adversary can generate a signature σ such that $\mathsf{Verify}_{\mathsf{pk}_i}(b, \sigma) = 1$, for any 1-bit message $b \in \{0, 1\}$, without knowledge of the corresponding sk_i (we restrict our attention to the case of 1-bit messages, since this minimal assumption suffices for the security of the protocols we will present in Chapters 4 and 5). We will say that such PKIs satisfy the *unforgeability* property.

Definition 16 (Unforgeability)**.** *A PKI shared among the parties of a set $\mathcal{P} = \{P_1 \ldots, P_n\}$ satisfies the unforgeability property against any PPT adversary if the probability that the adversary finds a string σ, such that $\mathsf{Verify}_{\mathsf{pk}_i}(b, \sigma) = 1$, for any party $P_i \in \mathcal{P}$ and any value $b \in \{0, 1\}$, without the knowledge of sk_i, is negligible in the security parameter κ.*

For the rest of this work, we will consider that any PKI must always satisfy the *unforgeability* property as stated above. In this case, we will consider that if $\mathsf{Verify}_{\mathsf{pk}_i}(b, \sigma) = 1$, for some value $b \in \{0, 1\}$, then σ was generated by P_i with *overwhelming* probability, i.e., $1 - \mathsf{negl}(\kappa)$. For convenience, we will use $\mathsf{Sign}_p(b)$ to denote the triple $(i, b, \mathsf{Sign}_{\mathsf{sk}_i}(b))$ that the party $p = P_i \in \mathcal{P}$ will send to other parties, so then they can verify that p has indeed signed for b.

3.2 Model and Assumptions

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n *parties*, i.e., PPT Interactive Turing Machines. A *distributed protocol* π is an n-tuple of PPT algorithms $(\pi_1, \ldots, \pi_n)$ used by the parties in $\mathcal{P}$ to interact with each other, where every party P_i executes the algorithm π_i. In many cases, we consider that the protocol π is executed after the parties have established a *message authentication mechanism*, i.e., each party generates a pair of secret-public keys and publishes its public key, so then $\mathcal{P}$ shares a PKI. When we assume *message authentication*, the parties (and the algorithms of the protocol as well) must be PPT with respect to the security parameter κ of the PKI, otherwise they must be PPT in n.

Communication. The parties communicate with each other through *point-to-point* channels. We assume a *synchronous* network, where the parties in $\mathcal{P}$ share a common clock and any protocol executed among the parties in $\mathcal{P}$ proceeds in *rounds*, i.e., at the beginning of a round every party receives all the messages sent to it by the parties in $\mathcal{P}$ in the previous round, does some local computation and sends messages to the parties in $\mathcal{P}$ at the end of the round.

Note that in most of the proofs of efficiency lower bounds in distributed algorithms, we usually consider a variation of the above assumption, where each party sends the messages at the beginning of the round and the messages are received by the parties at the end of the round. Throughout this work, we will also use the former approach for the description of distributed protocols and the latter for the proofs of the efficiency limitations.

3.2.1 Adversary

We consider the *adversary* to be an Interactive Turing Machine that may take control of some parties in $\mathcal{P}$. We will refer to the parties controlled by the adversary as *corrupted* parties and to the rest of the parties as *honest*. We assume that *honest* parties follow the instructions indicated by the protocol to the letter, while *corrupted* parties may deviate from the protocol up to an extent specified by the adversarial model. We will be interested in two main *types of failures*, i.e., ways that corrupted parties deviate from the protocol : *crash failures* and *byzantine corruptions*.

Crash Failures. *Crash failures* follow the protocol up to a specific round r decided by the adversary and never send a message after round r. In particular, we will say that a party *crashes* in round r if it only sends a subset of the messages indicated by the protocol in round r and never sends a message ever after.

Later in this work, it will be made clear that some efficiency bounds on distributed protocols are imposed by their requirement to succeed even if some parties *crash* throughout the execution.

Byzantine Corruptions. *Byzantine corruptions* may completely deviate from the protocol. The adversary can read all the messages destined for them, as well as their internal state and make them behave in an arbitrary malicious manner (e.g. pretend that they don't receive some messages or send conflicting messages).

A protocol which succeeds even in the presence of an adversary which makes parties act as *crash failures* or *byzantine corruptions* will be called *fault-tolerant*. In the specific case that a protocol *tolerates* the presence of *byzantine corruptions*, it will be a *byzantine protocol*.

Static and Adaptive Adversary. A *t-adversary* will be an adversary that takes control of at most t parties which may behave as *byzantine corruptions*. If a t-adversary can choose the set of (up to t) corrupted parties only at the beginning of the execution and has no right to corrupt any additional parties later on, then we will refer to it (as well as its corruptions) as *static*. On the other hand, an *adaptive* t-adversary may also corrupt parties throughout the execution of the protocol depending of its progress, but without exceeding the number of t corruptions in total.

Strongly-Adaptive Adversary. A t-adversary will be called *strongly adaptive*, if it is not only *adaptive*, but can also perform *after-the-fact-removal*, i.e., corrupt some party after observing the messages that it intends to send and remove any of these messages.

Note that throughout this work, by *honest* parties we will be referring only to the parties that remain honest throughout the execution of the protocol, while *corrupted* parties will be considered the ones which get corrupted at any time during the execution.

PPT and Unbounded Adversary. Whenever the parties in $\mathcal{P}$ share a PKI, we will consider the adversary to have polynomial time in κ in every round, in order to process the messages received by corrupted parties and generate the messages it plans to send on behalf of them. However, if

some protocol does not require the existence of a PKI, then this protocol should succeed even if the adversary has *unbounded* computational power.

3.2.2 Complexity Measures

In distributed computing, as in any theoretical computer science field, we are interested in the *efficiency* of our algorithms (or protocols in the case of distributed computing). Particularly, for *fault-tolerant* distributed protocols we usually restrict out attention to three main complexity measures, which are defined below.

Round Complexity. The *round complexity* of a protocol is defined as the maximum number of rounds that some honest party needs to run before terminating, in the worst case execution of the protocol. Note that the *round complexity* indicates the minimum send-receive phases required in order to satisfy the desired properties of the protocol.

Communication Complexity. The *communication complexity* consists of two metrics : the *message complexity* and the *bit complexity*. The *message complexity* of a protocol is defined as the total number of messages that honest parties collectively need to send throughout the worst case execution of the protocol and accordingly, *bit complexity* is defined as the total number of bits that honest parties collectively need to send throughout the worst case execution of the protocol.

Note that in the presence of a PKI, *bit complexity* is bounded from below by the number of signatures that honest parties collectively need to send in the wort case, and that anyway, *bit complexity* is as least as great as the *message complexity*. Since when assuming message authentication, *bit complexity* is heavily relied on the length of the signatures of the PKI, we will be mostly interested in the *message complexity* of authenticated protocols, which reflects the amount of information needed to be exchanged between honest parties, that is imposed primarily by the requirements of the protocols and not by any other cryptographic limitations.

Computational Complexity. The *computational complexity* of a protocol is defined as the maximum over the total computational time that some honest party needs in the worst case execution of the protocol.

Note that the time given to the parties in every round must be the time given to the adversary as well. So, in authenticated protocols, the *computational complexity* must be polynomial in κ, since otherwise, the adversary has a non-negligible probability of violating the *unforgeability* property of the PKI. In general, a fault-tolerant protocol will be considered *efficient* if its round complexity, bit complexity and computational complexity are both polynomial in n (or in κ in case of authenticated protocols).

3.3 Byzantine Agreement

The *Byzantine Agreement* problem was first introduced in [1, 2] (aka Byzantine Generals) as the problem where parties have to agree on a value proposed by a designated *sender* even in the presence

of *byzantine* corruptions. In case that the *sender* behaves honestly, then the value agreed by the honest parties must be the value proposed by the sender, otherwise honest parties just have to agree on a common (possibly default) value. Throughout this work, we will refer to protocols that solve this version of Byzantine Agreement as *byzantine Broadcast* protocols. We will consider that the input value of the designated sender $s \in \mathcal{P}$ is a bit $b_s \in \{0,1\}$ and that every honest party in $\mathcal{P}$ has to output a binary value $b \in \{0,1\}$ by the end of the protocol.

Definition 17 (Deterministic Byzantine Broadcast)**.** *Let $\mathcal{P}$ be a set of n parties. A protocol π executed by the parties of $\mathcal{P}$, where a designated sender $s \in \mathcal{P}$ holds an initial input value $b_s \in \{0,1\}$, is a* deterministic byzantine Broadcast protocol *tolerating any (unbounded) t-adversary, if the following properties are satisfied :*

- ***Validity:*** *If s is honest, then all honest parties output b_s.*
- ***Agreement:*** *All honest parties output the same value $b \in \{0,1\}$.*
- ***Termination:*** *There exists an a-priori-known round R such that the protocol is guaranteed to complete (i.e., all honest parties output a value) within R rounds.*

As shown in [2], there exists no protocol that meets the above definition when $t \geq n/3$. So, for the rest of this work we will restrict our attention to the case of byzantine Broadcast protocols which achieve *computational* security even in the presence of $t \geq n/3$ corruptions. Thus, when not referring explicitly to *deterministic* byzantine Broadcast protocols, we will be mostly interested in protocols that meet the following definition in the *authenticated* setting :

Definition 18 (Byzantine Broadcast)**.** *Let $\mathcal{P}$ be a set of n parties. A protocol π executed by the parties of $\mathcal{P}$, where a designated sender $s \in \mathcal{P}$ holds an initial input value $b_s \in \{0,1\}$, is a* byzantine Broadcast protocol *tolerating any (strongly adaptive) PPT t-adversary, if the properties of Definition 17 are satisfied with overwhelming probability, assuming the existence of a PKI (according to Definition 15).*

In most *byzantine Broadcast* protocols that meet the above definition, the property of *termination* is satisfied with probability 1, while *validity* and *agreement* are satisfied simultaneously with probability $1 - \mathsf{negl}(\kappa)$. If we allow the error probability of the protocol to be non-negligible in κ, then the protocol will be called *randomized*.

Definition 19 (Randomized Byzantine Broadcast)**.** *Let $\mathcal{P}$ be a set of n parties and δ be an error probability strictly less than $1/2$. A protocol π executed by the parties of $\mathcal{P}$, where a designated sender $s \in \mathcal{P}$ holds an initial input value $b_s \in \{0,1\}$, is a* randomized byzantine Broadcast protocol *tolerating any (strongly adaptive) PPT t-adversary, if the properties of Definition 17 are satisfied with probability at least $1 - \delta$, assuming the existence of a PKI (according to Definition 15).*

Note that we allow δ to be a function of the security parameter κ and that a *randomized byzantine Broadcast protocol* meets the Definition 18 of *byzantine Broadcast protocols* when δ is negligible in κ. The restriction $\delta < 1/2$ on the error probability is set to make clear that *randomized protocols* are of type *Monte Carlo*, i.e., always terminate in polynomial (in κ or n) number of rounds and that the rest of the desired properties are satisfied with probability strictly more than $1/2$. The choice of $1/2$ over any other constant number is made for convenience in the description of some impossibility results (some of which will be presented in this work).

Relevant Problems. Apart from the version of Byzantine Agreement described in [1, 2], there is another interesting problem of distributed agreement called *consensus*. In the *consensus* problem we assume that every party in $\mathcal{P}$ has an input message $b_p \in \{0,1\}$ and that all parties have to output the same value $b \in \{0,1\}$ by the end of the protocol. If all honest parties have the same input value b, then b must be the common value of the honest parties.

Definition 20 (Consensus)**.** *Let $\mathcal{P}$ be a set of n parties. A protocol π executed by the parties of $\mathcal{P}$, where every party $P_i \in \mathcal{P}$ holds an initial input value $b_i \in \{0,1\}$, is a consensus protocol tolerating any (unbounded) t-adversary, if the following properties are satisfied :*

- ***Validity:*** *If all honest have the same input value $b \in \{0,1\}$, then all honest parties output b.*
- ***Agreement:*** *All honest parties output the same value $b \in \{0,1\}$.*
- ***Termination:*** *There exists an a-priori-known round R such that the protocol is guaranteed to complete (i.e., all honest parties output a value) within R rounds.*

Note that there does not exist any *consensus protocol* that tolerates any t-adversary when $t \geq n/2$, even if we relax the above definition to allow *randomized* protocols as we did for Broadcast (see Definition 19). Intuitively, the reason for that is that in the scenario where there are $n/2$ honest parties with input 0 and $n/2$ corrupted parties which behave honestly as they had input 1, then the honest parties cannot figure out which one of the values is the common input of all honest parties. In other words, the validity property of consensus is such that can only be achieved when the majority of the *votes* on some value can be trusted to support the value of at least one honest party.

Thus, there is no way to agree on the common input of the honest parties (while preserving agreement anyway) when $t > n/2$. So, the question is what is the best form of agreement on honest parties inputs that can be achieved even if $t > n/2$. The answer is that honest parties can just agree on the input of every party in $\mathcal{P}$, without being able to distinguish between the inputs of honest and corrupted parties. In particular, all honest parties can output the same n-tuple of values $(b^1, \ldots, b^n)$ indicating that each value b^i is considered to be the input of every party $P_i \in \mathcal{P}$. This problem is called *interactive consistency* and is properly defined below :

Definition 21 (Interactive Consistency)**.** *Let $\mathcal{P}$ be a set of n parties. A protocol π executed by the parties of $\mathcal{P}$, where every party $P_i \in \mathcal{P}$ holds an initial input value $b_i \in \{0,1\}$, is an interactive consistency protocol tolerating any (unbounded) t-adversary, if the following properties are satisfied:*

- ***Validity:*** *If some party $P_i \in P$ is honest, then the i-th value of the output tuple of all honest parties is b_i.*
- ***Agreement:*** *All honest parties output the same binary n-tuple.*
- ***Termination:*** *There exists an a-priori-known round R such that the protocol is guaranteed to complete (i.e., all honest parties output a value) within R rounds.*

Note that *interactive consistency* is achievable whenever *Broadcast* is achievable, since assuming the existence of a Broadcast protocol, then we can easily construct an *interactive consistency* protocol by running Broadcast with designated sender P_i and input b_i, for every $P_i \in \mathcal{P}$, and then, collecting

all outputs into an n-tuple of values. Hence, *non-deterministic interactive consistency* can be solved even when $t > n/3$ in the authenticated setting.

Resilience. Up to this point, it must have been made clear to the reader that the number of corrupted parties that a distributed protocol can *tolerate* varies depending on the desired properties and the setting of the protocol. Whenever, a protocol succeeds only in the presence of $t < n/2$ corruptions, then we will say that it requires an *honest majority*, while if it is *resilient* to $t < (1-\epsilon)n$ corruptions, for any arbitrarily small constant $\epsilon > 0$, then we will say that it can *tolerate* a *dishonest majority*. In the special case that the protocol succeeds even in the presence of t corruptions for any $t < n$ (e.g. even if $t = n - O(1)$), then we will say that the protocol can tolerate *any number of corruptions*. Since $t = n - 1$ is trivial, we will be mostly interested in the case $t \leq n - 2$.

3.4 Mathematical Background

For the detailed analysis of some randomized protocols and lower bounds in Chapter 4, we will need to borrow some fundamental tools from the Probability Theory. We will denote with $\Pr[A] \in [0, 1]$, the probability of the event A and with $A \cup B$ and $A \cap B$ the *disjunction* and *conjunction* of A and B, i.e., the event that either A or B occurs and the event that both A and B occur, respectively. The probabilities of the events $A \cup B$ and $A \cap B$ are related according to the *disjunction rule*, which indicates that $\Pr[A \cup B] = \Pr[A] + \Pr[B] - \Pr[A \cap B]$.

The probability that the event A occurs given that the event B occurs will be called the *conditional* probability of A given B and will be denoted with $\Pr[A|B] = \Pr[A \cap B]/\Pr[B]$. So, let $\{B_i\}_{i=1}^n$ be a sequence of *mutually exclusive* and *collectively exhaustive* events, i.e., $\Pr[B_i \cap B_j] = 0$, for every $i, j \in [1, n]$ with $i \neq j$ and $\sum_{i=1}^n \Pr[B_i] = 1$, then by the *law of total probability*, we have that $\Pr[A] = \sum_{i=1}^n \Pr[A \cap B_i] = \sum_{i=1}^n \Pr[A|B_i]\Pr[B_i]$.

Let X be a *random variable*, then we will denote with $\mathbb{E}[X]$ the *expected* value of X. The famous *Markov*'s inequality indicates that the probability that X has a larger value than $\mathbb{E}[X]$, let's say $l > \mathbb{E}[X]$, is inversely proportional to l.

Fact 22 (Markov's Inequality)**.** *Let $X > 0$ be a random variable with expected value $\mathbb{E}[X]$, then for every $l > \mathbb{E}[X]$, it holds that $\Pr[X \geq l] \leq \mathbb{E}[X]/l$.*

We will say that X follows the *binomial distribution* with parameters n and p, if $X = \sum_{i=1}^n X_i$, where $X_i \in \{0, 1\}$ are *independent* random variables with $\Pr[X_i = 1] = p$, for every $i \in [1, n]$. The *Chernoff Bound* indicates that the probability of such a random variable to be larger than $(1 + \tau)\mathbb{E}[X]$ is exponentially small in τ, for every $\tau > 0$.

Fact 23 (Chernoff Bound)**.** *Let $X_i \in \{0, 1\}$ be n independent random variables, then for their sum $X = \sum_{i=1}^n X_i$ and every $\tau > 0$, it holds that $\Pr[X > (1 + \tau)\mathbb{E}[X]] \leq \exp(-\tau \min\{\tau, 1\}\mathbb{E}[X]/3)$.*

Since for the case that X follows the *binomial distribution* with parameters n and p we have that $\mathbb{E}[X] = np$, we conclude that :

Corollary 24. *Let X be a random variable that follows the binomial distribution with parameters n and p, then for every $\tau > 0$, it holds that $\Pr[X > (1 + \tau)np] \leq \exp(-\tau \min\{\tau, 1\}np/3)$.*

Chapter 4

Byzantine Agreement for Dishonest Majority

4.1 Complexity Lower Bounds

4.1.1 Round Complexity

Fischer and Lynch [3] showed that every deterministic byzantine Broadcast protocol that tolerates up to t corruptions requires at least $t+1$ rounds of communication, regardless of n. The $t+1$ lower bound was later proved to hold even in the presence of a message authentication mechanism [4, 5] and ever further to protocols that can only tolerate up to t crash faults [6, 7]. Although many of these proofs are of great interest, most of them are pretty complex and use techniques which are out of the scope of this work.

Instead, we will present a simple *bivalency* proof for the $t+1$ lower bound which was proposed by Aguilera and Toueg in [27] for the problem of *consensus*. The *bivalency* argument was first introduced in the famous FLP impossibility result [28] to prove that there exists no deterministic protocol that solves consensus in the *asynchronous* model even in the presence of only one crash fault. Similar approaches to the one of [27] were presented independently by Moses and Rajsbaum [29] and by Bar-Joseph and Ben-Or [30], applying the *bivalency* argument in the synchronous model.

Before giving the formal proof for the $t+1$ lower bound, we will first have to introduce some of the terminology used by Aguilera and Toueg, which is borrowed from FLP. So, let $\mathcal{P}$ be a set of n parties, then a *configuration* of the the system is defined as the internal state of all of the parties in $\mathcal{P}$. For instance, in the *initial* configuration for the consensus problem, every party holds an initial input and has an empty memory, while for Broadcast, only the designated sender holds an initial input and every party has an empty memory. In the asynchronous model, the system moves from one configuration to the next when a single party receives a message form its buffer and thus, changes its internal state. However, in the synchronous model we assume that within a round, all parties receive all the messages (they are supposed to receive) simultaneously at the end of the round. Therefore, in the synchronous model, we can refer to the configuration of round r, which consists of the internal state of all parties at the end of round r.

We are now ready to introduce the notion of *bivalency*. We will call a configuration C 0-*valent* if 0 is the only value that the honest parties may agree on when resuming the execution of a protocol from the configuration C. Accordingly, a configuration C will be called 1-*valent*, if 1 is the only reachable decision from C. If a configuration C is either 0-*valent* or 1-*valent*, i.e., the common output of the honest parties is already determined, then we will call C *univalent*, otherwise it will be called *bivalent*. Thus, a k-round execution α_k will be called 0-*valent*, 1-*valent*, *univalent* or *bivalent* if the configuration of round k is 0-*valent*, 1-*valent*, *univalent* or *bivalent*, respectively. In addition, any execution α that is identical to α_k up to round k will be called an *extension* of α_k.

We can now use the above terminology to prove the $t+1$ lower bound. While the original proof concerns the problem of consensus, we will show that with an argument almost identical to the one for consensus, we can show the same lower bound for the problem of Broadcast. We choose to unfold the proof as presented in [31], where the only difference from the original version is the expression of the intermediate lemmas. Note that the proof works even when the adversary takes control of at most one party in every round and when the corrupted parties behave like crash faults, which is of course a mush more benign model of failure than the byzantine corruptions.

The idea roughly proceeds as follows : assume that there exists a deterministic consensus protocol π that tolerates up to t crash failures, out of which at most one corruption occurs in every round, and that always terminates in at most t rounds. Then we show that i) there exists a bivalent initial (0-round) configuration in the consensus setting (same for Broadcast), ii) for every round $k \in [0, t-1]$, there exists a bivalent k-round execution of π in which at most k parties have crashed and iii) there exists a t-round execution of π in which at most t parties have crashed and two honest parties decide differently, which completes the proof.

Theorem 10. *There does not exist any deterministic consensus protocol (according to Definition 20) that terminates in fewer than $t+1$ rounds, when $t \leq n-2$, even when the adversary is restricted to make at most one party crash in every round.*

Proof. Assume for the contrary that there exists a deterministic consensus protocol π that tolerates up to t crash failures that occur in different rounds, and which always terminates in at most t rounds. Without loss of generality, we assume that the protocol π is *loquacious*, i.e., in every round, every party is supposed to send a message to every party. We will show that there exists a t-round execution with at most t crash failures, in which two honest parties decide differently, violating the agreement property of Definition 20.

Lemma 25. *There exists a bivalent initial configuration in the setting of consensus.*

Proof. First, we observe that the initial configuration C_0 in which every party has input value 0 is 0-valent, because of the validity property of consensus. Accordingly, the initial configuration C_1 in which every party has input value 1 is 1-valent. So, assume that every initial configuration is univalent, then there must exist two configurations C'_0, C'_1 which differ in the input value of a single party $p \in \mathcal{P}$ and are 0-valent and 1-valent, respectively. Then, we can think of the executions α_0, α_1 which have as initial configurations the C'_0, C'_1, respectively, and in which the party p is crashed in round 1 before sending any messages to the other parties of $\mathcal{P}$ and no other corruption occurs ever after. Hence, the honest parties have exactly the same view in executions α_0, α_1, and thus, they decide the same value, which contradicts our assumption. □

Lemma 26. *For every $k \in [0, t-1]$, there exists a bivalent k-round execution with at most k crash failures.*

Proof. We will prove the desired statement by induction on the round number k. As we have shown in Lemma 25, the statement holds for $k = 0$. Now, assume that for some round $k \in [0, t-2]$ there exists a bivalent k-round execution α_k with at most k crash failures, we will show that there exists a bivalent 1-round extension of α_k (i.e., $(k+1)$-round execution) with at most $k+1$ crash failures.

Assume for contradiction that every 1-round extension of α_k with at most $k+1$ crash failures is univalent. Then, the $(k+1)$-round execution obtained by extending α_k for one round such that no failure occurs in round $k+1$, let it be α^*_{k+1}, is univalent and w.l.o.g., assume that it is 1-valent. Since α_k is bivalent and every 1-round extension of α_k is univalent, then there must be another 1-round extension of α_k, let it be α^0_{k+1}, that is 0-valent. Note that α^*_{k+1} and α^0_{k+1} differ only in round $k+1$ and thus, there must be a single party $p \in \mathcal{P}$ that crashes in round $k+1$. Hence, in round $k+1$ the party p may have failed to send a message to the parties of some subset of $\mathcal{P}$, let it be $Q = \{q_1, q_2, \ldots, q_m\}$, for some $m \in [0, n]$. Thus, for every $i \in [1, m]$, we can define α^i_{k+1} to be a $(k+1)$-round execution that is identical to α^{i-1}_{k+1}, except that p sends a message to q_i before it crashes. Due to our assumption, for every $i \in [0, m]$, α^i_{k+1} is univalent, and therefore, we are led to a contradiction in both of the two following cases :

- if α^i_{k+1} is 0-valent for all $i \in [0, m]$, then so is α^m_{k+1}, which differs from α^*_{k+1} only to the fact that p crashes at the end of round $k+1$ after sending all of its messages. So, we can think of the extension α^* of α^*_{k+1}, in which the party p crashes at the beginning of round $k+2 \leq t$ before sending any messages to the other parties of $\mathcal{P}$ and no other failure occurs ever after. Since, α^*_{k+1} is 1-valent, every honest party must decide 1 in α^*, while in the extension α^m of α^m_{k+1} where no party crashes after round $k+1$, every honest party must decide 0. However, every honest party has the same view of the protocol in α^* and α^m, which is a contradiction.
- if there exists some $i \in [1, m]$, such that α^{i-1}_{k+1} is 0-valent and α^i_{k+1} is 1-valent, then the only difference of α^i_{k+1} from α^{i-1}_{k+1} is that p manages to send a message to q_i before it crashes. So, think of the extensions α^{i-1} and α^i of α^{i-1}_{k+1} and α^i_{k+1}, respectively, in which the party q_i crashes at the beginning of round $k+2 \leq t$ before sending any messages to the other parties of $\mathcal{P}$ and no other party crashes ever after. Then, every honest party has the same view of the protocol in α^{i-1} and α^i, while every honest party has to output 0 in α^{i-1} and 1 in α^i, a contradiction.

□

Lemma 27. *There exists a t-round execution with at most t crash failures in which two honest parties decide differently.*

Proof. As we have shown in Lemma 26, there exits a bivalent $(t-1)$-round execution α_{t-1} with at most $t-1$ crash failures. Since our protocol always terminates in at most t rounds, then in every 1-round extension of α_{t-1} all honest parties must be up to deciding the same value. So, let α^0 be the 1-round extension of α_{t-1} such that no party fails in round t and w.l.o.g. assume that all honest parties decide 0 in α^0. Since α_{t-1} is bivalent, there must be a another 1-round extension of α_{t-1} such that all honest parties decide 1, let it be α^1. Note that α^0 and α^1 differ only in round t and thus, there must be a party p that crashes in round t of α^1 and, additionally, fails to send

a message to some honest party $u \in \mathcal{P}$. Hence, we can think of the execution α^* that is identical to α^1, except that p manages to send a message to u before it crashes. Therefore, by the end of round t, u cannot distinguish between α^* and α^1, while every other honest party $v \in \mathcal{P}\backslash\{u\}$ cannot distinguish between α^* and α^0 (such honest party v exists since $t \leq n-2$). So, in α^* u decides 1, while v decides 0, which violates the agreement property of consensus. □

□

Note that the validity property of consensus was mentioned only in Lemma 25 in order to prove the existence of a bivalent initial configuration. Hence, by modifying sightly the above proof, we can show that $t+1$ rounds are necessary even in the case of Broadcast. So, we can claim that :

Theorem 11. *There does not exist any deterministic Broadcast protocol (according to Definition 17) that terminates in fewer than $t+1$ rounds, when $t \leq n-2$, even when the adversary is restricted to make at most one party crash in every round.*

Proof. We will show that there exists a bivalent initial configuration in the Broadcast setting. The rest of the proof in identical to one we presented for Theorem 10.

Lemma 28. *There exists a bivalent initial configuration in the setting of Broadcast.*

Proof. Let $s \in \mathcal{P}$ be the designated sender and let C_0, C_1 be the initial configuration where s holds an input value $0, 1$, respectively. So, assume that both configurations C_0, C_1 are univalent, then we can think of the executions $\alpha^{(0)}, \alpha^{(1)}$, which start from C_0, C_1, respectively and where s never crashes. By the validity property of Broadcast, all honest parties decide $0, 1$ in $\alpha^{(0)}, \alpha^{(1)}$, respectively and thus, $\alpha^{(0)}$ is 0-valent and $\alpha^{(1)}$ is 1-valent. However, we can also think of the executions α^0, α^1 which are identical to $\alpha^{(0)}, \alpha^{(1)}$, respectively, except that s crashes in round 1 before sending any messages to the parties of $\mathcal{P}$. Hence, the honest parties have exactly the same view in executions α^0, α^1, and thus, they decide the same value, which is a contradiction. □

□

Garay *et al.* [8] first provided a lower bound on the round complexity of the byzantine Broadcast protocols which holds even for randomized protocols with success probability strictly greater than $1/2$ (Definition 19). In particular, they used an information flow argument to show that there exists no randomized byzantine Broadcast protocol, tolerating up to $t \leq n-2$ corruptions, that always terminates in at most $2n/(n-t)-2$ rounds. Note that for $t = n-2$, this indicates that every randomized byzantine Broadcast protocol requires at least $n-1$ rounds in the wort case.

We will now give a high-level description of their approach for the $t = n-2$ case and then present a formal proof for the general case of $t \leq n-2$. The idea simply works as follows : consider a set of n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, where only two parties are honest and that there exists a randomized byzantine Broadcast protocol π that always terminates in at most $n-2$ rounds. Then, we can think of the following configuration: assume that the parties of $\mathcal{P}$ are connected with bilateral channels as shown in Figure 4.1.1, i.e., every party P_i communicates only with the parties P_{i-1}, P_{i+1}.

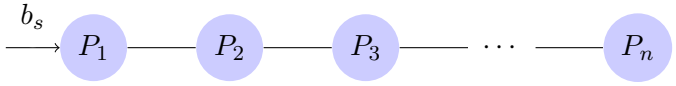

Figure 4.1.1: Configuration where byzantine Broadcast requires at least $n-1$ rounds

In this configuration, any honest party P_i cannot distinguish between the two following scenarios : i) every party in $\mathcal{P}\backslash\{P_{i-1}, P_i\}$ is corrupted and ii) every party in $\mathcal{P}\backslash\{P_i, P_{i+1}\}$ is corrupted. So, since the protocol π satisfies the agreement property, every party P_i has to output the same value with both P_{i-1} and P_{i+1} and thus, all parties in P have to output the same value. In addition, if the designated sender is P_1, then P_1 must output b_s because of the validity property and therefore, all parties in P have to output b_s. In this case, however, every party P_i has no information about the input value b_s before round $i-1$. Hence, if P_n terminates by round $n-2$, then its output will be equal to b_s with probability $1/2$, which is a contradiction since the error probability δ of π must be strictly less than $1/2$.

Using standard partition techniques we can generalize the above argument in order to prove the following lower bound :

Theorem 12. *There does not exist any randomized byzantine Broadcast protocol (according to Definition 19) that terminates in fewer than* $2n/(n-t)-1$ *rounds, when* $t \leq n-2$.

Proof. Let $\mathcal{P}$ be a set of n parties with $t \leq n-2$ corruptions and assume for the contrary that there exists a randomized byzantine Broadcast protocol (according to Definition 19) $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ that always terminates in at most $2n/(n-t)-2$ rounds when executed by the parties of $\mathcal{P}$. Then, we can consider the following configuration: assume that the parties are divided into $k = 2n/(n-t)$ sets $G_1, G_2, \ldots, G_k$ with $|G_i| = (n-t)/2$, for every $i \in [1,k]$. Now, for every $i \in [1,k]$, we can consider a protocol $\bar{\pi}_i$, that is the same as π_i except that :

- if $i = 1$, then the parties in G_1 ignore all the messages sent to them except for those from $G_1 \cup G_2$ and only send messages to the parties of $G_1 \cup G_2$.
- if $i \in [2, k-1]$, then the parties in G_i ignore all the messages sent to them except for those from $G_{i-1} \cup G_i \cup G_{i+1}$ and only send messages to the parties of $G_{i-1} \cup G_i \cup G_{i+1}$.
- if $i = k$, then the parties in G_k ignore all the messages sent to them except for those from $G_{k-1} \cup G_k$ and only send messages to the parties of $G_{k-1} \cup G_k$.

In addition, for every $i \in [1, k-1]$ and $b \in \{0,1\}$, we can define the scenario $S_i^{(b)}$ as follows :

- the designated sender is in G_1 and has an input value $b_s = b$.
- every party in $\mathcal{P}\backslash\{G_i \cup G_{i+1}\}$ is corrupted.
- the honest parties in G_i, G_{i+1} execute the protocols π_i, π_{i+1}, respectively, while the corrupted parties of every G_j with $j \notin \{i, i+1\}$ execute the protocol $\bar{\pi}_j$.

Thus, in scenario $S_1^{(b)}$ the honest parties of $G_1 \cup G_2$ have to output b because of the validity of π. However, in the configuration we described above, the behavior of the parties of $G_1 \cup G_2 \cup G_3$ towards the parties of G_2 is identical regardless of whether they execute π or $\bar{\pi}$. Therefore, in

scenario $S_1^{(b)}$ the honest parties of G_2 cannot distinguish between the scenarios $S_1^{(b)}$ and $S_2^{(b)}$ and hence, they have to output b in scenario $S_2^{(b)}$ as well. Thus, because of the agreement property of π, in scenario $S_2^{(b)}$ the honest parties of G_3 also have to output b.

Applying this logic repeatedly, we have that for every $i \in [2, k-1]$, the honest parties in G_i cannot distinguish between the scenarios $S_{i-1}^{(b)}$ and $S_i^{(b)}$ and thus, they have to output b in both $S_{i-1}^{(b)}$ and $S_i^{(b)}$. Nonetheless, in every scenario $S_i^{(b)}$, the parties of G_i have no information about b before round $i-1$. So, in scenario $S_{k-1}^{(b)}$ the view of the honest parties of G_k is independent of b by round $k-2$ and thus, if the input value b is chosen uniformly from $\{0,1\}$, then the parties of G_k output b with probability $1/2$, which violates the restriction of Definition 19 on π's error probability δ. □

Corollary 29. *There does not exist any randomized byzantine Broadcast protocol (according to Definition 19) that terminates in fewer than $\Omega(n)$ rounds, when $t = n - O(1)$. In particular, there does not exist any randomized byzantine Broadcast protocol that terminates in fewer than $n-1$ rounds, when $t = n-2$.*

Corollary 30. *There does not exist any randomized byzantine Broadcast protocol (according to Definition 19) that terminates in fewer than $\Omega(\frac{1}{\epsilon})$ rounds, when $t = (1-\epsilon)n$.*

Early-Stopping. The lower bounds on the round complexity that we showed above all indicate the number of rounds that honest parties need to reach agreement in the worst case, which is in most cases one where the adversary actually corrupts t parties. So, one could ask if there are any lower bounds on the round complexity, assuming that the actual number of corruptions, let it be f, is less than t, i.e., the adversary does not exhaust the boundary on the number of parties it is allowed to corrupt. Dolev and Reischuk [32] answered this question in the affirmative, showing that any deterministic Broadcast protocol that tolerates up to t crash failures, requires at least $\min\{n-1, t+1, f+2\}$ rounds in every execution where the adversary actually corrupts at most f parties. Up to this day, this bound has not been proven to be tight for protocols that tolerate a dishonest majority, while there are plenty of *early-stopping* protocols in the literature that terminate in $\min\{n-1, t+1, f+2\}$ rounds assuming an honest majority.

4.1.2 Communication Complexity

Dolev and Reischuk [9] first provided a lower bound on the number of messages that honest parties need to send in every deterministic byzantine Broadcast protocol that satisfies the properties of Definition 17. In particular, they showed that every deterministic byzantine Broadcast protocol must have a message complexity of $\Omega(n + t^2)$. Despite the nature of byzantine Broadcast protocols being such that it is hard to estimate the exact number of messages that some particular party has to send, they managed to show that there is an execution with $t/2$ corruptions such that the honest parties must collectively send at least $t/2$ messages to each one of the corruptions. Hence, they came up with the following theorem :

Theorem 13. *Let π be a deterministic byzantine Broadcast protocol (according to Definition 17) that tolerates up to $t \leq n-2$ corruptions, then there is an execution of π where the honest parties collectively need to send at least $\max\{(n-1)/2, t^2/4\}$ messages.*

Proof. First, we observe that one of the values $b \in \{0, 1\}$ must have the property that there exists a set $Q \subseteq \mathcal{P} \backslash \{s\}$ with at least $(n-1)/2$ parties that do not output b if they receive no messages (they may even not terminate at all). Without loss of generality, assume that 0 has the above property, then in the execution where every party in $\mathcal{P}$ is honest and the designated sender s holds an input value 0, the honest parties have to send at least $(n-1)/2$ messages, due to the validity property of Broadcast.

Thus, it suffices to prove the $t^2/4$ lower bound. So, let π be a deterministic byzantine Broadcast protocol π that tolerates up to $t \leq n-2$ corruptions. Let $V \subseteq Q$ be a subset of Q with $t/2$ parties and let U be the set of remaining parties including the sender, i.e., $U = \mathcal{P} \backslash V$ (such a set V exists since $t < n-1$), then we can consider the execution α of π that proceeds as follows :

- every party in U is honest and the sender has an input value 0.
- all parties in V are corrupted and not send any messages to each other.
- every party in V behaves like an honest party towards the parties in U except that it *ignores* (i.e., acts as if it does not receive) the first $t/2$ messages it receives from parties in U.

Note that in execution α only $t/2$ parties are corrupted and the sender is honest. So, due to the validity property of Broadcast, every honest party in U outputs 0. We will show that in execution α, the honest parties collectively send at least $t/2$ messages to each one of the parties in V and thus, send at least $t^2/4$ messages in total, which completes the proof.

So, assume for contradiction that there exists a party $p \in V$, such that the parties in U collectively send less than $t/2$ messages to p in execution α. Then, let $A(p) \subseteq U$ be the set of honest parties that send a message to p in α (potentially including the sender s), then $|A(p)| < t/2$ and thus, we can think of the following execution α', which is identical to α, except that :

- p is honest and all parties in $V \backslash \{p\}$ *ignore* any messages they receive from p.
- every party in $A(p)$ is corrupted and behaves like an honest party, except that it does not send any messages to p.

Hence, in execution α' at most t parties are corrupted and the parties in $V \cup A(p)$ behave towards the parties of U exactly as they do in α. Moreover, p behaves towards the parties of U exactly as it does in α, as it simulates the behavior of an honest party who does not receive any of the (at most) $t/2$ messages it is supposed to get. Therefore, every honest party $q \in U \backslash A(p)$ cannot distinguish between the executions α and α', and thus, outputs 0 in α' as well. However, p does not receive any messages in α' and thus, does not output 0, a contradiction. □

Quite recently, the $\Omega(t^2)$ lower bound on the message complexity of byzantine Broadcast protocols was proven to hold even for randomized protocols with success probability strictly greater than 3/4. As shown by Abraham *et al.* [10], in any randomized byzantine Broadcast protocol (Definition 19) with error probability $\delta \leq 1/4 - \epsilon$, for some $\epsilon > 0$, the honest parties collectively need to send at least $(\epsilon t)^2$ messages in expectation.

They manage to extend the proof of Dolev and Reischuk (Theorem 13) we presented above by relying heavily on the ability of the adversary to be *strongly* adaptive. In particular, for the $\Omega(t^2)$ lower bound to hold, the adversary must be able to perform *after-the-fact-removal*, i.e., corrupt

any honest party p in round r, after observing the messages that p intends to send in round r, and remove any of these messages. In addition, note that that the success probability of any randomized protocol must be calculated over the randomness of both the honest parties and the adversary. The same holds for the expectation of the number of messages needed to be sent by the honest parties.

Theorem 14. *Let π be a randomized byzantine Broadcast protocol (according to Definition 19) that tolerates up to $t \leq n-2$ corruptions and succeeds with probability $3/4+\epsilon$, then in any execution of π the honest parties collectively need to send at least $(\epsilon t)^2$ messages in expectation.*

Proof. Assume for contradiction that there exists a randomized byzantine Broadcast protocol π that tolerates up to t corruptions and succeeds with probability $3/4+\epsilon$ using less than $(\epsilon t)^2$ messages in expectation. We will show that there exists an execution with at most t corruptions, such that π fails to satisfy all properties of Broadcast with probability greater than $1/4-\epsilon$.

Without loss of generality, we assume that there exists a set $Q \subseteq \mathcal{P}\backslash\{s\}$ with at least $(n-1)/2$ parties that output 0 with probability at most $1/2$ if they receive no messages (i.e., with probability more than $1/2$ they either output 1 or not terminate at all). Let $V \subseteq Q$ be a subset of Q with $t/2$ parties and let U be the set of remaining parties, i.e., $U = \mathcal{P}\backslash V$, then we can consider the execution α of π that proceeds as follows :

- every party in U is honest and the sender has an input value 0.
- all parties in V are corrupted and not send any messages to each other.
- every party in V behaves like an honest party towards the parties in U except that it *ignores* the first $t/2$ messages it receives from parties in U.

Let Z be a random variable denoting the number of messages that the honest nodes in U send to parties in V during an execution of π, then we have assumed that $\mathbb{E}[Z] < (\epsilon t)^2$. Let X_1 be the event that the parties in U send at most $\frac{\epsilon}{2}t^2$ messages to the parties in V, i.e., $X_1 = (Z \leq \frac{\epsilon}{2}t^2)$. Using the Markov inequality, we have that $\Pr[Z > \frac{1}{2\epsilon}\mathbb{E}[Z]] < 2\epsilon$ and thus, $\Pr[X_1] = \Pr[Z \leq \frac{\epsilon}{2}t^2] \geq \Pr[Z \leq \frac{1}{2\epsilon}\mathbb{E}[Z]] > 1-2\epsilon$.

Let X_2 be the event that a party p picked uniformly at random from V receives at most $t/2$ among the first $\frac{\epsilon}{2}t^2$ messages that the parties in U send to the parties in V. Note that for these first $\frac{\epsilon}{2}t^2 = 2\epsilon|V|t^2$ messages, there exist at most $2\epsilon|V|$ parties in V that receive more than $t/2$ of them and thus, $\Pr[X_2] \geq 1-2\epsilon$. Hence, in execution α the probability that both the parties in U send at most $\frac{\epsilon}{2}t^2$ messages to the parties in V and that a random party $p \in V$ receives at most $t/2$ of them is :

$$\Pr[X_1 \cap X_2] = \Pr[X_1] + \Pr[X_2] - \Pr[X_1 \cup X_2] > (1-2\epsilon) + (1-2\epsilon) - 1 = 1-4\epsilon$$

Now consider another execution α' of π that is identical to α except that :

- a party p picked uniformly at random from V is honest and all parties in $V\backslash\{p\}$ *ignore* any messages they receive from p.
- the first $t/2$ messages that the parties in U send to p are not received by p, i.e., whenever the adversary observes one of the first $t/2$ attempts of some party $q \in U$ to send a message to p, then it immediately corrupts q (if it is not corrupted already) and only removes this particular

message that is destined for p (q behaves honestly towards the parties in U throughout the execution and starts behaving honestly to p as well, once the first $t/2$ first messages from U to p are dropped).

Note that in α' the messages that honest parties in U receive from p are drawn from the same distribution as in α (which is indicated by π), since in both executions p ignores the first $t/2$ messages destined for it by the parties in U, either intentionally or because they were removed by the adversary. Accordingly, the honest parties in U cannot either distinguish between the distribution of the messages they get from $V\backslash\{p\}$ in the executions α and α', since in both executions the parties in $V\backslash\{p\}$ ignore any messages they get from p. Similarly, in α' honest parties in U receive messages from corrupted parties in U drawn from the same distribution as in α and thus, in the end, they cannot distinguish the behavior of the adversary between the executions α and α'.

In α the designated sender holds an input value 0 and remains honest throughout the execution. So, the event that all honest parties output 0 in α occurs with probability at least $3/4 + \epsilon$, since π must satisfy the validity property of Broadcast with probability at least as much as its overall success probability. However, as we have shown above, the honest parties in U cannot distinguish the adversarial strategies between α and α' and thus, for the event that all honest parties in U output 0 in α', let it be Y_1, it also holds that $\Pr[Y_1] \geq 3/4 + \epsilon$.

In addition, under the condition that the event $X_1 \cap X_2$ occurs in α, p receives no messages at all in α' and recall that V was defined as the set of parties that output 0 with probability at most $1/2$ if they receive no messages. Hence, let Y_2 be the event that p does not output 0 in α', then we can argue that $\Pr[Y_2] \geq \Pr[Y_2|X_1 \cap X_2] \cdot \Pr[X_1 \cap X_2] > \frac{1}{2}(1 - 4\epsilon)$.

Now, it suffices to observe that when the event $Y_1 \cap Y_2$ occurs, then either agreement or termination is violated in α', i.e., either all honest parties in U output 1 or at least one of them does not terminate, while in any case p outputs 0. Therefore, we can argue that for the error probability δ of the protocol π in α', it holds that :

$$\delta \geq \Pr[Y_1 \cap Y_2] = \Pr[Y_1] + \Pr[Y_2] - \Pr[Y_1 \cup Y_2] > \left(\frac{3}{4} + \epsilon\right) + \frac{1}{2}(1 - 4\epsilon) - 1 = \frac{1}{4} - \epsilon$$

which contradicts our initial hypothesis that the protocol π succeeds with probability $3/4 + \epsilon$ in any execution with at most t corruptions. □

Note that for the success probability of the protocol to be overwhelming we can set $\epsilon = 1/4 - r(\kappa)$, for some negligible function $r(\cdot)$.

Corollary 31. *Let π be a byzantine Broadcast protocol (according to Definition 18) that tolerates up to $t \leq n - 2$ corruptions, then in any execution of π the honest parties collectively need to send at least $(1/4 - \mathsf{negl}(\kappa))t^2$ messages in expectation.*

Surprisingly, it turns out that that any byzantine Broadcast protocol tolerating up to $t \leq n - 2$ corruptions, must have an expected (roughly) $t^2/4$ message complexity, which equals the lower bound on the number of messages the honest parities have to send in any deterministic byzantine Broadcast protocol.

4.1.3 Computational Complexity

For the deterministic versions of Broadcast and consensus, the first solutions that came out required the transmission of an exponential amount of information in the number of parties ([2, 33]). However, as shown in [2], every byzantine Broadcast protocol that tolerates up to $t \geq n/3$ parties essentially needs a message authentication mechanism. So, let $\kappa \in \mathbb{N}$ be the security parameter of any signature scheme used to solve byzantine Broadcast, then the time given to the parties, and thus, to the adversary as well, must be polynomial in κ; otherwise, the adversary has a non-negligible probability of "breaking" the security of the signature scheme by forging a signature.

For this reason, all byzantine Broadcast protocols tolerating a dishonest majority require a computational complexity that is polynomial in the security parameter of a PKI. Throughout the rest of this work, we will consider the security parameter to be polynomial in the number of parties, so then it suffices that the protocol requires polynomial computational complexity in the number of parties.

Moreover, note that the computational complexity of all protocols that will be presented later is bounded from below by the bit complexity of a single round, since honest parties should be able to read the whole messages they receive in every round (that is because of the possibility that in some execution there are actually no corruptions and due to the assumption that whenever some honest party sends a message, then the party the message is destined for receives it by the next round). Therefore, a common practice for proving implicitly that the computational complexity of a protocol is polynomially bounded is to show that its bit complexity is polynomial and that the honest parties only have to perform simple operations with the messages they receive.

4.2 The Dolev-Strong Protocol

The famous Dolev-Strong protocol [4] solves Byzantine Broadcast against any adaptive PPT t-adversary when $t < n$. While the original protocol requires signing values recursively, we present a different but yet equivalent form of the protocol, similar to the one of [34]. In particular, let $b_s \in \{0, 1\}$ be the input value of the designated sender $s \in \mathcal{P}$, then the protocol works as follows :

- In round $r = 0$, every party $p \in \mathcal{P}$, initializes the set of its accepted values A_p to be empty. The designated sender s signs its input b_s and multicasts it, i.e., sends it to every party.
- In every round $r = 1, \ldots, t$, if some party $p \in \mathcal{P}$ receives a batch of signatures on some value $b \in \{0, 1\}$ from r distinct parties including s and $b \notin A_p$, then p adds b to A_p. Then, it signs b and multicasts all the signatures it has observed on b (including its own).
- In round $t + 1$, if some party $p \in \mathcal{P}$ receives a batch of signatures on some value $b \in \{0, 1\}$ from $t + 1$ distinct parties including s and $b \notin A_p$, then p adds b to A_p. Every party $p \in \mathcal{P}$ outputs 1 if $A_p = \{1\}$, else it outputs 0.

Note that the protocol is based on a *quorum certificate*, i.e., r signatures (*votes*) on some value b are sufficient to convince any party to accept b in round r. Moreover, this certificate is *transferable* in the sense that whenever some party p receives such a certificate for some value b for the first time in round r, then p can easily make use of it to produce a new *certificate* for round $r + 1$, by adding its own signature. Hence, once p accepts b in round $r \leq t$, it can easily convince the rest

of the honest parties to accept b by round $r+1$ (*relay property*). Now if p accepts some value b in round $t+1$, then it must have received $t+1$ signatures on b and, since there are at most t corrupted parties in $\mathcal{P}$, at least one of these signatures must have been generated by some honest party at some round prior to $t+1$. Thus, every honest party must have received a certificate for b by round $t+1$, which retains the protocol's consistency. These intuitive arguments will be formally proved in section 4.2.2.

4.2.1 Formal Description of Dolev-Strong protocol

We will now give a formal description of the Dolev-Strong protocol. We first introduce the notion of *valid* batches of signatures and *well-formed* messages.

Valid batches of signatures. We will call B a r-*valid* batch of signatures on some value $b \in \{0,1\}$, if it consists of signatures on b from at least r distinct parties including s. Formally, let $B = \{\mathsf{Sign}_v(b)\}_{v\in V}$, for some $V \subseteq \mathcal{P}$, then B is a r-*valid* batch of signatures on b if $|V| \geq r$ and $s \in V$.

Well-formed messages. In the DS protocol, the honest parties have to multicast batches of signatures in support of their newly accepted values. Hence, in order to prevent the adversary from sending arbitrarily many batches of signatures on the same value, every party shall multicast at most one batch of signatures for each newly accepted value. So, let M_p^r be the message that some party $p \in \mathcal{P}$ multicasts at the end of round r, then M_p^r will be called *well-formed* if $M_p^r = \{B\}$, where B is a batch of signatures on some value $b \in \{0,1\}$, i.e., $B = \{\mathsf{Sign}_v(b)\}_{v\in V}$, for some $V \subseteq \mathcal{P}$ or $M_p^r = \{B_0, B_1\}$, where B_b are batches of signatures in support of the values 0, 1, respectively, such that $B_b = \{\mathsf{Sign}_v(b)\}_{v\in V_b}$, for some $V_b \subseteq \mathcal{P}$. In the following form of the DS protocol honest parties will only process *well-formed* messages and discard the rest.

We give a formal description of the Dolev-Strong protocol in Figure 4.2.1.

For every party $p \in \mathcal{P}$, we denote with A_p^r the set of values that p has accepted by round r. Note that every party $p \in \mathcal{P}$ stores for every newly accepted value $b \in A_p^r \backslash A_p^{r-1}$ all the signatures it has observed in support of b in $C_{p,b}$. Also, in every round $r \in [1,t]$, every party $p \in \mathcal{P}$ forms a new batch of signatures in support of each $b \in A_p^r \backslash A_p^{r-1}$, by adding to $C_{p,b}$ its own signature on b, and multicasts these new batches of signatures at the end of round r.

4.2.2 Security Proofs

In this section, we will formally prove that the Dolev-Strong protocol satisfies the properties of the byzantine Broadcast (Definition 18) when $t < n$.

Lemma 32 (Validity)**.** *If the sender s is honest, then every honest party outputs b_s.*

Proof. If s is honest, then it multicasts $M_s^0 = \{\{\mathsf{Sign}_s(b_s)\}\}$ at the end of round 0 and thus, every honest party $p \in \mathcal{P}$ receives it in round 1. Since $\{\mathsf{Sign}_s(b_s)\}$ is a 1-valid batch of signatures for b_s, p accepts b_s in round 1. Now assume for the contrary that some honest party p accepts some value

Protocol π_{ds}

Round 0: Every party $p \in \mathcal{P}$ initializes $A_p^0 = C_{p,0} = C_{p,1} = \emptyset$. The designated sender s multicasts $M_s^0 = \{\{\mathsf{Sign}_s(b_s)\}\}$.

Loop: For $r \in [1,t]$, every party $p \in \mathcal{P}$ does the following :

1. Initializes $A_p^r = A_p^{r-1}$.
2. Upon receiving some well-formed message M, for all $B \in M$, do the following : if B is a r-valid batch of signatures for some value $b \notin A_p^{r-1}$, then set $A_p^r \leftarrow A_p^r \cup \{b\}$ and $C_{p,b} \leftarrow C_{p,b} \cup B$.
3. Multicasts $M_p^r = \{C_{p,b} \cup \{\mathsf{Sign}_p(b)\}\}_{b \in A_p^r \setminus A_p^{r-1}}$.

Round $\mathbf{t+1}$: Every party $p \in \mathcal{P}$ does the following :

1. Initializes $A_p^{t+1} = A_p^t$.
2. Upon receiving some well-formed message M, for all $B \in M$, do the following : if B is a $(t+1)$-valid batch of signatures for some value $b \notin A_p^t$, then set $A_p^{t+1} \leftarrow A_p^{t+1} \cup \{b\}$.
3. Terminates and outputs 1, if $A_p^{t+1} = \{1\}$, else outputs 0.

Figure 4.2.1: Dolev-Strong Protocol

b with $b \neq b_s$ at some round $r \in [1, t+1]$, then p must have received a r-valid batch of signatures for b in round r, let it be B, such that $\mathsf{Sign}_s(b) \in B$. Hence, with overwhelming probability s has signed b at some round prior to r, which is a contradiction, since s is honest. Therefore, every honest party p observes $A_p^r = \{b_s\}$ in every round $r \in [1, t+1]$, and thus terminates with output b_s in round $t+1$. □

Lemma 33 (Relay)**.** *If some honest party $p \in \mathcal{P}$ accepts some value $b \in \{0,1\}$ in round $r \in [1,t]$, then every honest party q accepts b by round $r+1$.*

Proof. If p accepts b at round $r \in [1,t]$, then p receives a r-valid batch of signatures on b in round r, let it be $B = \{\mathsf{Sign}_v(b)\}_{v \in V}$, for some $V \subseteq \mathcal{P}$, and observes $b \notin A_p^{r-1}$, $s \in V$ and $|V| \geq r$. We observe that with overwhelming probability $p \notin V$, since $p \in V$ would imply that p's signature on b was forged at some round prior to r. Since p is honest, it multicasts M_p^r at the end of round r, which is well-formed by construction and contains some batch of signatures on b, let it be $B' = \{\mathsf{Sign}_v(b)\}_{v \in V'}$, with $V \cup \{p\} \subseteq V' \subseteq \mathcal{P}$. Therefore, even if q has not accepted b by round r, then q receives B' in round $r+1$, which is $(r+1)$-valid, since $s \in V'$ and $|V'| \geq |V \cup \{p\}| \geq r+1$, and observes $b \notin A_q^r$. Thus, q accepts b in round $r+1$. □

Lemma 34 (Agreement)**.** *All honest parties output the same value.*

Proof. We will show that $A_p^{t+1} = A_q^{t+1}$ for any two honest parties $p, q \in \mathcal{P}$. So, let $b \in A_p^{t+1}$ be some value that p accepts by round $t+1$, then we have to consider the two following cases : i) if p accepts b in round $r \in [1,t]$, then using Lemma 33, we have that every honest party q accepts b by round $r+1$, and ii) if p accepts b at round $t+1$, then p receives a $(t+1)$-valid batch of signatures on b in round $t+1$, let it be $B = \{\mathsf{Sign}_v(b)\}_{v \in V}$, for some $V \subseteq \mathcal{P}$ with $|V| \geq t+1$. Since

there are at most t corrupted parties in $\mathcal{P}$, there must be at least one honest party in V. So, with overwhelming probability, this honest party has accepted b at some round prior to $t+1$, and thus, using Lemma 33, we have that every honest party q accepts b by round $t+1$. Therefore, in any case, we have that $b \in A_q^{t+1}$, which completes the proof. □

Complexity Measures. Obviously, the protocols requires $t+1$ rounds. In the worst case, each party has send to every party two batches of $O(n)$ signatures. In that case, the honest parties have to send in total $O(n^2)$ messages that consist of $O(n^3)|\sigma|$ bits, where $|\sigma|$ is the signature length of the PKI.

Theorem 15. *The Dolev-Strong protocol π_{ds} (Figure 4.2.1) is a byzantine Broadcast protocol (according to Definition 18) that is secure against any adaptive PPT t-adversary, when $t < n$. Moreover, the protocol requires $t+1$ rounds and has a communication complexity of $O(n^2)$ messages with $O(n^3)|\sigma|$ bits, where $|\sigma|$ it the signature length of the PKI.*

Remark 35. *Note that the protocol is secure as long as all honest parties run for R rounds, where R can be any number greater than t. So, even if the honest parties have no information about the upper bound on the number of the byzantine corruptions t, then they can still run Dolev-Strong for $n-1$ rounds retaining the protocol's consistency. Hence, the protocol is optimal in terms of its worst-case round complexity as a protocol that tolerates any number of corruptions, i.e., $t = n - O(1)$, in at most $n-1$ rounds (Corollary 29).*

Remark 36. *The message complexity of the Dolev-Strong protocol is $O(n^2)$, which differs from the $\Omega(t^2)$ lower bound of Theorem 14 only by a constant factor when $t = \Omega(n)$. Therefore, the Dolev-Strong protocol can be considered as optimal with respect to the order of its message complexity.*

4.3 Sublinear-Round Byzantine Broadcast Protocol

In the presence of an honest majority, a line of works in the literature ([35, 36, 37]) has shown that there are various ways to solve Byzantine Broadcast with an expected constant round complexity. In the case of corrupt majority, Garay *et al.* [8] showed that there exists a protocol that tolerates up to $t = n/2 + k$ corruptions and terminates in $O(k^2)$ rounds in expectation, which was later improved by Fitzi and and Nielsen [25] to $O(k)$ rounds. However, the lower bound of Garay *et al.* [8] that we presented in subsection 4.1.1 implies that there exists no (even randomized) byzantine Broadcast protocol, tolerating any number of corruptions, that terminates in $O(1)$ rounds in expectation. So, the main question (which remains up to this day) is whether we can construct a randomized byzantine Broadcast protocol with sublinear round complexity that tolerates any number of corruptions.

Chan *et al.* [11] made a significant step towards this direction by constructing a randomized byzantine Broadcast protocol tolerating $t = (1-\epsilon)n$ corruptions that terminates in $O(\log(1/\delta)/\epsilon)$ rounds and errs with probability at most $\delta + \mathsf{negl}(\kappa)$, for any $\delta > 0$. Note that given the $\Omega(1/\epsilon)$ lower bound on the round complexity (Corollary 30), their result is optimal up to a $\log(1/\delta)$ factor, which, nonetheless, must be superlogarithmic in κ if we want the protocol to have an overwhelming success probability, i.e., $\delta = \mathsf{negl}(\kappa)$. Moreover, notice that the protocol retains its sublinear round complexity as long as ϵ is a constant number or a large enough subconstant function of n (e.g. $\epsilon = \Theta(1/\sqrt{n})$), but becomes linear when $\epsilon = O(1/n)$, i.e., $t = n - O(1)$.

We will now present their idea in two simple steps : first, we will describe a byzantine Broadcast protocol with the desired properties that is resilient to $t = (1 - \epsilon)n$ static corruptions and then, we will show how to make our protocol achieve adaptive security using an ideal functionality called $\mathcal{F}_{\mathsf{mine}}$.

4.3.1 Sublinear-Round Protocol for Static Corruptions

The idea follows the *committee* election paradigm that is a quite common practice for reducing the round complexity of byzantine Broadcast protocols, but, nonetheless, cannot be directly applied in the corrupt majority setting. In fact, assume that we have a $O(n)$-round byzantine Broadcast protocol π in the honest majority setting, then we easily turn it into a $O(\mathsf{polylog}(n))$-round protocol as follows : we can randomly select a subset of the parties with polylogarithmic size in n, called the *committee*, and run π among the parties of the committee. Using a simple probability argument, we can show that the committee also has an honest majority and thus, all honest committee members will agree on the same output of π. Hence, we can use an extra round for the committee members to vote on their output (by multicasting it) and thus, all honest parties of the initial party set can now decide on a common value by taking the majority of the votes.

On the contrary, in the corrupt majority setting, agreement on a common value cannot be relied solely on the votes from the committee members (non-committee honest parties cannot distinguish between honest and corrupt votes). Instead, to construct a sublinear-round byzantine Broadcast protocol resilient to corrupt majority, one has to combine the committee approach with the Dolev-Strong idea that we presented in section 4.2. In particular, we can randomly select a committee of size $\log(1/\delta)$ (after the adversary chooses the corrupted parties), so then with probability $O(\delta)$ there exists at least one honest committee member (this will be formally proved in subsection 4.3.4). The idea is that only the committee members (along with the designated sender) will be responsible for signing values and that all signatures from non-committee members will be ignored. Thus, in order to retain the transferability property of the valid batches of signatures that we had in Dolev-Strong we have to split each round of Dolev-Strong into two mini-rounds : in the first one, whenever some party observes a certificate for some value it will just forward it to all parties, while in the second, the committee members will first have to add their signature in the certificate and then multicast it. Therefore, let $b_s \in \{0, 1\}$ be the input value of the designated sender $s \in \mathcal{P}$, then the protocol works as follows :

- In round $r = 0$, every party $p \in \mathcal{P}$, initializes the set of its accepted values A_p to be empty. The designated sender s signs its input b_s and multicasts it.
- In every round $r = 1, \ldots, \log(1/\delta)$:
  1. in the first min-round, if some party $p \in \mathcal{P}$ receives a batch of signatures on some value $b \in \{0, 1\}$ from r distinct valid signers including s and $b \notin A_p$, then p adds b to A_p. Then, it multicasts all the signatures it has observed on b.
  2. in the second min-round, if some committee member $p \in \mathcal{P}$ receives a batch of signatures on some value $b \in \{0, 1\}$ from r distinct valid signers including s and $b \notin A_p$, then p adds b to A_p. Then, it signs b and multicasts all the signatures it has observed on b (including its own).

– In round $\log(1/\delta)+1$, if some party $p \in \mathcal{P}$ receives a batch of signatures on some value $b \in \{0,1\}$ from $\log(1/\delta)+1$ distinct valid signers including s and $b \notin A_p$, then p adds b to A_p. Every party $p \in \mathcal{P}$ outputs 1 if $A_p = \{1\}$, else it outputs 0.

Note that the only *valid* signers are the members of the committee and the designated sender s. Hence, the above protocol is based on a *quorum certificate* very similar to Dolev-Strong, since r signatures on some value b from valid signers are sufficient to convince any party to accept b in round r. Moreover, although transferability is not always guaranteed, we can argue that whenever some party p receives a certificate for some value b for the first time in the first mini-round of round r, then, as long as there exists an honest committee member q, then q will be able to produce a new *certificate* for round $r+1$ in the second mini-round of round r, by adding its own signature. Thus, once p accepts b in the first mini-round of round $r \leq t$, then it suffices that there exists an honest party in the committee, so then every honest party accepts b by the first mini-round of round $r+1$. Now if p accepts some value b in round $\log(1/\delta)+1$, then it must have received $\log(1/\delta)$ signatures on b from committee members (along with the signature of the sender) and thus, for b to not be accepted by all honest parties by round $\log(1/\delta)+1$, it must be the case that all committee members are corrupted. Note that in any case, the protocol is heavily relied on the estimation that the committee will always have an honest member, which will be formally proved to hold with $1-\delta$ probability in subsection 4.3.4.

4.3.2 Adaptive Security in the $\mathcal{F}_{\mathsf{mine}}$-Hybrid World

Although the protocol we just described has indeed a sublinear round complexity and $1-\delta$ success probability, we can easily observe that it blatantly fails against an adaptive adversary. In fact, once the committee members are chosen, the adversary can just corrupt all members in the committee and thus, block the transferability of the protocol. In order to overcome this issue, the committee will no longer have to be determined a priori; on the contrary, it will be elected on the fly using an ideal functionality called $\mathcal{F}_{\mathsf{mine}}$ (inspired by the blockchain mining).

The main idea of the functionality is that whenever some honest party p accepts some value b, then p will have to ask the functionality to see whether it has the privilege to sign (*mine*) b, i.e., if p is in the b-committee. In addition, when some honest party p receives a batch of signatures for some value b, it accepts b only if all these signatures are generated by parties *verified* by the functionality to belong in the b-committee.

Note that, in comparison with the previous approach, there is not a common committee elected for both values, but instead each value $b \in \{0,1\}$ has its own committee. As a result, when some party p is elected in the b-committee, then the adversary gains no advantage in voting for $1-b$ by corrupting p than by corrupting any other party in $\mathcal{P}$. This bit-specific committee approach was inspired by [10] and [38].

Note that the probability p is considered to be a public parameter and must be such that the size of each b-committee is $O(\log(1/\delta))$. In [11], they show how to realize the $\mathcal{F}_{\mathsf{mine}}$ functionality in the real world using adaptively secure cryptographic tools, which are out of scope of this work, and hence, are omitted.

Functionality $\mathcal{F}_{\mathsf{mine}}$

Mine: If some party $p \in \mathcal{P}$ calls $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$ for the first time for some $b \in \{0,1\}$, then $\mathcal{F}_{\mathsf{mine}}$ flips a random coin (parameterized with an appropriate probability p) which determines if p is in the b-committee. Moreover, $\mathcal{F}_{\mathsf{mine}}$ responds to p accordingly and stores the returned value, so whenever p calls $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$ again, $\mathcal{F}_{\mathsf{mine}}$ gives the same answer.

Verification: If some party $p \in \mathcal{P}$ calls $\mathcal{F}_{\mathsf{mine}}.\mathsf{verify}(b,q)$ for some pair $(b,q) \in \{0,1\} \times \mathcal{P}$ to check whether q is in the b-committee, then $\mathcal{F}_{\mathsf{mine}}$ responds positively only if q has already called $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$ successfully, otherwise it responds negatively.

Figure 4.3.1: The $\mathcal{F}_{\mathsf{mine}}$ Functionality

4.3.3 Formal Protocol in the $\mathcal{F}_{\mathsf{mine}}$-Hybrid World

We will now give a formal description of the sublinear-round byzantine Broadcast protocol, tolerating $t = (1-\epsilon)n$ corruptions, that terminates in $O(\log(1/\delta)/\epsilon)$ rounds and has $1 - \delta - \mathsf{negl}(\kappa)$ success probability. We first introduce the notion of *valid* batches of signatures in the $\mathcal{F}_{\mathsf{mine}}$-Hybrid World. Well-formed messages are already defined in subsection 4.2.1.

Valid batches of signatures. We will call B a *r-valid* batch of signatures on some value $b \in \{0,1\}$, if it consists of signatures on b from at least r distinct parties in the b-committee including s. Formally, let $B = \{\mathsf{Sign}_v(b)\}_{v \in V}$, for some $V \subseteq \mathcal{P}$, then B is a *r-valid* batch of signatures on b if $|V| \geq r$, $s \in V$ and $\mathcal{F}_{\mathsf{mine}}.\mathsf{verify}(b,v)$ is successful for at least $r-1$ parties $\{v_i\}_{i=1}^{r-1} \subseteq V \backslash \{s\}$.

We give a formal description of the sublinear-round protocol in Figure 4.3.2.

Note that the protocol proceeds in stages, where each stage consists of two rounds (except from stages 0 and $R+1$ which consist of a single round). For every party $p \in \mathcal{P}$, we denote with $A_p^r, \hat{A}_p^r$ the set of values that p has accepted by the first and second round of stage r, respectively. The only point where the protocol deviates from Dolev-Strong is that if some b-committee member accepts b in the first round of stage r, then it accepts it again in the second round of stage r, so then it constructs a valid $(r+1)$-valid batch of signatures on b which will make all honest parties in $\mathcal{P}$ accept b by round $r+1$. Moreover, note that non-committee members take no action in the second round of any stage. The reason for setting these specific values to the parameters R and p will be made clear by the proof of the above protocol's correctness.

4.3.4 Proof of Correctness

We will now formally prove that the protocol of Figure 4.3.2 satisfies the properties of Definition 19 when $t = (1-\epsilon)n$, with up to a $\mathsf{negl}(\kappa)$ difference in the success probability. Note that the protocol π_{sb} requires $2R+1 = 2\lceil \frac{3}{\epsilon} \cdot \ln \frac{1}{\delta} \rceil + 1$ rounds and thus, it suffices to prove that π_{sb} satisfies validity and agreement with probability $1 - \delta - \mathsf{negl}(\kappa)$, for any choice of $\delta > 2e^{-\epsilon n}$. Note that we restrict our attention in the case that $\delta > 2e^{-\epsilon n}$, since for $\delta \leq 2e^{-\epsilon n} \leq 2e^{-\epsilon n/6}$, we have that π_{sb} has an $\Omega(n)$ round complexity.

Protocol π_{sb}

Let $R = \lceil \frac{3}{\epsilon} \cdot \ln \frac{2}{\delta} \rceil$ and that $\mathcal{F}_{\mathsf{mine}}$ is instantiated with $p = \frac{1}{\epsilon n} \ln \frac{2}{\delta}$.

Stage 0: Every party $p \in \mathcal{P}$ initializes $\hat{A}_p^0 = C_{p,0} = C_{p,1} = \emptyset$. The designated sender s multicasts $M_s^0 = \{\{\mathsf{Sign}_s(b_s)\}\}$.

Loop: Every stage $r \in [1, R]$ consists of 2 rounds.

1. In the first round, every party $p \in \mathcal{P}$ does the following :
   - Initializes $A_p^r = \hat{A}_p^{r-1}$.
   - Upon receiving some well-formed message M, for all $B \in M$, do the following : if B is a r-valid batch of signatures for some value $b \notin \hat{A}_p^{r-1}$, then set $A_p^r \leftarrow A_p^r \cup \{b\}$ and $C_{p,b} \leftarrow C_{p,b} \cup B$.
   - Multicasts $M_p^r = \{C_{p,b}\}_{b \in A_p^r \setminus \hat{A}_p^{r-1}}$.
2. In the second round, every party $p \in \mathcal{P}$ does the following :
   - Initializes $\hat{A}_p^r = A_p^r$.
   - Upon receiving some well-formed message M, for all $B \in M$, do the following : if B is a r-valid batch of signatures for some value $b \notin \hat{A}_p^{r-1}$ and $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$ is successful, then set $\hat{A}_p^r \leftarrow \hat{A}_p^r \cup \{b\}$ and $C_{p,b} \leftarrow C_{p,b} \cup B$.
   - If $\hat{A}_p^r \neq A_p^r$, then it multicasts $\hat{M}_p^r = \{C_{p,b} \cup \{\mathsf{Sign}_p(b)\}\}_{b \in \hat{A}_p^r \setminus \hat{A}_p^{r-1}}$.

Stage $\mathbf{R+1}$: Every party $p \in \mathcal{P}$ does the following :

1. Initializes $A_p^{R+1} = \hat{A}_p^R$.
2. Upon receiving some well-formed message M, for all $B \in M$, do the following : if B is a $(R+1)$-valid batch of signatures for some value $b \notin \hat{A}_p^R$, then set $A_p^{R+1} \leftarrow A_p^{R+1} \cup \{b\}$.
3. Terminates and outputs 1, if $A_p^{R+1} = \{1\}$, else outputs 0.

Figure 4.3.2: Sublinear-Round Byzantine Broadcast Protocol

Lemma 37 (Validity)**.** *If the sender s is honest, then every honest party outputs b_s.*

Proof. If s is honest, then it multicasts $M_s^0 = \{\{\mathsf{Sign}_s(b_s)\}\}$ at the end of stage 0 and thus, every honest party $p \in \mathcal{P}$ receives it in the first round of stage 1. Since $\{\mathsf{Sign}_s(b_s)\}$ is a 1-valid batch of signatures for b_s, p accepts b_s in stage 1. Now assume for the contrary that some honest party p accepts some value b with $b \neq b_s$ at some round $w \in \{1, 2\}$ of some stage $r \in [1, R+1]$, then p must have received a r-valid batch of signatures for b in round w of stage r, let it be B, such that $\mathsf{Sign}_s(b) \in B$. Hence, with overwhelming probability s has signed b at some round prior to round w of stage r, which is a contradiction, since s is honest. Therefore, every honest party p observes $A_p^r = \{b_s\}$ in every stage $r \in [1, R+1]$, and thus terminates with output b_s in stage $R+1$. □

Hence, validity is satisfied with overwhelming probability. We will know prove that π_{sb} achieves agreement with probability $1 - \delta - \mathsf{negl}(\kappa)$. Note that for agreement to be violated at least one event of the following types must occur at lest once throughout the execution :

- Type A : some value b is accepted by some honest party $p \in \mathcal{P}$ at some stage $r \in [1, R]$, but is

not accepted by some party $q \in \mathcal{P}$ by the end of the protocol.

– Type B : some value b is accepted by some honest party $p \in \mathcal{P}$ in stage $R+1$, but is not accepted by some party $q \in \mathcal{P}$ by the end of the protocol.

Claim 38 (Agreement)**.** *If no events of Type A or B occur throughout an execution of the protocol π_{sb}, then π_{sb} satisfies the agreement property of byzantine Broadcast.*

We will show that that any event of Type A or B occurs with probability at most $\delta/2 + \mathsf{negl}(\kappa)$. First, we will prove that all honest b-committee members are able to transfer a certificate for b. The proof is almost identical to one of Lemma 33, but is presented for completeness.

Lemma 39 (Relay)**.** *If some honest party $p \in \mathcal{P}$ in the b-committee accepts b in the second round of some stage $r \in [1, R]$, then every honest party accepts b in the first round of stage $r+1$.*

Proof. If p accepts b in the second round of stage $r \in [1, t]$, then p receives a r-valid batch of signatures on b in round r, let it be $B = \{\mathsf{Sign}_v(b)\}_{v \in V}$, for some $V \subseteq \mathcal{P}$, and observes $b \notin \hat{A}_p^{r-1}$, $s \in V$, $|V| \geq r$ and the calls $\mathcal{F}_{\mathsf{mine}}.\mathsf{verify}(b, v)$ being successful for at least $r-1$ parties $\{v_i\}_{i=1}^{r-1} \subseteq V$. We observe that with overwhelming probability $p \notin V$, since $p \in V$ would imply that p's signature on b was forged at some round prior to r. Since p is honest and $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$ is successful, it multicasts $\hat{M}_p^r$ at the end of stage r, which is well-formed by construction and contains some batch of signatures on b, let it be $B' = \{\mathsf{Sign}_v(b)\}_{v \in V'}$, with $V \cup \{p\} \subseteq V' \subseteq \mathcal{P}$. Therefore, even if q has not accepted b by stage r, then q receives B' in the first round of stage $r+1$, which is $(r+1)$-valid, since $s \in V'$, $|V'| \geq |V \cup \{p\}| \geq r+1$ and the call $\mathcal{F}_{\mathsf{mine}}.\mathsf{verify}(b, v)$ is successful at least r parties $\{v_i\}_{i=1}^{r-1} \cup \{p\} \subseteq V' \backslash \{s\}$, and observes $b \notin \hat{A}_q^r$. Thus, q accepts b in the first round of stage $r+1$. □

Lemma 40. *For any value $b \in \{0, 1\}$, an event of Type A for b occurs with probability at most $\delta/2 + \mathsf{negl}(\kappa)$.*

Proof. We begin with observing that if some honest party $p \in P$ accepts b in the second round of some stage $r \in [1, R]$, then p must be in the b-committee and thus, using Lemma 39, every honest party accepts b by stage $r+1$ with all but negligible probability. So, the only case where an event of Type A may occur with non-negligible probability is that p accepts b in the first round of stage r. In that case, in the second round of stage r every honest party receives a r-valid batch of signatures on b by p. So, assume that there exists at least one honest party in the b-committee, then using Lemma 39, every honest party accepts b by stage $r+1$. Thus, an event of Type A occurs only when none of the honest parties is elected in the b-committee, i.e., all $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$ calls from honest parties turn out to be unsuccessful. Since, $\mathcal{F}_{\mathsf{mine}}$ uses a random coin with probability p to determine whether each one of the ϵn honest parties is in the b-committee, we have that the above event occurs with probability $(1-p)^{\epsilon n} \leq e^{-\epsilon n p} \leq \delta/2$. Therefore, the overall probability of an event of Type A to occur is at most $\delta/2 + \mathsf{negl}(\kappa)$. □

Lemma 41. *For any value $b \in \{0, 1\}$, an event of Type B for b occurs with probability at most $\delta/2 + \mathsf{negl}(\kappa)$.*

Proof. If some honest party $p \in \mathcal{P}$ accepts b in stage $R+1$, then p receives a $(R+1)$-valid batch of signatures on b in stage $R+1$, let it be $B = \{\mathsf{Sign}_v(b)\}_{v \in V}$, for some $V \subseteq \mathcal{P}$ with $|V| \geq R+1$.

Hence, there must be at least R parties $\{v_i\}_{i=1}^{r} \subseteq V \backslash \{s\}$ that have called $\mathcal{F}_{\mathsf{mine}}.\mathsf{mine}(b)$ successfully. Now if at least one of these parties is honest, let it be q, then q must have accepted b by some stage prior to $R+1$ (and in particular, in the second round of that stage) and thus, using Lemma 39, every honest party accepts b by stage $R+1$ with all but negligible probability. So, the only case when an event of Type B occurs is that there are at least R corrupted parties in the b-committee. However, let X be a random variable denoting the number of corrupted parties in the b committee, then X follows a binomial distribution with parameters $(1-\epsilon)n$ and p, and thus, has an expected value $\mathbb{E}[X] = (1-\epsilon)np$. Therefore, the probability $\Pr[X \geq R]$ can be bonded from above in both of the following cases :

- if $\epsilon \geq 1/4$, let $\tau = \frac{3\epsilon}{1-\epsilon} \geq 1$, then we have that $(1+\tau)(1-\epsilon)np = (1+2\epsilon)n\frac{1}{\epsilon n}\ln\frac{2}{\delta} \leq \frac{3}{\epsilon}\ln\frac{2}{\delta} \leq R$. Hence, using the Chernoff Bound of Corollary 24, we have that $\Pr[X \geq R] \leq \Pr[X \geq (1+\tau)(1-\epsilon)np] \leq \exp(-\frac{\tau(1-\epsilon)np}{3}) = \delta/2$.
- if $\epsilon < 1/4$, let $\tau = 1$, then we have that $(1+\tau)(1-\epsilon)np = 2(1-\epsilon)n\frac{1}{\epsilon n}\ln\frac{2}{\delta} \leq \frac{3}{\epsilon}\ln\frac{2}{\delta} \leq R$. Thus, using the Chernoff Bound of Corollary 24, we have that $\Pr[X \geq R] \leq \Pr[X \geq (1+\tau)(1-\epsilon)np] \leq \exp(-\frac{(1-\epsilon)np}{3}) = \exp(-\frac{1-\epsilon}{3\epsilon}) \cdot \delta/2 \leq \delta/2$, since $\frac{1-\epsilon}{3\epsilon} \geq 1$.

So, in any case we have that $\Pr[X \geq R] \leq \delta/2$ and thus, the overall probability that an event of Type B occurs is at most $\delta/2 + \mathsf{negl}(\kappa)$. □

Theorem 16. *For any $\epsilon, \delta \in (0,1)$, with $\delta > 2e^{-\epsilon n}$, the protocol π_{sb} (Figure 4.3.2) is a randomized byzantine Broadcast protocol (according to Definition 19), tolerating any adaptive PPT t-adversary, when $t < (1-\epsilon)n$, that terminates in $2\lceil\frac{3}{\epsilon}\cdot\ln\frac{2}{\delta}\rceil+1$ rounds and succeeds with probability $1-\delta-\mathsf{negl}(\kappa)$.*

Proof. The protocol obviously terminates in $2R+1 = 2\lceil\frac{3}{\epsilon}\cdot\ln\frac{2}{\delta}\rceil+1$ rounds. Using Lemma 37, we have that π_{sb} satisfies the validity property of Broadcast with overwhelming probability. Finally, we observe that π_{sb} satisfies the agreement property as well, as long as no events of Type A or B occur throughout its execution. Using Lemmas 40 and 41, we have that $\Pr[E_A] \leq \delta/2 + \mathsf{negl}(\kappa)$ and $\Pr[E_B] \leq \delta/2 + \mathsf{negl}(\kappa)$, where E_A, E_B denote the events that at least one event of Type A, B occurs, respectively. Hence, using Claim 38, we have that agreement is violated with probability at most $\Pr[E_A \cup E_B] \leq \Pr[E_A] + \Pr[E_B] \leq \delta + \mathsf{negl}(\kappa)$. Hence, the overall success probability of π_{sb} is $1 - \delta - \mathsf{negl}(\kappa)$, which completes the proof. □

Communication Complexity. Exactly as in Dolev-Strong, each party has send to every party at most two messages throughout the execution of the protocol and thus, π_{sb} has a message complexity of $O(n^2)$ (which is optimal up to constant factor, as shown in Theorem 14). However, in π_{sb} each message contains $O(\mathbb{E}[C])$ signatures in expectation, where C is a random variable denoting the number of parties in any b-committee. Since $\mathbb{E}[C] = np = \ln(2/\delta)/\epsilon$, we have that each one of the $O(n^2)$ messages has $O(\ln(1/\delta)/\epsilon)$ signatures in expectation and thus, the overall bit complexity of π_{sb} is $O(n^2\ln(1/\delta)/\epsilon)|\sigma|$, where $|\sigma|$ is the signature length of the PKI.

Remark 42. *Note that the restriction $\delta > 2e^{-\epsilon n}$ allows δ to be strictly less than $1/2$, as long as $2e^{-\epsilon n} < 1/2$, which is equivalent to $\epsilon > \ln(4)/n$. Hence, for any meaningful application, we could say that π_{sb} achieves the desired properties for any reasonable choice of δ, even as a function of n.*

Chapter 5

Unknown Participants Broadcast for Any Number of Corruptions

5.1 The Unknown Participants Setting

In the *unknown participants setting*, we assume that the parties willing to participate in some execution of a protocol have no a priori knowledge about the rest of the participants, neither have an estimation of their number. For convenience, we can consider a (large) set of parties that may participate in some execution, called the *universe* $\mathcal{U}$, and assume that only some relatively small proportion of them will actually take part in the execution. We will consider that the *active* parties, i.e., the parties that actively participate, form a set denoted by $\mathcal{P}$ and that the parties of $\mathcal{P}$ have no clue about each other at the beginning of the execution.

Since we are mostly interested in the case of corrupted majorities, we will have to assume the existence of a signature scheme. However, in this setting $\mathcal{P}$ is not known in advance, and thus, the security parameter of the signature scheme that would suffice to guarantee the security of the protocol cannot be determined by $|\mathcal{P}|$ as in the classic setting. In fact, we will have to consider a fixed security parameter $\kappa \in \mathbb{N}$ and refer to the cardinalities of $\mathcal{U}$ and $\mathcal{P}$ with respect to κ. In particular, we will consider that the universe contains a number of potential participants that is superpolynomial in κ and that in every execution of a protocol, only a polynomial number of parties actually participate. Notice that the superpolynomial gap between the number of potential and active parties is essential in order to differentiate the unknown participants setting from the classic one. In addition, the necessity of the restriction on the number of active parties to be polynomial in κ is relied on the need to preserve the unforgeability property of the signature scheme. Indeed, if we allowed a superpolynomial number of parties to participate in the execution, then we would have to give superpolynomial time to every party so then it could just read the messages it could receive from the rest of the parties. Hence, the signature scheme would be insecure.

Authentication. In the unknown participants setting, the use of a standard PKI (as defined in section 3.1) would require that the parties have locally stored the public keys of all possible participants in $\mathcal{U}$ and therefore, need memory superpolynomial in κ. Since we want to overcome

this efficiency drawback and make our model even stronger, we assume that the authentication keys of the parties in $\mathcal{U}$ are not predetermined and that all parties willing to participate must generate their secret-public key pair at the beginning of the execution. Then, in order for their signatures to be considered as valid by the rest of the parties, all participants should ask a *certification authority* for a *certificate* for the validity of their public keys. Once they get their certificate, then they will be allowed to produce valid signatures.

Communication. We will assume that parties in $\mathcal{U}$ can communicate with each other via a *diffusion* network, where any message which is sent by some honest party in some round, is delivered to all parties by the beginning of the following round. This can be achieved by means of network assumptions which are standard in the blockchain literature: parties use "gossiping" over an incomplete but connected synchronous network to send their messages, where a protocol round last as much as the upper bound to the time/network-rounds it takes for a message to be delivered (which is the length of the longest path in the underlying communication graph).

For convenience, we will consider that all parties in $\mathcal{U}$ have access to a diffusion functionality $\mathcal{F}_{\mathsf{diffuse}}$, which in practice can be considered as an implementation of "multicast" in the unknown participants setting. We assume that every party has the right to call $\mathcal{F}_{\mathsf{diffuse}}(m)$ only once in every round for any message $m \in \{0,1\}^{\mathsf{poly}(\kappa)}$, and that m will be delivered to all parties in $\mathcal{U}$ by the end of the round.

Adversary. In the unknown participants setting the adversary can choose a subset of the parties of the universe to activate and force them to participate in the execution under its orders. Since we want the total number of active parties to be polynomial in κ, we will assume that in the beginning of the execution there is a polynomially large set of parties willing to participate honestly and that the adversary not only can corrupt any of these parties throughout the execution (except for two, otherwise the agreement property is meaningless) but can also activate another polynomially large set of parties in $\mathcal{U}$ (apart from the originally honest ones) and make these parties participate maliciously. Notice that any party of the universe will be considered as honest, only if it joins the execution from the beginning and also consistently follows the protocol until its termination.

Since the protocols that we will present later are resilient even to corruptions that may behave in a totally arbitrary way, we can consider that the adversary cannot only use the diffusion functionality that is available to the honest parties in every round, but it can also take advantage of the diffusion network to selectively send messages to specific parties. Hence, our adversarial model in the unknown participants setting is as strong as in the classic model (as defined in section 3.2).

5.2 Byzantine Agreement in the Unknown Participants Setting

Once we have described the unknown participants setting in detail, we can now define some new versions of byzantine agreement problems, which naturally generalise the classic definitions in the case of unknown participants. Although many interesting solutions for *unknown participants byzantine agreement* have been published in the last few years [19, 20], none of these works manages to describe accurately the requirements of an efficient byzantine agreement protocol in the unknown participants setting, and thus, we consider our following definitions to be a valuable contribution

to the literature.

Since we are mostly interested in the case of dishonest majorities, we will focus on the problem of *Broadcast*. Recall that we assume the existence of a PKI (as described in Section 5.1) with a security parameter κ and that there exits a set that contains all possible participants, denoted by $\mathcal{U}$, with size superpolynomial in κ. We will denote by $\mathcal{P}_0 \subseteq \mathcal{U}$ the set of honest parties willing to participate from the beginning of the execution and by $\mathcal{P}_r \subseteq \mathcal{U}$, for every $r > 0$, the set of all parties which have been activated at some point by the end of round r. For convenience, in the rest of this work we will very often simply use $\mathcal{P}$ to denote the set of parties that get activated at some time before the completion of the protocol and we will refer to it as the set of *active* parties.

Notice that the every secure protocol in the unknown participants setting must terminate in $\mathsf{poly}(\kappa)$ many rounds, since, otherwise, the adversary has enough time to forge signatures with non-negligible probability. On the other hand, as we know from the boundaries of the classical setting, the protocol will necessarily have to run for a number of rounds that is proportional to the number of active parties, at least in the worst case execution of the protocol. Hence, an unavoidable restriction to the capabilities of the adversary would be to have to stop activating new parties after an unknown but yet polynomial in κ number of rounds, i.e., there exists some $R = \mathsf{poly}(\kappa)$, such that $\mathcal{P}_r = \mathcal{P}_R$, for every $r \geq R$.

Therefore, we end up with the following definition of the *unknown participants Broadcast*, which is the analogous of the *byzantine Broadcast* (Definition 18) in the unknown participants setting :

Definition 43 (Unknown Participants Broadcast)**.** *Let $\mathcal{U}$ be a set of parties. A protocol π is a unknown participants Broadcast protocol (in short, UP-Broadcast) with respect to $\mathcal{U}$, where a designated sender $s \in \mathcal{U}$ holds an initial input value $b_s \in \{0,1\}$, if for every (strongly) adaptive PPT adversary and every evolving set of active parties $\mathcal{P} \subseteq \mathcal{U}$, the following properties are satisfied with overwhelming probability:*

- ***Validity:*** *If the sender s is honest, then all honest parties output b_s.*
- ***Agreement:*** *All honest parties output the same value $b \in \{0,1\}$.*
- ***Termination:*** *All honest parties terminate and output a value in $\mathsf{poly}(\kappa)$ rounds.*

Notice that the any immediate application of the classic approach of Dolev-Strong does not meet the above definition. In general, any protocol that runs for a predetermined number of rounds would fail to satisfy all the above properties, e.g. assume that there exists some protocol that runs for a fixed (polynomial in κ) number of rounds, let it be $\tilde{R}$, then even a static adversary can activate $\tilde{R} + 2$ parties from the beginning of the execution, so then the protocol would require at least $\tilde{R} + 1$ rounds to guarantee all properties of Broadcast.

Hence, a protocol satisfying the properties of the above definition would have to run for a number of rounds that is proportional to the number of active parties, while all existing approaches (which are based on Dolev-Strong) require the knowledge of the actual number of participants in order to determine the number of rounds required to maintain security.

Difficulty of UP-Broadcast. A UP-Broadcast protocol should definitely have a termination condition that could be met within a $\mathsf{poly}(\kappa)$ number of rounds from the beginning of the execution

and that should simultaneously ensure the parties' agreement on the input of the sender at he time the condition is met. Notice that in the classic approach of Dolev-Strong, whenever some honest party p accepts some possible input value b of the sender in some round r, then p has the power to convince the rest of the honest parties to accept b by round $r+1$ (or, in case of the the final round, is just certain that all other honest parties have already accepted b). However, in case of a more complicated termination condition, it is not so straightforward to guarantee that all honest parties that may have not accepted b by round r will keep participating in the execution by round $r+1$ to receive any certificate for b by p. In other words, correctness of any UP-Broadcast protocol would require that all parties which have not accepted b by round r have neither satisfied the termination condition indicated by the protocol by round r and consequently, participated in round $r+1$ as well.

Our approach. The Dolev-Strong approach in the classic model relies on the idea that a certificate for a value in round r requires the support of at least r parties and thus, the parties can safely terminate simultaneously in round $n-1$, since any certificate in round n would be pointless. Hence, we considered that the corresponding certificates in the unknown participants setting should have the same property, so then the parties could terminate once they realise that they could not receive any valuable certificates in the following rounds. In the unknown participants model, however, the set of active parties may be arbitrarily large and to make things worse, the parties cannot easily distinguish between honest parties communicating with all other honest parties via the diffusion functionality and corrupted ones which selectively communicate only with a subset of the honest parties. Taking this observation into consideration, we concluded that in order to solve UP-Broadcast we would have to come up with a protocol that allows the parties to dynamically agree on a set of active parties $S \subseteq \mathcal{P}$, such that the certificates for the possible input values of the sender would have to contain enough signatures from this specific set S. Hence, once the parties observe this set S to contain no more parties than the round number they could safely terminate. Nonetheless, note that transferability of certificates among parties would require that all honest parties are included in S and that the perceived S sets of honest parties should be updated in parallel.

5.3 Active Parties Agreement Protocol

Once we have explained the motivation, we are now ready to present an *active parties agreement* (APA) protocol, i.e., a protocol that allows the honest parties to agree on a set of active parties $S \subseteq \mathcal{P}$, such that every honest party is included in S. The protocol tolerates any number of corruptions in the unknown participants model.

We begin by formally describing the properties of an APA protocol. For simplicity, we will consider that the parties shall agree on a set of active parties, while in fact they can only agree on a set of identifiers that correspond to active parties. Hence, we provide the following definition of APA:

Definition 44 (Active Parties Agreement (APA))**.** *Let $\mathcal{U}$ be a set of parties. A protocol π is an active parties agreement protocol with respect to $\mathcal{U}$ if for every adaptive PPT adversary and every evolving set of active parties $\mathcal{P} \subseteq \mathcal{U}$, the following properties are satisfied with overwhelming probability:*

– ***Correctness:*** *If some honest party p outputs a set S, then $S \subseteq \mathcal{P}$.*

– ***Validity:*** *If some honest party p outputs a set S, then $p \in S$.*

– ***Agreement:*** *All honest parties output the same set.*

– ***Termination:*** *All honest parties terminate and output a set in $\mathsf{poly}(\kappa)$ rounds.*

Notice that in the above definition the validity property seems to be weaker than the one of the byzantine agreement in the classic model, since it only requires honest parties to be included in their own output sets and not to every honest party's output set. However, once agreement is guaranteed, even this minimal requirement immediately implies that all honest parties are included in the outputs of all honest parties. Hence, this validity expression suffices to satisfy our desired properties and thus, is adopted for convenience in the proofs.

5.3.1 Adapting the idea of Dolev-Strong

In order to achieve agreement on a set of active parties, we adapt the idea of Dolev and Strong (Section 4.2). In particular, instead of transferring votes for the input of a designated sender, the parties shall transfer votes for the participation of other parties along with some proof that these parties have actively participated at some point during the execution of the protocol. Note that the only way some party $p \in \mathcal{P}$ should get convinced about the active participation of some other party $u \in \mathcal{U}$ is by observing u's signature (along with a certificate from the certification authority). We will usually refer to the acceptance of a party for convenience, while in fact, we mean the acceptance of its identifier.

So, in the first round of our protocol ($r = 0$), every party $p \in \mathcal{P}$ signs id_p and diffuses it to the network. From that point and on and in every round $r \geq 1$, p accepts some other party $u \in \mathcal{U}$ as active, only if it has not accepted u by round $r-1$ and observes signatures on id_u from at least r distinct parties including u, out of which everyone, except from u, was accepted by p as active by round $r-1$. Note that for p to accept some new party in round r, it must have accepted at least $r-1$ parties (apart from itself) by round $r-1$. So, if p observes the number of its so far accepted parties to not be greater than r, there is no way it could accept some new party in the rounds to follow and thus, it can safely terminate. However, if that is not the case, then it shall convince the rest of the honest parties about the participation of every party it newly accepted in round r. Hence, it collects all the signatures it observed in round r in support of the participation of u, it adds its own signature on id_u and diffuses this new batch of signatures.

Here, one can ask why is p certain that none of the other honest parties has terminated at some round prior to $r+1$ without accepting u, violating the *agreement* property of the APA. Intuitively, we can argue that, since every honest party can easily convince all honest parties about its participation by round 1 and make them proceed to round 2, it can also convince them in every round r about the participation of every party it has accepted by round $r-1$ including itself, and thus, make them proceed to round $r+1$. We will prove the robustness of this argument in Section 5.3.3.

5.3.2 Formal Description of APA protocol

We will now give a formal description of our APA protocol, which tolerates any PPT adversary. We first introduce the notion of *valid batches of signatures* and *well-formed messages*. In order to

maintain the security of our PKI, we consider that every party $p \in \mathcal{P}$ does not sign its public key itself, but the string produced by concatenating pk_p with a randomly selected salt, denoted by salt_p. We will consider $\mathsf{id}_p = \mathsf{pk}_p||\mathsf{salt}_p$ to be the identifier of p, which is guaranteed to uniquely identify p by the certification authority. Let cert_p be the certificate returned to p by the authority, we will use $\mathsf{Sign}_p(m)$ to denote the quadruple $(\mathsf{id}_p, m, \mathsf{Sign}_{\mathsf{sk}_p}(m), \mathsf{cert}_p)$ for any message $m \in \{0,1\}^{\mathsf{poly}(\kappa)}$.

Valid batches of signatures. Let $S_p^r \subseteq \mathcal{U}$ be the set of parties that some active party $p \in \mathcal{P}$ has accepted by round $r \geq 0$ and let B be a batch of signatures on id_u, for some $u \in \mathcal{U}$. Then, we will call B a (p,r)-*valid batch of signatures* for u if it consists of signatures on id_u from at least r distinct parties including u, such that at least $r-1$ of them are included in S_p^{r-1}. More formally, let $B = \{\mathsf{Sign}_v(\mathsf{id}_u)\}_{v \in V}$, for some $V \subseteq \mathcal{P}$, then B is a (p,r)-*valid batch of signatures* if $u \in V$ and $|V \cap S_p^{r-1}| \geq r-1$.

Well-formed messages. As we mentioned earlier, in every round $r \geq 0$, we would like all honest parties (which have decided to proceed to the next round) to diffuse batches of signatures in support of the participation of every party they have newly accepted. So, since we want to protect the honest parties from reading messages generated by corrupted parties, every party $p \in \mathcal{P}$ shall include at most one batch of signatures for each newly accepted party in the message it diffuses at the end of every round r. In particular, we will call such a message M_p^r *well-formed* if $M_p^r = \{B_i\}_{i=1}^l$, for some $l \geq 1$, where B_i are batches of signatures in support of distinct parties $u_i \in \mathcal{U}$, such that $B_i = \{\mathsf{Sign}_v(\mathsf{id}_{u_i})\}_{v \in V_i}$, for some $V_i \subseteq \mathcal{U}$ with $u_i \in V_i$, for all $i \in [1,l]$. In the following protocol, honest parties will only process *well-formed* messages and discard the rest.

We give a formal description of our protocol in Figure 5.3.1.

Protocol π_{apa}

Initialization: Every party $p \in \mathcal{U}$ willing to participate, generates $(\mathsf{sk}_p, \mathsf{pk}_p) \leftarrow \mathsf{KeyGen}(1^\kappa)$ and $\mathsf{salt}_p \xleftarrow{\$} \{0,1\}^\kappa$. Then, it sends $\mathsf{id}_p = \mathsf{pk}_p||\mathsf{salt}_p$ to the certification authority and receives cert_p.

Round 0: Every party $p \in \mathcal{P}^0$ initializes $S_p^0 = \{p\}$ and calls $\mathcal{F}_{\mathsf{diffuse}}(\{\{\mathsf{Sign}_p(\mathsf{id}_p)\}\})$.

Loop: For $r = 1, 2, \ldots$, every party $p \in \mathcal{P}$ does the following:

1. Initializes $S_p^r = S_p^{r-1}$.
2. Upon receiving some well-formed message M, for all $B \in M$, do the following: if B is a (p,r)-valid batch of signatures for some party $u \notin S_p^{r-1}$, then:
   - if $u \notin S_p^r$, initialize $C_{p,u}^r = \emptyset$
   - set $S_p^r \leftarrow S_p^r \cup \{u\}$ and $C_{p,u}^r \leftarrow C_{p,u}^r \cup B$.
3. If $|S_p^r| \leq r$, then terminate with output S_p^r, else call $\mathcal{F}_{\mathsf{diffuse}}(\{C_{p,u}^r \cup \{\mathsf{Sign}_p(\mathsf{id}_u)\}\}_{u \in S_p^r \setminus S_p^{r-1}})$ and proceed to round $r+1$.

Figure 5.3.1: Active Parties Agreement Protocol

Note that in every round $r \geq 1$, every party $p \in \mathcal{P}$ stores for every $u \in S_p^r \setminus S_p^{r-1}$ all the signatures

it has observed in support of u in round r in $C^r_{p,u}$. In case p decides to proceed to round $r+1$, it forms a new batch of signatures in support of each $u \in S^r_p \setminus S^{r-1}_p$, by adding to $C^r_{p,u}$ its own signature on id_u, and multicasts these new batches at the end of round r.

5.3.3 Correctness of APA protocol

In this section, we will formally prove that the protocol of Figure 5.3.1 satisfies the conditions of Definition 44 with overwhelming probability.

We will say that some party $p \in \mathcal{U}$ *accepts* some party $u \in \mathcal{U}$ in round r, if $r = \min\{k \in \mathbb{N} : u \in S^k_p\}$.

Lemma 45 (Correctness)**.** *If some honest party $p \in P$ accepts some party $u \in \mathcal{U}$ before terminating, then $u \in \mathcal{P}$.*

Proof. If u is accepted by some honest party $p \in \mathcal{P}$, at some round $r \geq 1$, then p received a (p, r)-valid batch of signatures in support of u, let it be B, so by the definition of valid batches we have that $\mathsf{Sign}_u(\mathsf{id}_u) \in B$. Thus, with overwhelming probability $u \in \mathcal{P}$. □

Notice that the above lemma immediately implies the *termination* property as well, since honest parties only accept active parties throughout the execution. Hence, in every round r every honest party $p \in \mathcal{P}$ observes $S^r_p \subseteq \mathcal{P}$ and thus, observes $|S^r_p| \leq r$ by round $|\mathcal{P}|$.

Lemma 46 (Validity)**.** *Every honest party $p \in \mathcal{P}$ includes p in its output.*

Proof. Notice that every honest party $p \in \mathcal{P}$ sets $S^0_p = \{p\}$ at the beginning of the execution and thus, let r be the round in which p terminates, then it outputs S^r_p with $S^0_p \subseteq S^r_p$. □

So, now it suffices to show that the protocol satisfies the *agreement* property of APA. Notice that for the agreement property to be violated, there must exist two honest parties $p, q \in \mathcal{P}$ and some party $u \in \mathcal{U}$, such that u is included in the output of p but not in the one of q. Hence, to complete our proof, it suffices to show that whenever p accepts some party u, then q also accepts u before terminating. Will will prove an even stronger statement (similar to one of Dolev-Strong's security proof) using an induction on the number of rounds.

Lemma 47 (Agreement)**.** *If some honest party $p \in \mathcal{P}$ accepts some party $u \in \mathcal{U}$ in round r, then every honest party $q \in \mathcal{P}$ accepts u by round $r+1$ before terminating.*

Proof. First, we will show that the lemma holds for $r = 0$. In round 0, the only party that every honest party p accepts is itself and thus, it diffuses $\{\{\mathsf{Sign}_p(\mathsf{id}_p)\}\}$ at the end of round 0. Hence, every honest party $q \neq p$ receives p's message by round 1 and accepts p, since $p \notin S^0_q$ and $\{\mathsf{Sign}_p(\mathsf{id}_p)\}$ is a $(q, 1)$-valid batch of signatures for p.

Now assume that the lemma holds for every round $r' < r$, where $r \geq 1$. Since p accepts u in round r, p receives a (p, r)-valid batch of signatures for u in round r, let it be $B = \{\mathsf{Sign}_v(\mathsf{id}_u)\}_{v \in V}$, for some $V \subseteq \mathcal{P}$, and observes $u \notin S^{r-1}_p$ and $|V \cap S^{r-1}_p| \geq r-1$. So, there exists a set of parties $W \subseteq V \cap S^{r-1}_p$ with $|W| \geq r-1$, such that p has accepted every party of W by round $r-1$. We observe that with overwhelming probability $p \notin V$ and thus $p \notin W$, since $p \in V$ would imply that p's signature on id_p was forged at some round prior to r. Moreover, let q be some honest party with $q \neq p$, then using

the induction hypothesis for all parties in W, we have that q accepts all parties of W by round r, except with probability $|W| \cdot \mathsf{negl}(\kappa)$ which is negligible, since $W \subseteq \mathcal{P}$. Now, if q has accepted u by round r, then this completes the proof. If q has not accepted u by round r, we can use the same argument we used for p in order to show that $q \notin W$. So, let $r' \leq r$ be the last round in which q accepts some party of W, then, since q observes $W \cup \{p, q\} \subseteq S_q^{r'}$ in round r', it also observes $|S_q^{r''}| \geq |S_q^{r'}| \geq r+1 > r''$ in every round $r'' \in [r', r]$ and thus, participates in round $r+1$. Since p is honest in round r, it diffuses a message at the end of round r, which is well-formed by construction and contains some batch of signatures for u, let it be $B' = \{\mathsf{Sign}_v(\mathsf{id}_u)\}_{v \in V'}$, with $V \cup \{u\} \subseteq V'$. Therefore, q receives B' in round $r+1$ and observes $u \in V' \setminus S_q^r$ and $|V' \cap S_q^r| \geq |W \cup \{p\}| \geq r$. Thus, q accepts u in round $r+1$. □

Once we have proved that our protocol achieves agreement and termination in at most $|\mathcal{P}|$ rounds, one could ask if there is some connection between the round numbers that the honest parties terminate, e.g., whether there is a chance that some honest parties terminate much earlier than others. Using a simple observation, we can prove that all honest parties terminate simultaneously and actually, the termination takes exactly $|S|$ rounds, where S is the common output of the parties.

Lemma 48 (Simultaneous Termination)**.** *Let S be the common output of the honest parties, then all honest parties terminate simultaneously in round $|S|$.*

Proof. Let S_p be the output of some honest party p, which terminates in round $r \geq 1$. Then p observes $S_p = S_p^r$ and $|S_p^r| \leq r$ in round r. Now assume that $|S_p^r| < r$, then $|S_p^{r-1}| \leq |S_p^r| \leq r-1$ and thus, p would have terminated by round $r-1$, a contradiction. So, for round r it holds that $r = |S_p^r| = |S|$. Using Lemma 47, we have that all honest parties output the same set $S \subseteq \mathcal{P}$ and thus, terminate simultaneously in round $r = |S|$. □

Hence, we can state the following theorem:

Theorem 17. *Protocol π_{apa} (Figure 5.3.1) is an active parties agreement protocol (according to Definition 44) that is secure against any adaptive PPT adversary. Moreover, let S be the common output of the honest parties, then all honest parties terminate simultaneously in round $|S| \leq |\mathcal{P}|$.*

5.3.4 APA Implies UP-Broadcast

Notice that the APA protocol present above suffices to solve unknown participants Broadcast. In particular, one can easily come up with a UP-Broadcast protocol by running APA to achieve agreement on a set of active parties and then, by running Dolev-Strong on the agreed party set. However, this solution doubles the number of rounds required for termination.

Another way to achieve unknown participants Broadcast using APA is the following: every party willing to participate in UP-Broadcast, except potentially for the sender, starts running an APA instance, and the sender participates in the APA only if its input value is equal to 1. Hence, once the APA execution is completed, then the parties shall output 1, only if the sender's identifier is included in the agreed set of active parties and otherwise, output 0. This simple reduction is shown in Figure 5.3.2.

Protocol $\pi_{\mathsf{b}}(\pi_{\mathsf{apa}})$

Initialization: Every party $p \in \mathcal{U}\backslash\{s\}$ willing to participate, generates $(\mathsf{sk}_p, \mathsf{pk}_p) \leftarrow \mathsf{KeyGen}(1^\kappa)$ and $\mathsf{salt}_p \xleftarrow{\$} \{0,1\}^{l_s}$. Then, it sends $\mathsf{id}_p = \mathsf{pk}_p||\mathsf{salt}_p$ to the certification authority and receives cert_p and id_s (we assume that id_s is predetermined).

Active Parties Agreement: Every party $p \in \mathcal{P}\backslash\{s\}$ starts running a π_{apa} instance and the sender s participates in π_{apa} iff $b_s = 1$, otherwise remains silent throughout its execution.

Output: Upon receiving the output S from π_{apa}, every party $p \in \mathcal{P}$ outputs 1 if $s \in S$, otherwise outputs 0.

Figure 5.3.2: UP-Broadcast Protocol

Note that in UP-Broadcast the identifier of the sender s must be determined before the execution of the protocol, so then the parties have a common reference on the party whose message they have to agree on. One can also easily observe that all properties of UP-Broadcast are guaranteed by the construction of the above protocol and the corresponding properties of APA.

Theorem 18. *Protocol $\pi_{\mathsf{b}}(\pi_{\mathsf{apa}})$ (Figure 5.3.2) is a unknown participants Broadcast protocol (according to Definition 43) that is secure against any adaptive PPT adversary. Moreover, the protocol guarantees simultaneous termination in at most $|\mathcal{P}|$ rounds.*

Proof. If s is honest and has input value $b_s = 1$, then it participates in π_{apa} and by the validity and agreement properties of APA, every honest party includes s in its output set. Hence, every honest party outputs 1. If s is honest and has input value $b_s = 0$, then it behaves as inactive, and thus, by the correctness property of APA, s in not included in the output of any honest party. Therefore, every honest party outputs 0. Finally, agreement and termination of UP-Broadcast are immediately implied by the agreement on the output of the APA execution and its guaranteed termination in at most $|\mathcal{P}|$ rounds (agreement and termination properties of APA), which completes the proof. □

Nonetheless, the reduction above heavily relies on the fact that the input value of the designated sender can only take two possible values, which is anyway the assumption that we have made so far throughput this work. Having said that, one may wonder whether there is a way to modify APA so then it can be extended to UP-Broadcast even in the case where the sender has a *multi-valued* input, i.e., a string of more than one bits.

In the next section, we evolve our APA protocol into an *unknown participants interactive consistency protocol*, i.e., a protocol that solves the analogous of interactive consistency (as defined in 21) in the unknown participants setting. As we will explain later, this immediately implies that even the *multi-valued* version of UP-Broadcast can be solved in the same number of rounds as APA.

5.4 Unknown Participants Interactive Consistency Protocol

In this section, we will present an *unknown participants interactive consistency* (UP) protocol that tolerates any number of corruptions. We begin by providing a definition for the interactive consistency protocol in the unknown participants model. Even though our motivation for UP-Interactive Consistency is the need to support a multi-valued-input of the sender in PB, we will once again study the binary version of the protocol, where every party $p \in \mathcal{P}$ willing to participate holds an initial input value $b_p \in \{0,1\}$ (the extension to multi-valued UP-Interactive Consistency is obvious and hence, omitted).

Definition 49 (Unknown Participants Interactive Consistency)**.** *A protocol π is an unknown participants interactive consistency (in short, UP-Interactive Consistency) protocol with respect to $\mathcal{U}$ if for every adaptive PPT adversary and every evolving set of active parties $\mathcal{P} \subseteq \mathcal{U}$, where every party $p \in \mathcal{P}$ holds an initial input value $b_p \in \{0,1\}$, the following properties are satisfied except for negligible probability:*

- ***Correctness:*** *If some honest party p outputs a set A, then $A \subseteq \mathcal{P} \times \{0,1\}$.*
- ***Validity:*** *If some honest party p outputs a set A, then $(p, b_p) \in A$.*
- ***Agreement:*** *All honest parties outputs the same set of party-value pairs.*
- ***Termination:*** *All honest parties terminate and output a set in $\mathsf{poly}(\kappa)$ rounds.*

Notice that the above definition constitutes a natural extension of our definition for APA (Definition 44) for the case that all parties have input values. In this case, the honest parties not only have to agree on set of active parties including the honest ones, but also have to agree on the same value for each one of the accepted parties.

5.4.1 Building on the APA protocol

Once we have a protocol that achieves agreement on a set of active parties (which includes all honest parties) we can use it as the basis of an interactive consistency protocol.

Now, in round 0, every party $p \in \mathcal{P}$, instead of signing id_p, signs $\mathsf{id}_p||b_p$ and diffuses it to the network. So, all honest parties shall diffuse batches of signatures in support of party-value pairs of the form $\mathsf{Sign}_v(\mathsf{id}_u||b)$, where $u, v \in \mathcal{U}$, $b \in \{0,1\}$. The acceptance condition for an party-value pair (u,b), which is also the acceptance condition for the participation of u, is almost identical to the one for the acceptance of an party that we had in our APA protocol, i.e., any honest party $p \in \mathcal{P}$ accepts some pair (u,b) (as well as the participation of u) in round $r \geq 1$, only if it has not accepted (u,b) by round $r-1$ and observes signatures on $\mathsf{id}_u||b$ from at least r distinct parties including u, out of which everyone except for u, was accepted by p by round $r-1$. Once some party observes the number of its so far accepted parties to be less or equal than the round number, it can safely terminate, exactly as in our APA protocol.

The difference now is that the output of every honest party $p \in \mathcal{P}$ will be a set of party-value pairs, which will indicate the values that p decides for each one of its accepted parties. In particular, for every party u that p has accepted, p will decide 1, if it has only accepted 1 with respect to u and 0, otherwise. In this way, the agreement on the honest parties' output is implied from a relay

property very similar to the one we proved in Lemma 47 for our APA protocol, which will now concern the acceptance of party-value pairs.

5.4.2 Formal Description of UP-Interactive Consistency Protocol

We will now give a formal description of our interactive consistency protocol which tolerates any PPT adversary in the unknown participants setting. First, we have to extend the definitions of *valid batches of signatures* and *well-formed messages* that we gave in Section 5.3.2.

Valid batches of signatures. We now extend the definition of *valid batches of signatures* to the case that the signatures of the batch are in support of a party-value pair. So, let B be a batch of signatures on some party-value pair $(u, b) \in \mathcal{P} \times \{0, 1\}$, then we will call B a (p, r)-*valid batch of signatures*, if it consists of signatures on $\mathsf{id}_u||b$ from at least r distinct parties including u, such that at least $r - 1$ of them are included in S_p^{r-1}, i.e., let $B = \{\mathsf{Sign}_v(\mathsf{id}_u||b)\}_{v \in V}$, for some $V \subseteq \mathcal{P}$, then B is a (p, r)-*valid batch of signatures* if $u \in V$ and $|V \cap S_p^{r-1} \backslash \{u\}| \geq r - 1$.

Well-formed messages. In our interactive consistency protocol, the honest parties will have to diffuse batches of signatures in support of the participation of their newly accepted party-value pairs, exactly as they did in our APA protocol, and in order to maintain APA's efficiency, every party shall diffuse at most one batch of signatures for each newly accepted pair. So, let $\overset{r}{p}$ be the message that some party $p \in \mathcal{P}$ diffuses at the end of round r, then $\overset{r}{p}$ will be called *well-formed* if $\overset{r}{p} = \{B_j\}_{j=1}^{l}$, for some $l \geq 1$, where B_i are batches of signatures in support of distinct pairs (u_i, b_i), such that for all $i \in [1, l]$, $B_i = \{\mathsf{Sign}_v(\mathsf{id}_{u_i}||b_i)\}_{v \in V_i}$, for some $V_i \subseteq \mathcal{U}$ with $u_i \in V_i$.

Output. Let A_p^r be the set of party-value pairs that some party $p \in \mathcal{P}$ has accepted by round r and for every $u \in S_p^r$, let $A_{p,u}^r = \{b \in \{0, 1\} : (u, b) \in A_p^r\}$ be the set of values that p has accepted by round r with respect to u. Them, if p decides to terminate in round r, it outputs:

$$\mathsf{Out}(A_p^r) = \{(u, 1)\}_{u \in \hat{S}_p^r} \cup \{(u, 0)\}_{u \in S_p^r \backslash \hat{S}_p^r}$$

where $\hat{S}_p^r = \{u \in S_p^r : A_{p,u}^r = \{1\}\}$.

We give a formal description of our protocol in Figure 5.4.1.

Note that p stores in every round r all the signatures it has observed in support of every party-value pair $(u, b) \in A_p^r \backslash A_p^{r-1}$ in the $C_{p,u,b}^r$ sets. The way it constructs the message that it diffuses at the end of round r in a similar way to the one of our APA protocol.

5.4.3 Proof of Correctness

In this section, we will formally prove that the protocol of Figure 5.4.1 satisfies the conditions of Definition 49.

Accordingly to the acceptance of identifiers, we will say that some party $p \in \mathcal{P}$ *accepts* some pair (u, b) in round r, if $r = \min\{k \in \mathbb{N} : (u, b) \in A_p^k\}$.

Protocol π_{ic}

Initialization: Every party $p \in \mathcal{P}$ generates $(\mathsf{sk}_p, \mathsf{pk}_p) \leftarrow \mathsf{KeyGen}(1^\kappa)$ and $\mathsf{salt}_p \xleftarrow{\$} \{0,1\}^{\ell_s}$. Then, it sends $\mathsf{id}_p = \mathsf{pk}_p||\mathsf{salt}_p$ to the certification authority and receives cert_p.

Round 0: Every party $p \in \mathcal{P}$ initializes $S_p^0 = \{p\}$ and $A_p^0 = \{(p, b_p)\}$ and calls $\mathcal{F}_{\mathsf{diffuse}}(\{\{\mathsf{Sign}_p(\mathsf{id}_p||b_p)\}\})$.

Loop: For $r = 1, 2, \ldots$, every party $p \in \mathcal{P}$ does the following:

1. Initializes $S_p^r = S_p^{r-1}$ and $A_p^r = A_p^{r-1}$.
2. Upon receiving some well-formed message M, for all $B \in M$, do the following: if B is a (p, r)-valid batch of signatures for some pair $(u, b) \notin A_p^{r-1}$, then :
   - if $(u, b) \notin A_p^r$, initialize $C_{p,u,b}^r = \emptyset$
   - set $S_p^r \leftarrow S_p^r \cup \{u\}$, $A_p^r \leftarrow A_p^r \cup \{(u, b)\}$ and $C_{p,u,b}^r \leftarrow C_{p,u,b}^r \cup B$.
3. If $|S_p^r| \leq r$, then terminate with output $\mathsf{Out}(A_p^r)$, else call $\mathcal{F}_{\mathsf{diffuse}}(\{C_{p,u,b}^r \cup \{\mathsf{Sign}_p(\mathsf{id}_u||b)\}\}_{(u,b)\in A_p^r \setminus A_p^{r-1}})$ and proceed to round $r + 1$.

Figure 5.4.1: Unknown Participants Interactive Consistency Protocol

Lemma 50 (Correctness)**.** *If some honest party $p \in P$ accepts some pair $(u, \cdot)$, where $u \in \mathcal{U}$ before terminating, then $u \in \mathcal{P}$.*

Proof. If (u, b) is accepted by some honest party $p \in \mathcal{P}$ for some $b \in \{0, 1\}$, at some round $r \geq 1$, then p received a (p, r)-valid batch of signatures in support of (u, b), let it be B, so by the definition of valid batches we have that $\mathsf{Sign}_u(\mathsf{id}_u||b) \in B$. Thus, with overwhelming probability $u \in \mathcal{P}$. □

Lemma 51 (Validity)**.** *If some party $p \in \mathcal{P}$ is honest, then (p, b_p) is included in the output of p.*

Proof. If p is honest, then it accepts (p, b_p) from round 0. Now assume for the contrary that p accepts some other pair of the form (p, b) for some $b \neq b_p$ in some round $r \geq 1$, then p must have received a (q, r)-valid batch of signatures for (p, b) in round r, let it be B, such that $\mathsf{Sign}_p(\mathsf{id}_p||b) \in B$. Hence, with overwhelming probability p has signed $\mathsf{id}_p||b$ at some round prior to r, which is a contradiction, since p is honest. Hence, by construction of the output set ($\mathsf{Out}(\cdot)$ function), (p, b_p) is included in the output of p. □

Notice that the termination condition of the UP-Interactive Consistency protocol as well the handling of the sets of accepted parties is exactly the same as in our APA protocol. In addition, the number of pairs accepted with respect to some particular identifier does not have any impact on the decision of the parties about the termination of the protocol other than the acceptance of the identifier itself. Hence, all the properties regarding the acceptance of parties and the satisfaction of the termination condition that we proved for our APA protocol (Lemmas 47 and 48) must hold in our UP-Interactive Consistency protocol as well.

Therefore, the only property left to complete the proof is the agreement on the output of the honest parties. Our proof relies on the fact that let A_p and A_q be the sets that some honest parties p and

q output once they terminate, then the only way the agreement property may be violated is that there exists some pair (u, b), such that $(u, b) \in A_p \backslash A_q$ (otherwise, the $\mathsf{Out}(\cdot)$ function will map the sets A_p and A_q to the same output set). Thus, we will once again prove a relay property, which will now concern the acceptance of party-value pairs.

Lemma 52 (Agreement)**.** *If some honest party $p \in \mathcal{P}$ accepts some pair (u, b) in round r, then every honest party q accepts (u, b) by round $r + 1$ before terminating.*

Proof. First, we will prove that the lemma holds for $r = 0$. In round 0, p only accepts the (p, b_p) pair and diffuses $\{\{\mathsf{Sign}_p(\mathsf{id}_p||b_p)\}\}$ at the end of round 0. In round 1, every honest party $q \neq p$ receives p's message and accepts (p, b_p), since $p \notin S_q^0$ and $\{\mathsf{Sign}_p(\mathsf{id}_p||b_p)\}$ is a $(q, 1)$-valid batch of signatures for (p, b_p).

Now if p accepts (u, b) in round $r \geq 1$ for some $u \in \mathcal{U}$, then p receives a (p, r)-valid batch of signatures for (u, b) in round r, let it be $B = \{\mathsf{Sign}_v(\mathsf{id}_u||b)\}_{v \in V}$, for some $V \subseteq \mathcal{P}$, and observes $(u, b) \notin A_p^{r-1}$ and $|V \cap S_p^{r-1}| \geq r - 1$. So, there exists a set of parties $W \subseteq V \cap S_p^{r-1}$ with $|W| \geq r - 1$, such that p has accepted every party of W by round $r - 1$. We observe that with overwhelming probability $p \notin V$ and thus $p \notin W$, since $p \in V$ would imply that p's signature on $u||b$ was forged at some round prior to r. Moreover, let q be some honest party with $q \neq p$, then using Lemma 47 for all parties in W, we have that q accepts all parties of W by round r. Now, if q has accepted (u, b) by round r, then this completes the proof. If q has not accepted (u, b) by round r, we can use the same argument we used for p in order to show that $q \notin W$. So, let $r' \leq r$ be the last round in which q accepts some party of W, then, since q observes $W \cup \{p, q\} \subseteq S_q^{r'}$ in round r', it also observes $|S_q^{r''}| \geq |S_q^{r'}| \geq r + 1 > r''$ in every round $r'' \in [r', r]$ and thus, participates in round $r + 1$. Since p is honest in round r, it diffuses a message at the end of round r which is well-formed by construction and contains some batch of signatures for (u, b), let it be $B' = \{\mathsf{Sign}_v(\mathsf{id}_u||m)\}_{v \in V'}$, with $V \cup \{p\} \subseteq V'$. Therefore, q receives B' in round $r + 1$ and observes $(u, b) \notin A_q^r$ and $|V' \cap S_q^r| \geq |W \cup \{p\}| \geq r$. Thus, q accepts (u, b) in round $r + 1$. □

Hence, we can summarise the above statements in the following theorem:

Theorem 19. *Protocol π_{ic} (Figure 5.4.1) is an unknown participants interactive consistency protocol (according to Definition 49) that is secure against any adaptive PPT adversary. Moreover, let A be the common output of the honest parties, then all honest parties terminate simultaneously in round $|A| \leq |\mathcal{P}|$.*

5.4.4 Relation with Unknown Participants Broadcast

One can easily observe that the UP-Interactive Consistency protocol presented above solves the problem of unknown participants Broadcast as well. In particular, in case that a particular sender s is determined and has an input value b_s, then UP-Broadcast can be achieved by running an UP-Interactive Consistency instance, where s inputs b_s, while the (rest of) honest parties input a default value. Once the UP-Interactive Consistency execution is completed, then the honest parties can decide 1 if $(s, 1)$ is included in their output and 0, otherwise. In this way, the properties of validity, agreement and termination of UP-Interactive Consistency guarantee the corresponding ones of UP-Broadcast.

Protocol $\pi_{\mathsf{b}}(\pi_{\mathsf{ic}})$

Initialization: Every party $p \in \mathcal{P}$ is given id_s.

UP-Interactive Consistency: Every party $p \in \mathcal{P}\backslash\{s\}$ starts running a π_{ic} instance with input $b_p = 0$ and the sender s participates in π_{ic} with input b_s.

Output: Upon receiving the output A from π_{ic}, every party $p \in \mathcal{P}$ outputs 1 if $(s, 1) \in A$, otherwise outputs 0.

Figure 5.4.2: Unknown Participants Broadcast Protocol (extendable to multi-valued version)

Theorem 20. *Protocol π_{b} (Figure 5.4.2) is an unknown participants Broadcast protocol (according to Definition 43) that is secure against any adaptive PPT adversary. Moreover, the protocol guarantees simultaneous termination in at most $|\mathcal{P}|$ rounds.*

Proof. If the sender is honest, then using the validity and agreement properties of UP-Interactive Consistency, we have that all honest parties observe (s, b_s) in their outputs. Hence, all honest parties output b_s. Agreement and termination of UP-Broadcast are implied by the agreement and termination properties of UP-Interactive Consistency and the construction of the protocol. □

However, notice that the above reduction does not imply that every UP-Broadcast protocol necessarily has to rely on an underlying UP-Interactive Consistency protocol. In fact, although Broadcast and interactive consistency are almost totally equivalent in the classic model, the obvious reduction from the latter to the former, where a Broadcast instance is run for every possible sender, clearly cannot be applied in the unknown participants model where the set of actual participants is not known in advance. Thus, there may exist a UP-Broadcast protocol that could not be transformed into a UP-Interactive Consistency protocol (at least with a similarly simple adaptation as in the classic model) and hence, one could consider the UP-Interactive Consistency problem to be potentially strictly harder than UP-Broadcast. Although, we have no evidence either towards this direction or its opposite, we are inclined to believe that these two problems (along with APA) are may in fact be equally hard. The reason for this, is that, as will prove in the following section, all three problems that we have investigated in the unknown participants model have the same worst-case round complexity, which happens to coincide the round complexity of the protocols we presented above, i.e., the number of active parties in the execution.

5.5 Tightness of Wort-Case Round Complexity

In this section, we will prove that all three problems we have studied in the unknown participants model, i.e., UP-Broadcast, APA, UP-Interactive Consistency, require at least $|\mathcal{P}|$ rounds in the worst case execution of any protocols that succeed in solving them even with a non-negligible probability. In fact, we show that for any *randomized* protocol that solves any of the three problems, there exists an execution, where the protocol errs with probability at least $1/2$. Here we refer to the randomized versions of the protocols defined in Sections 5.3, 5.2, 5.4, as the protocols that achieve

the same properties with the original definitions with probability strictly greater than 1/2 and not necessarily overwhelming.

Our argument adopts the approach of Garay *et al.* [8] that we presented in subsection 4.1.1, who managed to show that every randomized Broadcast protocol requires at least $n-1$ rounds in the classic model. Particularly, we prove that in the unknown participants setting, the parties may actually need to run for at least $|\mathcal{P}|$ rounds in order to achieve Broadcast, which means that they have to run at least one more round that what they could in case they knew the actual number of participants.

In fact, while in the classic model the honest parties can make sure they have consistent views with each other about the behaviour of the sender in $n-1$ rounds, in the unknown participants model, when the sender is not active there may exist some honest party which is aware of the participation of n parties (including itself) by round $n-1$, but is not certain whether these parties have received an input of the sender or not. More precisely, the party may not be able to distinguish between the scenario where the sender is not active and the one where the sender message has been received by all other parties except for itself. Hence, this party has to run for at least one more round to resolve the situation.

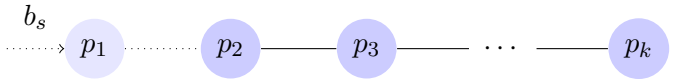

Figure 5.5.1: Configuration where PB requires at least $|\mathcal{P}|$ rounds

Theorem 21. *There does not exist any randomized unknown participants Broadcast protocol that terminates in fewer than $|\mathcal{P}|$ rounds.*

Proof. Assume for the contrary that there exists a randomized unknown participants Broadcast protocol π that terminates in at most $|\mathcal{P}|-1$ rounds. Let $P = \{p_i\}_{i=1}^{k} \subseteq \mathcal{U}$ be a set of $k = \mathsf{poly}(\kappa)$ parties, where p_1 is the designated sender with input $b_s = b$. Thus, we can consider k scenarios S_i, $i \in [1,k]$ of π, where for $i \in [1,k-1]$ we consider that $\mathcal{P}_i = P$ and $\mathcal{P}_k = P \backslash \{p_1\}$.

Moreover, in each scenario S_i, $i \in [1,k-1]$ we consider that the adversary $\mathcal{A}_i$ corrupts all parties in $\mathcal{P}_i$ except for p_i and p_{i+1}, while in S_k, $\mathcal{A}_k$ corrupts all parties in $\mathcal{P}_k$ except for p_{k-1} and p_k. We also consider that whenever some $\mathcal{A}_i$ corrupts some party p_j, it forces it to follow the instructions of π_j, except that:

- if $j = 1$, then p_1 ignores all the messages sent to it except for those from p_2, and only sends messages to p_2.
- if $j \in [2,k-1]$, then p_j ignores all the messages sent to it except for those from p_{j-1} and p_{j+1}, and only sends messages to p_{j-1} and p_{j+1}.
- if $j = k$, then p_k ignores all the messages sent to it except for those from p_{k-1}, and only sends messages to p_{k-1}.

Thus, for every $i \in [2,k-1]$, p_i cannot distinguish whether it is on S_{i-1} or S_i. Aditionally, in scenario S_1, p_1 and p_2 are both honest and hence, they both have to output b at the end of the

protocol. But since p_2 cannot distinguish whether it is in S_1 or S_2, we observe that p_2 has to output b in scenario S_2 as well. Moreover, prior to round 1, the view of p_2 is completely independent of b and thus, prior to round 2, the view of p_3 is completely independent of b.

In general, in every scenario S_i, $i \in [1, k-1]$, parties p_i and p_{i+1} have to output b, while the view of p_{i+1} is independent of b prior to round i. In addition, p_k cannot distinguish whether is is in S_k or S_{k-1} by round $k-2$, while in S_k we have that $|\mathcal{P}_k| = k-1$. So, p_k terminates by round $k-2$ in S_k by assumption and therefore, terminates by round $k-2$ in S_k as well. However, in S_k it does not have any knowledge of b and thus, outputs b with probability $1/2$, a contradiction. □

Corollary 53. *There does not exist any randomized active parties agreement protocol that terminates in fewer than $|\mathcal{P}|$ rounds.*

Proof. Assume for the contrary that there exists a randomized active parties agreement protocol π_{apa} that terminates in fewer than $|\mathcal{P}|$ rounds then we can construct an unknown participants Broadcast protocol π_{b} as in Figure 5.3.2. Then, π_{b} is a randomized unknown participants Broadcast protocol which terminates in fewer than $|\mathcal{P}|$ rounds and succeeds with probability at least as great as the success probability of π_{apa}. However, this contradicts Theorem 21. □

Corollary 54. *There does not exist any randomized unknown participants interactive consistency protocol that terminates in fewer than $|\mathcal{P}|$ rounds.*

Proof. Assume for the contrary that there exists a randomized unknown participants interactive consistency protocol π_{ic} that terminates in fewer than $|\mathcal{P}|$ rounds then we can construct an unknown participants Broadcast protocol π_{b} as in Figure 5.4.2. Then, π_{b} is a randomized unknown participants Broadcast protocol which terminates in fewer than $|\mathcal{P}|$ rounds, which contradicts Theorem 21. □

Chapter 6

Conclusion

The problem of byzantine agreement is one of the most fundamental primitives in fault-tolerant distributed computing. The development of blockchain technologies has attracted the attention of the scientific community towards constructing efficient solutions for this problem for the case of dynamic participation. However, most existing works either assume a static participation or a majority of honest parties.

In this work, we provide the first byzantine Broadcast protocol that tolerates any number of corruptions in the unknown participants model (UP-Broadcast). Along with concrete definitions of our model, we also introduce a new type of problem, called active parties agreement, which is the basis for our solution for Broadcast. Additionally, we solve the analogous of interactive consistency in the unknown participants model (UP-Interactive Consistency) and show how UP-Broadcast can by reduced to UP-Interactive Consistency. We also prove that the number of rounds required in all our protocols equals the number of active parties in every execution.

Finally, both three problems that we study in the unknown participants model require at least as many rounds as the number of active parties in any execution. Hence, it turns out, that our protocols are optimal in terms of their worst-case round complexity.

6.1 Future Work

Our results can motivate further research on fault-tolerant protocols in the unknown participants model.

First of all, it would be very interesting to investigate whether one can apply techniques similar to the ones of [11, 20], in order to construct a sublinear-round UP-Broadcast protocol. In addition, one could also attempt to modify the tools of [39] in an effort to achieve early-stopping in the UP model. These tools heavily rely on a commonly agreed party set, but the idea of creating transferable certificates among honest parties using accusation messages may be adaptable to the UP model.

Towards a different direction, one could try to analyse the bit complexity of UP-Broadcast, or even the message complexity in the sense of number of calls to the diffusion functionality. Notice

that our protocols have a greater bit complexity than the one indicated by classic communication lower bounds [9] and hence, it is not clear whether the agreement on a set of active parties (which is the main cause of this bit complexity gap) is necessary to achieve UP-Broadcast. In terms of round complexity, one could also try to disprove the sublinear-round protocol feasibility or even the early-stopping feasibility.

Moreover, it would be interesting to find out whether protocols with improved complexity measures can be constructed in our model for the case of weaker adversarial behaviours, like crash faults or send-omission faults, as presented in [40]. The difficulties of achieving UP-Broadcast in our model could be overcame assuming that accusation messages are trustworthy.

Finally, an interesting direction for further research would be to study Broadcast under an increased number of corruptions in different adversarial and knowledge models, which have been widely investigated in multiple variations [41, 42, 43, 44, 45].

Bibliography

[1] M. C. Pease, R. E. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM*, vol. 27, no. 2, pp. 228–234, 1980.

[2] L. Lamport, R. E. Shostak, and M. C. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, 1982.

[3] M. J. Fischer and N. A. Lynch, "A lower bound for the time to assure interactive consistency," *Information Processing Letters*, vol. 14, no. 4, pp. 183–186, 1982.

[4] D. Dolev and H. R. Strong, "Authenticated algorithms for Byzantine agreement," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 656–666, 1983.

[5] R. A. DeMillo, N. A. Lynch, and M. Merritt, "Cryptographic protocols," in *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA* (H. R. Lewis, B. B. Simons, W. A. Burkhard, and L. H. Landweber, eds.), pp. 383–400, ACM, 1982.

[6] V. Hadzilacos, "A lower bound for byzantine agreement with fail-stop processors, technical report 21-83, department of computer science, harvard university, cambridge, ma," 1983.

[7] L. Lamport and M. J. Fischer, "Byzantine generals and transaction commit protocols, technical report 62, computer science lab., sri international, menlo park, ca," 1982.

[8] J. A. Garay, J. Katz, C.-Y. Koo, and R. Ostrovsky, "Round complexity of authenticated broadcast with a dishonest majority," in *48th FOCS*, pp. 658–668, IEEE Computer Society Press, 2007.

[9] D. Dolev and R. Reischuk, "Bounds on information exchange for byzantine agreement," *J. ACM*, vol. 32, no. 1, pp. 191–204, 1985.

[10] I. Abraham, T. H. Chan, D. Dolev, K. Nayak, R. Pass, L. Ren, and E. Shi, "Communication complexity of byzantine agreement, revisited," *Distributed Comput.*, vol. 36, no. 1, pp. 3–28, 2023.

[11] T.-H. H. Chan, R. Pass, and E. Shi, "Sublinear-round Byzantine agreement under corrupt majority," in *PKC 2020, Part II*, vol. 12111, pp. 246–265, 2020.

[12] M. Borderding, "Levels of authentication in distributed agreement," in *10th International Workshop on Distributed Algorithms WDAG*, pp. 40–55, 1996.

[13] R. Pass and E. Shi, "The sleepy model of consensus," in *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II* (T. Takagi and T. Peyrin, eds.), vol. 10625 of *Lecture Notes in Computer Science*, pp. 380–409, Springer, 2017.

[14] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol with chains of variable difficulty," in *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I* (J. Katz and H. Shacham, eds.), vol. 10401 of *Lecture Notes in Computer Science*, pp. 291–323, Springer, 2017.

[15] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018* (D. Lie, M. Mannan, M. Backes, and X. Wang, eds.), pp. 913–930, ACM, 2018.

[16] J. A. Garay and A. Kiayias, "SoK: A consensus taxonomy in the blockchain era," in *CT-RSA 2020*, vol. 12006 of *LNCS*, pp. 284–318, 2020.

[17] M. Okun and A. Barak, "Efficient algorithms for anonymous byzantine agreement," *Theory Comput. Syst.*, vol. 42, no. 2, pp. 222–238, 2008.

[18] J. Augustine, V. King, A. R. Molla, G. Pandurangan, and J. Saia, "Scalable and secure computation among strangers: Resource-competitive byzantine protocols," *CoRR*, vol. abs/1907.10308, 2019.

[19] P. Khanchandani and R. Wattenhofer, "Byzantine agreement with unknown participants and failures," in *35th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2021, Portland, OR, USA, May 17-21, 2021*, pp. 952–961, IEEE, 2021.

[20] A. Momose and L. Ren, "Constant latency in sleepy consensus," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022* (H. Yin, A. Stavrou, C. Cremers, and E. Shi, eds.), pp. 2295–2308, ACM, 2022.

[21] J. A. Garay, J. Katz, R. Kumaresan, and H.-S. Zhou, "Adaptively secure broadcast, revisited," in *30th ACM PODC*, pp. 179–186, 2011.

[22] J. Wan, H. Xiao, E. Shi, and S. Devadas, "Expected constant round Byzantine broadcast under dishonest majority," in *TCC 2020, Part I*, vol. 12550 of *LNCS*, pp. 381–411, 2020.

[23] J. Wan, H. Xiao, S. Devadas, and E. Shi, "Round-efficient Byzantine broadcast under strongly adaptive and majority corruptions," in *TCC 2020, Part I*, vol. 12550 of *LNCS*, pp. 412–456, 2020.

[24] R. Cohen, J. A. Garay, and V. Zikas, "Completeness theorems for adaptively secure broadcast," in *CRYPTO 2023, Part I*, vol. 14081 of *LNCS*, pp. 3–38, Springer, Heidelberg, 2023.

[25] M. Fitzi and J. B. Nielsen, "On the number of synchronous rounds sufficient for authenticated byzantine agreement," in *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings* (I. Keidar, ed.), vol. 5805 of *Lecture Notes in Computer Science*, pp. 449–463, Springer, 2009.

[26] T. H. Chan, R. Pass, and E. Shi, "Round complexity of byzantine agreement, revisited," *IACR Cryptol. ePrint Arch.*, p. 886, 2019.

[27] M. K. Aguilera and S. Toueg, "A simple bivalency proof that t-resilient consensus requires $t + 1$ rounds," *Inf. Process. Lett.*, vol. 71, no. 3-4, pp. 155–158, 1999.

[28] M. J. Fischer, N. A. Lynch, and M. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[29] Y. Moses and S. Rajsbaum, "The unified structure of consensus: A *Layered Analysis* approach," in *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, PODC '98, Puerto Vallarta, Mexico, June 28 - July 2, 1998* (B. A. Coan and Y. Afek, eds.), pp. 123–132, ACM, 1998.

[30] N. Santoro and P. Widmayer, "Time is not a healer," in *STACS 89, 6th Annual Symposium on Theoretical Aspects of Computer Science, Paderborn, FRG, February 16-18, 1989, Proceedings* (B. Monien and R. Cori, eds.), vol. 349 of *Lecture Notes in Computer Science*, pp. 304–313, Springer, 1989.

[31] "Lecture notes for 6.852: Distributed algorithms fall, 2009," *MIT Open Courseware*. .

[32] D. Dolev, R. Reischuk, and H. R. Strong, "Early stopping in byzantine agreement," *J. ACM*, vol. 37, no. 4, pp. 720–741, 1990.

[33] A. Bar-Noy, D. Dolev, C. Dwork, and H. Raymond Strong, "Shifting gears: Changing algorithms on the fly to expedite byzantine agreement," *Information and Computation*, vol. 97, no. 2, pp. 205–233, 1992.

[34] B. Pfitzmann and M. Waidner, "Information-theoretic pseudosignatures and byzantine agreement for t n / 3 *," 1996.

[35] P. Feldman and S. Micali, "An optimal probabilistic protocol for synchronous byzantine agreement," *SIAM J. Comput.*, vol. 26, no. 4, pp. 873–933, 1997.

[36] J. Katz and C. Koo, "On expected constant-round protocols for byzantine agreement," *J. Comput. Syst. Sci.*, vol. 75, no. 2, pp. 91–112, 2009.

[37] I. Abraham, S. Devadas, D. Dolev, K. Nayak, and L. Ren, "Synchronous byzantine agreement with expected O(1) rounds, expected o(n^2) communication, and optimal resilience," in *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers* (I. Goldberg and T. Moore, eds.), vol. 11598 of *Lecture Notes in Computer Science*, pp. 320–334, Springer, 2019.

[38] T. H. Chan, R. Pass, and E. Shi, "Consensus through herding," *IACR Cryptol. ePrint Arch.*, p. 251, 2019.

[39] J. Loss and J. B. Nielsen, "Early stopping for any number of corruptions," in *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part III* (M. Joye and G. Leander, eds.), vol. 14653 of *Lecture Notes in Computer Science*, pp. 457–488, Springer, 2024.

[40] J. Loss and G. Stern, "Zombies and ghosts: Optimal byzantine agreement in the presence of omission faults," in *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV* (G. N. Rothblum and H. Wee, eds.), vol. 14372 of *Lecture Notes in Computer Science*, pp. 395–421, Springer, 2023.

[41] M. Hirt and U. M. Maurer, "Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract)," in *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997* (J. E. Burns and H. Attiya, eds.), pp. 25–34, ACM, 1997.

[42] C. Koo, "Broadcast in radio networks tolerating byzantine adversarial behavior," in *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004* (S. Chaudhuri and S. Kutten, eds.), pp. 275–282, ACM, 2004.

[43] C. Litsas, A. Pagourtzis, and D. Sakavalas, "A graph parameter that matches the resilience of the certified propagation algorithm," in *Ad-hoc, Mobile, and Wireless Network - 12th International Conference, ADHOC-NOW 2013, Wrocław, Poland, July 8-10, 2013. Proceedings* (J. Cichon, M. Gebala, and M. Klonowski, eds.), vol. 7960 of *Lecture Notes in Computer Science*, pp. 269–280, Springer, 2013.

[44] A. Pagourtzis, G. Panagiotakos, and D. Sakavalas, "Reliable broadcast with respect to topology knowledge," in *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings* (F. Kuhn, ed.), vol. 8784 of *Lecture Notes in Computer Science*, pp. 107–121, Springer, 2014.

[45] A. Pagourtzis, G. Panagiotakos, and D. Sakavalas, "Reliable communication via semilattice properties of partial knowledge," in *Fundamentals of Computation Theory - 21st International Symposium, FCT 2017, Bordeaux, France, September 11-13, 2017, Proceedings* (R. Klasing and M. Zeitoun, eds.), vol. 10472 of *Lecture Notes in Computer Science*, pp. 367–380, Springer, 2017.